

LỜI MỞ ĐẦU



Khi bắt đầu làm quen với ngôn ngữ lập trình – Cụ thể là ngôn ngữ C – Sinh Viên thường gặp khó khăn trong việc chuyển vấn đề lý thuyết sang cài đặt cụ thể trên máy. Sách “**Giáo Trình Bài Tập Kỹ Thuật Lập Trình**” nhằm cung cấp cho các Học Sinh - Sinh Viên Trường CĐ Công Nghệ Thông Tin Tp. Hồ Chí Minh hệ thống các bài tập, những **kỹ năng thực hành cơ bản và nâng cao về ngôn ngữ lập trình C**. Cuốn sách này được xem như tài liệu hướng dẫn từng bước cho Học Sinh - Sinh Viên của Trường trong việc học và áp dụng kiến thức lý thuyết trên lớp một cách thành thạo và sâu rộng.

Giáo trình được chia thành **10 chương** theo từng nội dung kiến thức, kèm theo **Các đề thi mẫu** và **1 phụ lục** hướng dẫn viết chương trình, chuẩn đoán lỗi và sửa lỗi. Mỗi chương gồm 2 phần:

- ❖ **Phần lý thuyết:** được tóm tắt ngắn gọn với đầy đủ ví dụ minh họa kèm theo.
- ❖ **Phần bài tập:** với nhiều bài tập được chia làm hai mức độ cơ bản và luyện tập nâng cao, bài tập có đánh dấu * là bài tập khó dành cho sinh viên luyện tập thêm.
- ❖ **Phần kết luận:** Tóm tắt nội dung và các thao tác mà sinh viên cần nắm hay những lưu ý của chương đó.

Trong quá trình biên soạn, chúng tôi đã cố gắng trích lọc những kiến thức rất cơ bản, những lỗi hay gặp đối với người mới lập trình. Bên cạnh đó chúng tôi cũng bổ sung thêm một số bài tập nâng cao để rèn luyện thêm kỹ năng lập trình.

Tuy nhiên, chủ đích chính của giáo trình này là phục vụ cho một môn học nên chắc chắn không thể tránh khỏi những thiếu sót, vì thế, rất mong nhận được những góp ý quý báu của các thầy cô, các đồng nghiệp và các bạn Học Sinh – Sinh Viên để giáo trình này ngày càng hoàn thiện hơn.

Chân thành cảm ơn.

LỊCH TRÌNH THỰC HÀNH



Tổng thời gian: 90 tiết.

| STT | NỘI DUNG | SỐ TIẾT |
|-----|---|-----------|
| 1 | Chương 1: Lưu đồ thuật toán | 03 |
| 2 | Chương 2: Cấu trúc điều khiển | 06 |
| 3 | Chương 3: Hàm con | 12 |
| 4 | Chương 4: Mảng một chiều | 24 |
| 5 | Chương 5: Chuỗi ký tự | 06 |
| 6 | Chương 6: Mảng hai chiều | 12 |
| 7 | Chương 7: Kiểu dữ liệu có cấu trúc | 12 |
| 8 | Chương 8: Tập tin | 06 |
| 9 | Chương 9: Đệ qui | 06 |
| 10 | Chương 10: Hướng dẫn lập trình bằng phương pháp Project | 03 |

CHƯƠNG 1 LƯU ĐỒ THUẬT TOÁN (FLOWCHART)

Các ký hiệu biểu diễn lưu đồ thuật toán, cách biểu diễn các cấu trúc điều khiển rẽ nhánh, cấu trúc lặp và các kỹ thuật liên quan đến lưu đồ thuật toán.

I. TÓM TẮT LÝ THUYẾT

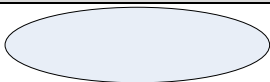


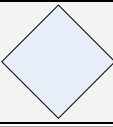

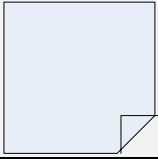

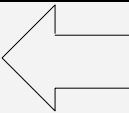

I.1. Khái niệm

Lưu đồ thuật toán là công cụ dùng để **biểu diễn thuật toán**, việc mô tả **nhập** (input), dữ liệu **xuất** (output) và luồng xử lý thông qua các **ký hiệu hình học**.

I.2. Phương pháp duyệt

- Duyệt từ trên xuống.
- Duyệt từ trái sang phải.

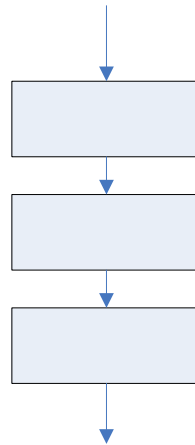
I.3. Các ký hiệu

| STT | KÝ HIỆU | DIỄN GIẢI |
|-----|---|---|
| 1 |  | Bắt đầu chương trình |
| 2 |  | Kết thúc chương trình |
| 3 |  | Luồng xử lý |
| 4 |  | Điều khiển lựa chọn |
| 5 |  | Nhập |
| 6 |  | Xuất |
| 7 |  | Xử lý, tính toán hoặc gán |
| 8 |  | Trả về giá trị (return) |
| 9 |  | Điểm nối liên kết tiếp theo (Sử dụng khi lưu đồ vượt quá trang) |

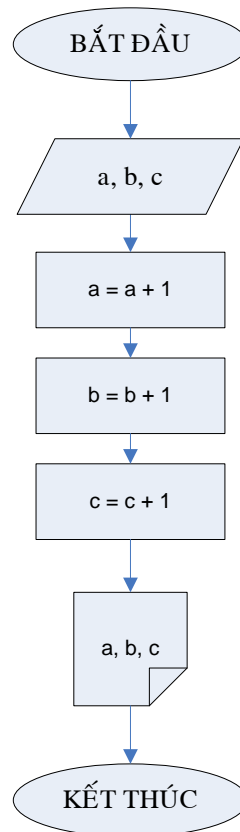
I.4. Các cấu trúc điều khiển cơ bản

a. Cấu trúc tuần tự

Tuần tự thực thi tiến trình. Mỗi lệnh được thực thi theo một chuỗi **từ trên xuống**, xong lệnh này rồi **chuyển xuống** lệnh kế tiếp.



Ví dụ: Nhập vào 3 số nguyên a, b, c và xuất ra màn hình với giá trị của mỗi số tăng lên 1.

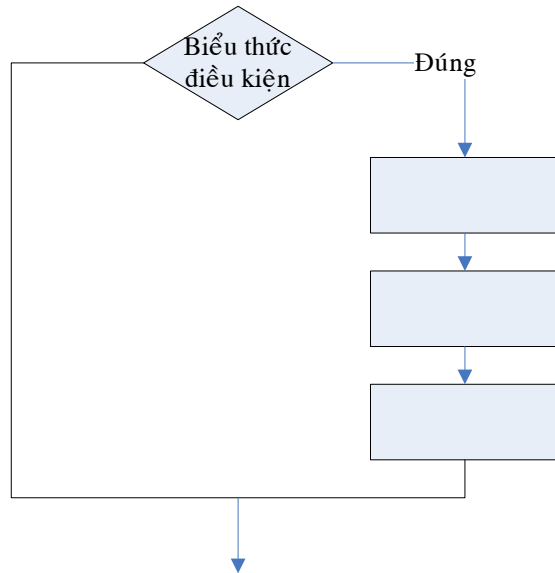


b. Cấu trúc lựa chọn

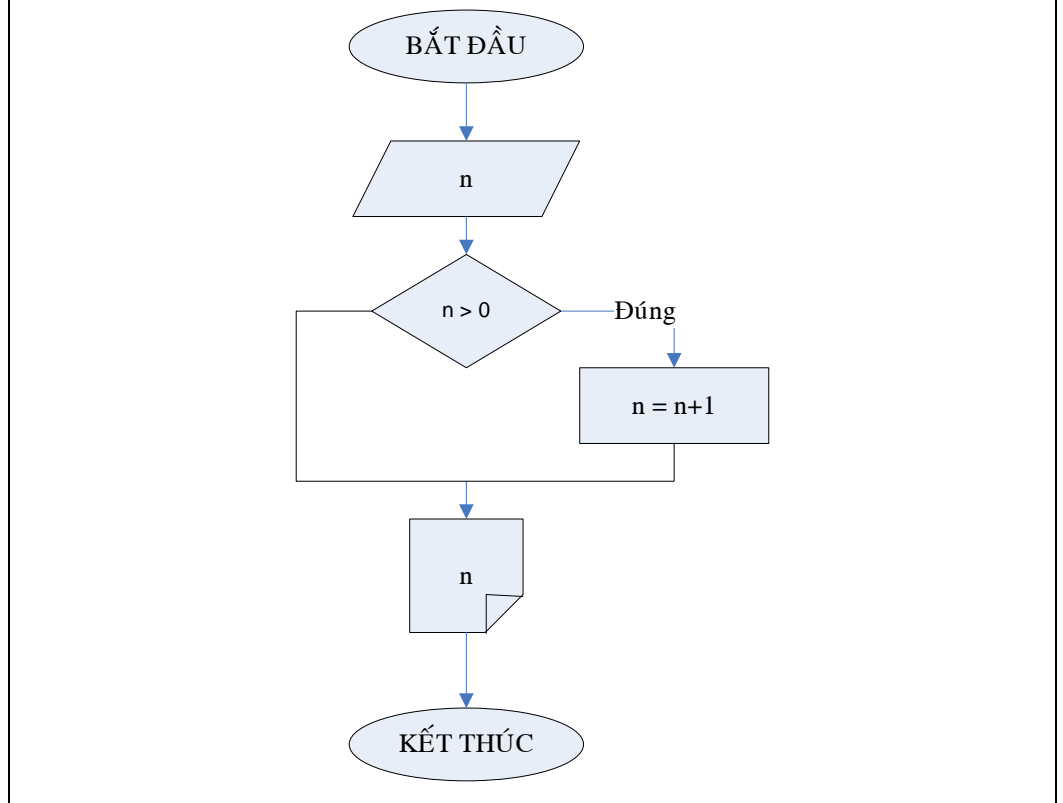
Điểm quyết định cho phép **chọn một trong hai trường hợp**.

- **if**

Chỉ xét trường hợp đúng.

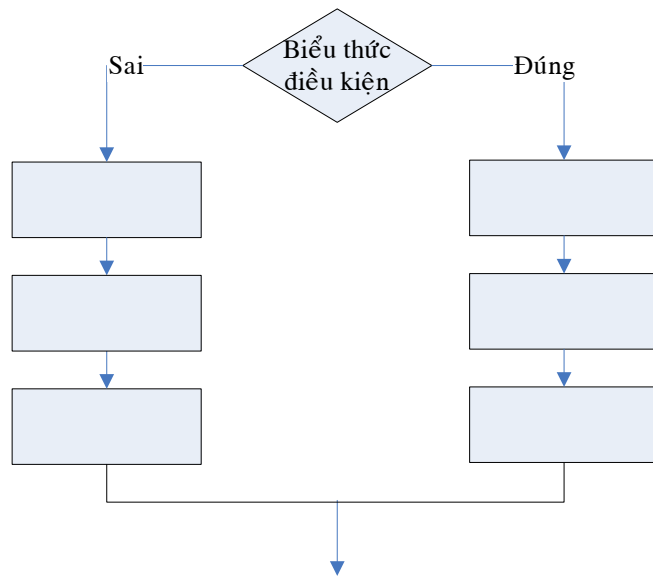


Vi dụ: Nhập vào số nguyên n . Kiểm tra nếu $n > 0$ tăng n lên 1 đơn vị. Xuất kết quả.

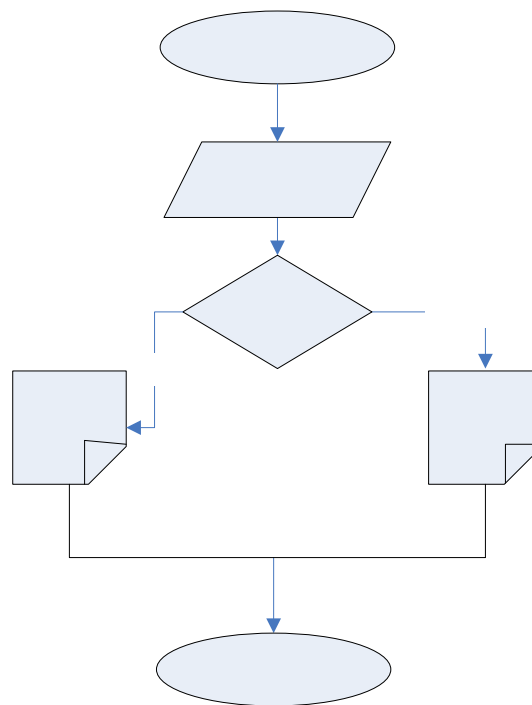


- **if...else**

Xét trường hợp đúng và trường hợp sai.



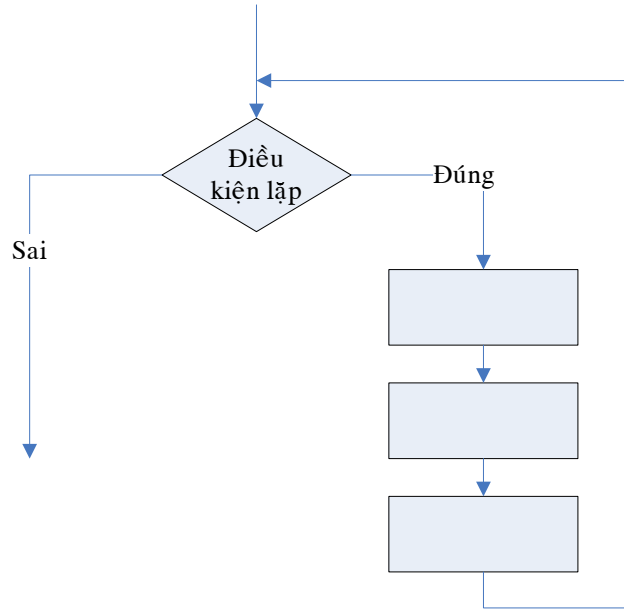
Ví dụ: Nhập vào số nguyên n. Kiểm tra nếu n chẵn xuất ra màn hình “n chẵn”, ngược lại xuất “n lẻ”.



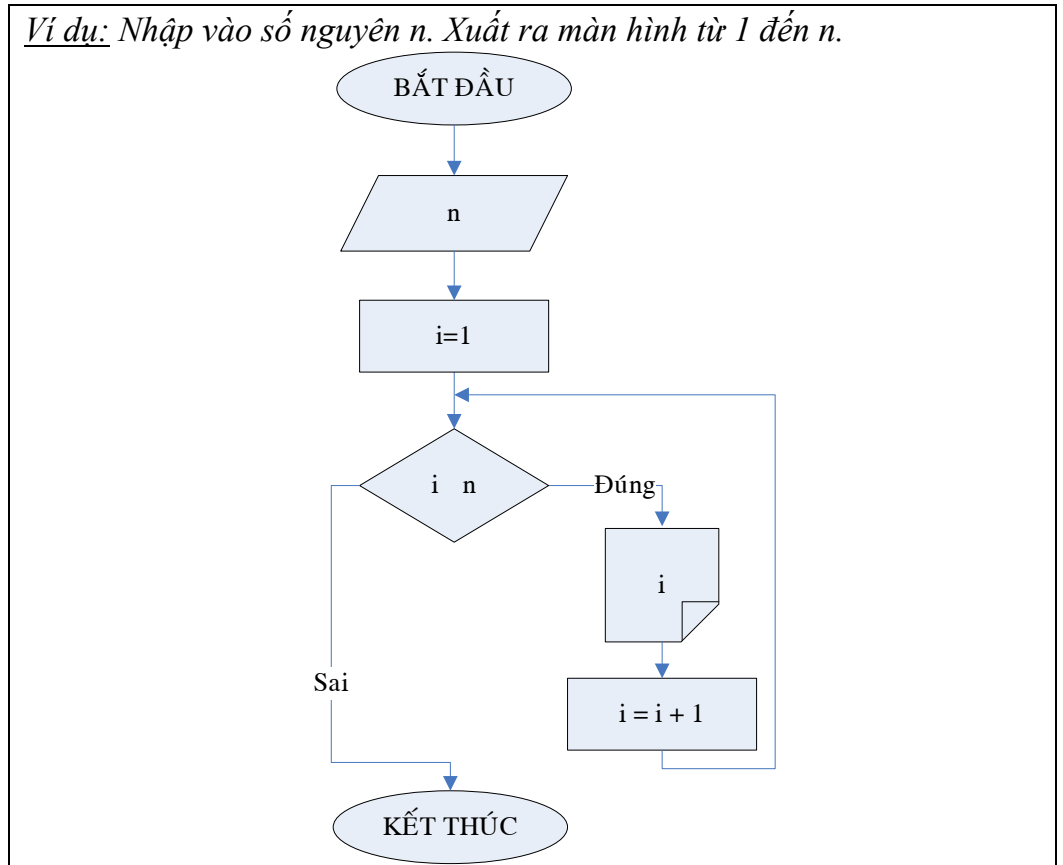
c. Cấu trúc lặp

Thực hiện liên tục 1 lệnh hay tập lệnh với số lần lặp dựa vào điều kiện. Lặp sẽ kết thúc khi điều kiện được thỏa.

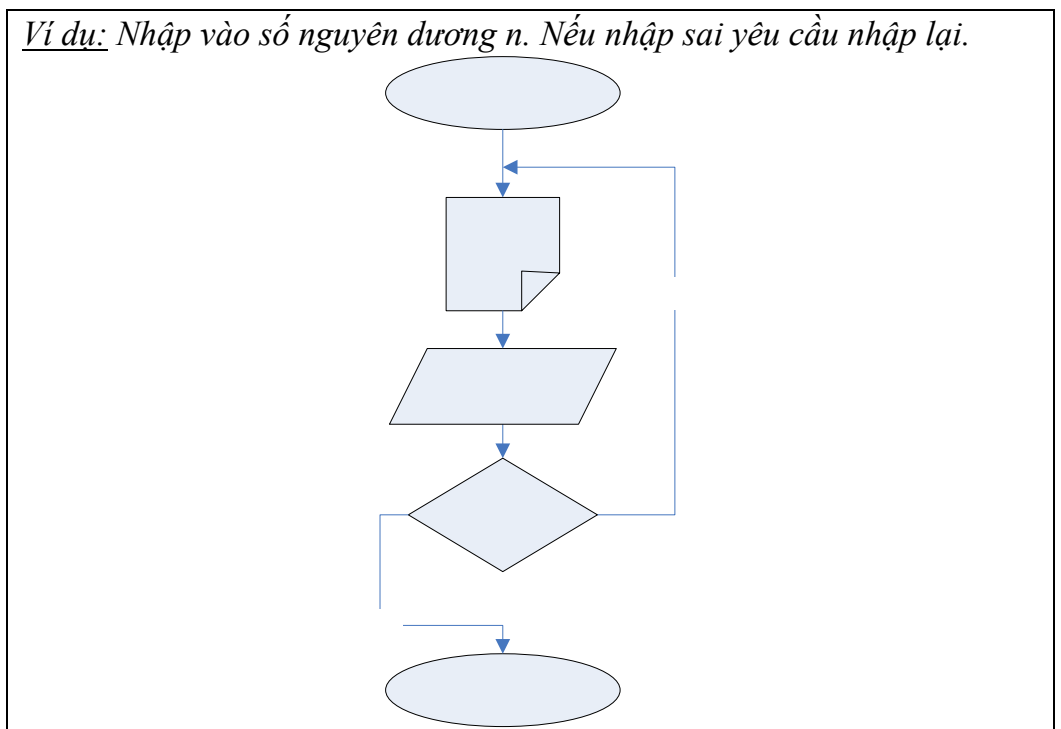
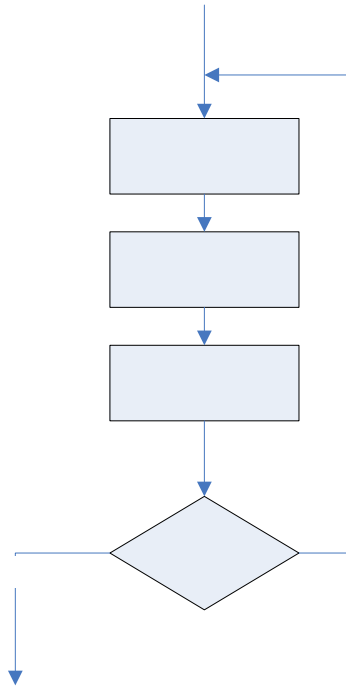
- **for / while (Kiểm tra điều kiện trước khi lặp)**



Ví dụ: Nhập vào số nguyên n. Xuất ra màn hình từ 1 đến n.

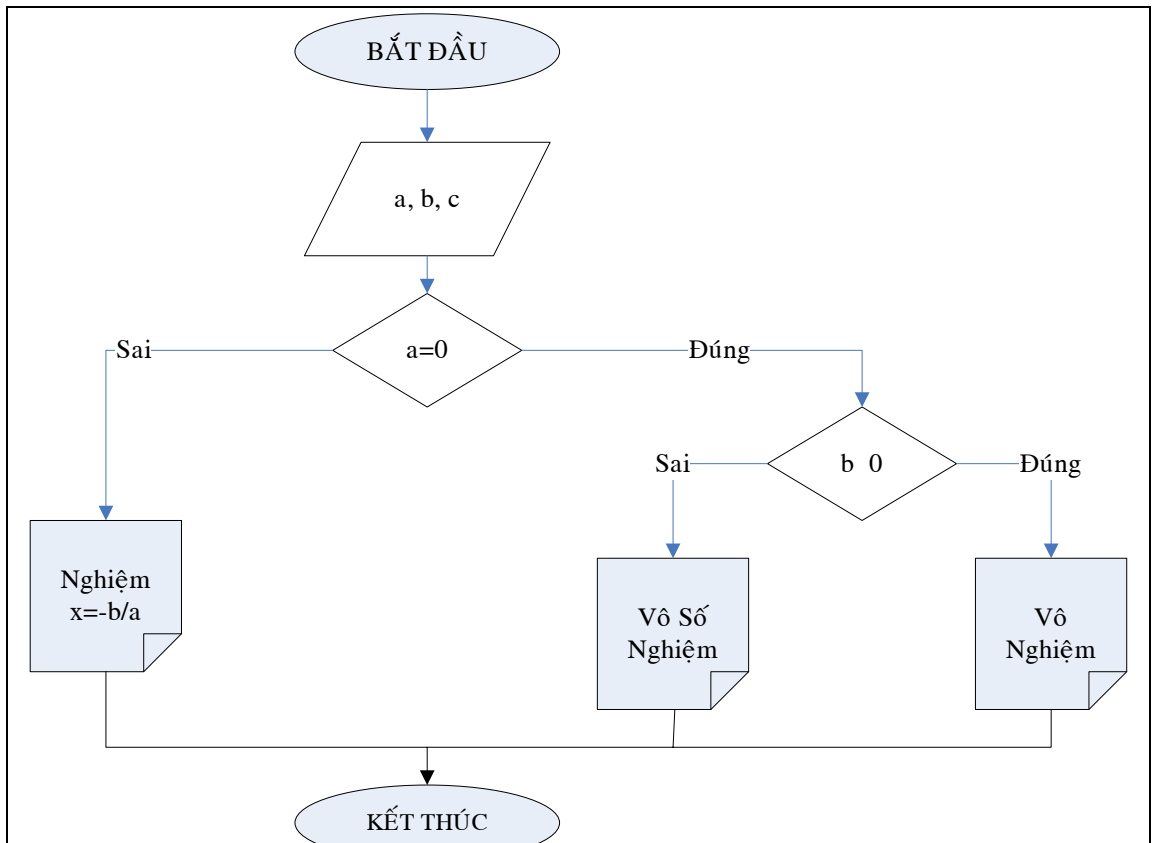


- do ... while (Thực hiện lặp trước khi kiểm tra điều kiện)

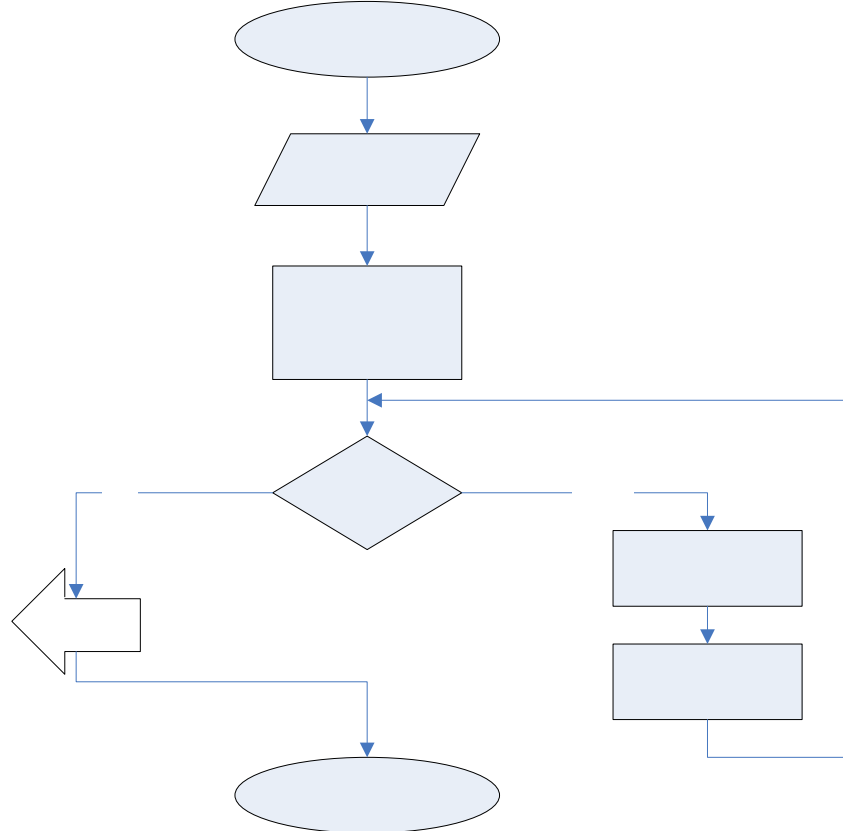


d. Các ví dụ

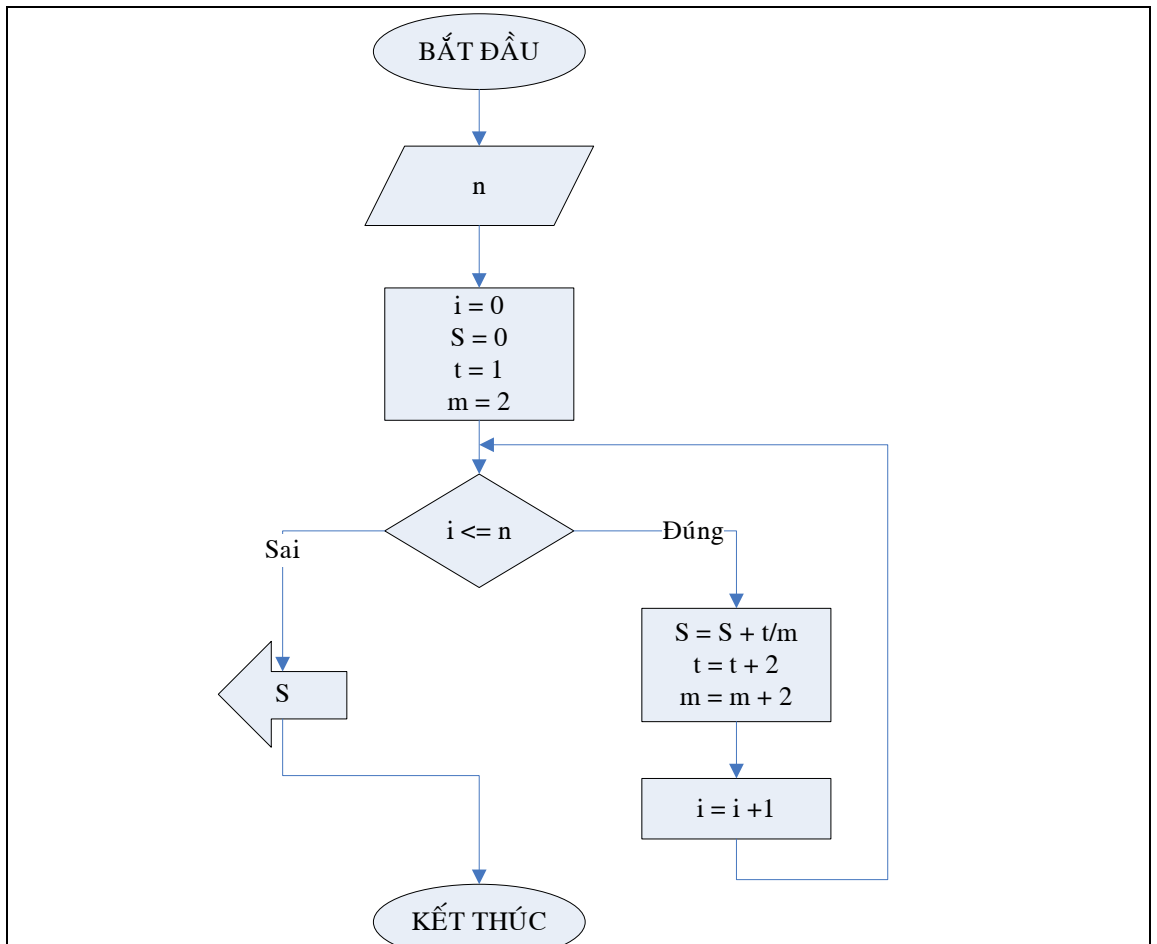
Ví dụ 1: Giải và biện luận phương trình: $ax+b=0$.



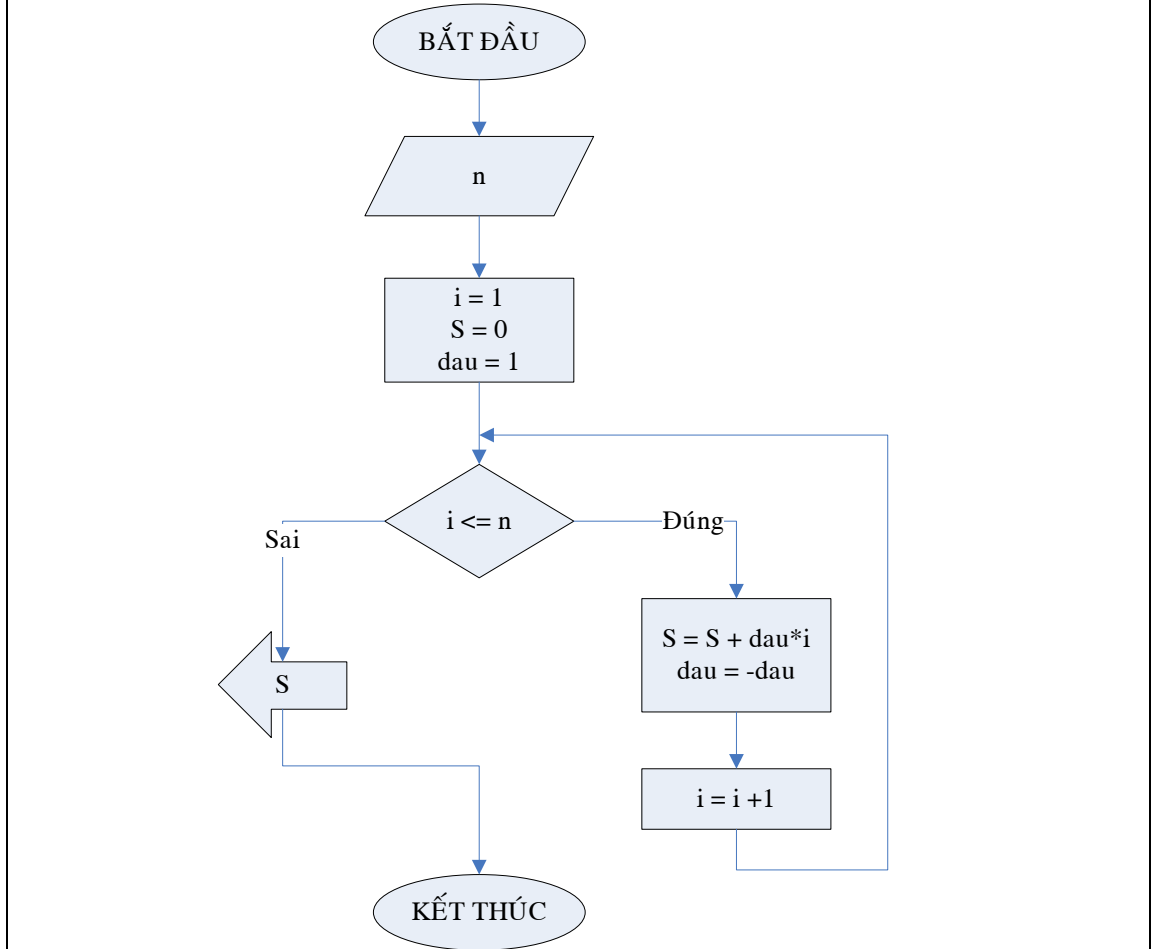
Ví dụ 2: Tính tổng: $S = 1 + 2 + 3 + \dots + n$, với $n > 0$



Ví dụ 3: Tính tổng: $S(n) = \frac{1}{2} + \frac{3}{4} + \frac{5}{6} + \dots + \frac{2n+1}{2n+2}$, với $n > 0$



Ví dụ 4: Tính tổng: $S(n) = 1 - 2 + 3 - 4 + \dots + (-1)^{n+1}n$, với $n > 0$



II. BÀI TẬP

Vẽ lưu đồ thuật toán sau

II.1. Bài tập cơ bản

1. Nhập vào hai số x, y . Xuất ra màn hình tổng, hiệu, tích, thương của hai số trên.
2. Nhập vào số nguyên n , kiểm tra xem n chẵn hay lẻ và xuất ra màn hình.
3. Nhập vào ba cạnh a, b, c của tam giác. Xuất ra màn hình tam giác đó thuộc loại tam giác gì? (Thường, cân, vuông, đều hay vuông cân).
4. Nhập vào số nguyên n . Xuất ra n màn hình (Nếu n chẵn thì gấp đôi giá trị).
5. Nhập vào số nguyên n . Nếu $n > 5$ thì tăng n lên 2 đơn vị và trả về giá trị n , ngược lại trả về giá trị 0.
6. Tính $n!$, với $n \geq 0$
7. Tính $P(n) = 1.3.5 \dots (2n+1)$, với $n \geq 0$
8. Tính $S(n) = 1 + 3 + 5 + \dots + (2 \times n + 1)$, với $n \geq 0$
9. Tính $S(n) = 1 - 2 + 3 - 4 + \dots + (-1)^{n+1} n$, với $n > 0$
10. Tính $S(n) = 1 + 1.2 + 1.2.3 + \dots + 1.2.3 \dots n$, với $n > 0$
11. Tính $S(n) = 1^2 + 2^2 + 3^2 + \dots + n^2$, với $n > 0$
12. Tính $S(n) = 1 + \frac{1}{2} + \frac{1}{3} + \dots + \frac{1}{n}$, với $n > 0$
13. (*) Tính $S(n) = 1 + \frac{1}{1+2} + \frac{1}{1+2+3} + \dots + \frac{1}{1+2+3+\dots+n}$, với $n > 0$
14. Tính $P(x, y) = x^y$.
15. Tính $S(n) = 1 + (1+2) + (1+2+3) + \dots + (1+2+3+\dots+n)$, với $n > 0$
16. Cho số nguyên n . Tính trị tuyệt đối của n .
17. Cho số nguyên dương n gồm k chữ số. Tìm chữ số có giá trị lớn nhất.
18. Đếm số lượng ước số chẵn của số nguyên dương n .
19. In ra chữ số đầu tiên của số nguyên dương n gồm k chữ số.
20. Cho 2 số nguyên dương a, b . Tìm USCLN của a và b .
21. Cho 2 số nguyên dương a, b . Tìm BSCNN của a và b .
22. Cho số nguyên dương x . Kiểm tra xem x có phải là số nguyên tố không?
23. Cho số nguyên dương x . Kiểm tra x có phải là số chính phương không?
24. Cho số nguyên dương x . Kiểm tra xem x có phải là số hoàn thiện không?

II.2. Bài tập luyện tập và nâng cao

25. Tính $S(n) = 1 + 2^2 + 3^3 + \dots + n^n$, với $n \geq 0$

26. Tính $S(n) = \frac{1}{2} + \frac{2}{3} + \frac{3}{4} + \dots + \frac{n}{n+1}$, với $n > 0$

27. Tính $S(n) = 1 + \frac{1}{2!} + \frac{1}{3!} + \dots + \frac{1}{n!}$, với $n > 0$

28. Tính $S(n) = 1 + \frac{1+2}{2!} + \frac{1+2+3}{3!} + \dots + \frac{1+2+3+\dots+n}{n!}$, với $n > 0$

29. Giải và biện luận phương trình: $ax^2 + bx + c = 0$

30. Giải và biện luận phương trình: $ax^4 + bx^2 + c = 0$

31. (*) Tính $S(n) = \sqrt{n + \sqrt{(n-1) + \sqrt{(n-2) + \dots + \sqrt{1}}}}$, với $n > 0$

32. (***) Tính $S(n) = \sqrt{1 + \sqrt{2 + \sqrt{3 + \dots + \sqrt{n}}}}$, với $n > 0$

III. KẾT LUẬN

Lưu đồ thuật toán rất hữu ích trong việc mô tả cách giải quyết của một bài toán. Việc mô tả này rất trực quan thông qua các ký hiệu hình học, đây là giai đoạn đầu tiên trước khi bắt tay vào lập trình trên một ngôn ngữ lập trình cụ thể.

Khi xây dựng lưu đồ thuật toán, chúng ta cần chú ý một vài điểm sau:

- ❖ Một lưu đồ phải có điểm **bắt đầu** và điểm **kết thúc** (điều kiện kết thúc).
- ❖ Phải có **dữ liệu vào**, **dữ liệu ra** sau khi xử lý tính toán.
- ❖ Tại mỗi vị trí quyết định lựa chọn rẽ nhánh phải ghi rõ điều kiện **đúng hoặc sai** thì đi theo nhánh nào.

CHƯƠNG 2 CẤU TRÚC ĐIỀU KHIỂN

Tìm hiểu và cài đặt các cấu trúc rẽ nhánh, lựa chọn, lặp và các ký hiệu phép toán trong ngôn ngữ C. Mô tả cách hoạt động và hướng dẫn chạy từng bước chương trình.

I. TÓM TẮT LÝ THUYẾT**I.1. Các ký hiệu**

| STT | KÝ HIỆU | DIỄN GIẢI | VÍ DỤ |
|-----|----------|---|--|
| 1 | { } | Bắt đầu và kết thúc hàm hay khối lệnh. | void main() { } |
| 2 | ; | Kết thúc khai báo biến, một lệnh, một lời gọi hàm, hay khai báo nguyên mẫu hàm. | int x; voidNhapMang(int a[], int &n); |
| 3 | // | Chú thích (ghi chú) cho một dòng. Chỉ có tác dụng đối với người đọc chương trình. | //Ham nay dung de nhap mang voidNhapMang(int a[], int &n); |
| 4 | /* */ | Tương tự như ký hiệu //, nhưng cho trường hợp nhiều dòng. | /* Dau tien nhap vao n. Sau do nhap cho tung phan tu */ voidNhapMang(int a[], int &n); |

I.2. Các kiểu dữ liệu cơ bản trong C

| STT | KIỂU | GHI CHÚ | KÍCH THƯỚC | ĐỊNH DẠNG |
|---------------------------------|---------------|-----------------|------------|-----------|
| KIỂU LIÊN TỤC (SỐ THỰC) | | | | |
| 1 | float | | 4 bytes | %f |
| 2 | double | | 8 bytes | %lf |
| 3 | long double | | 10 bytes | %lf |
| KIỂU RỜI RẠC (SỐ NGUYÊN) | | | | |
| 1 | char | Ký tự | 1 byte | %c |
| | | Số nguyên | 1 byte | %d |
| 2 | unsigned char | Số nguyên dương | 1 byte | %d |
| 3 | int | Số nguyên | 2 bytes | %d |
| 4 | unsigned int | Số nguyên dương | 2 bytes | %u |
| 5 | long | Số nguyên | 4 bytes | %ld |
| 6 | unsigned long | Số nguyên dương | 4 bytes | %lu |
| 7 | char * | Chuỗi | | %s |

I.3. Bảng ký hiệu các phép toán

| STT | PHÉP TOÁN | Ý NGHĨA | GHI CHÚ |
|------------------------------------|-----------|----------------------|--|
| PHÉP TOÁN SỐ HỌC | | | |
| 1 | + | Cộng | |
| 2 | - | Trừ | |
| 3 | * | Nhân | |
| 4 | / | Chia lấy phần nguyên | |
| 5 | % | Chia lấy phần dư | |
| PHÉP TOÁN QUAN HỆ | | | |
| 1 | > | Lớn hơn | |
| 2 | < | Nhỏ hơn | |
| 3 | >= | Lớn hơn hoặc bằng | |
| 4 | <= | Nhỏ hơn hoặc bằng | |
| 5 | == | Bằng nhau | |
| 6 | != | Khác nhau | |
| PHÉP TOÁN LOGIC | | | |
| 1 | ! | NOT | |
| 2 | && | AND | |
| 3 | | OR | |
| TOÁN TỬ TĂNG GIẢM | | | |
| 1 | ++ | Tăng 1 | Nếu toán tử tăng giảm đặt trước thì tăng giảm trước rồi tính biểu thức hoặc ngược lại. |
| 2 | -- | Giảm 1 | |
| PHÉP TOÁN THAO TÁC TRÊN BIT | | | |
| 1 | & | AND | |
| 2 | | OR | |
| 3 | ^ | XOR | |
| 4 | << | Dịch trái | |
| 5 | >> | Dịch phải | |
| 6 | ~ | Lấy phần bù theo bit | |

I.4. Các hàm cơ bản

| STT | TÊN HÀM | THƯ VIỆN | DIỄN GIẢI | VÍ DỤ |
|-----|-----------|-------------------|---|---|
| 1 | printf | #include<stdio.h> | Xuất ra màn hình. | <pre>#include<stdio.h> #include<conio.h> #include<dos.h> void main() { int c = 1, n; clrscr(); printf("Nhap n:"); scanf("%d", &n); do{ textcolor(c); gotoxy(20, 10); cprintf("%d", n); c++; if (c>15) c = 1; delay(200); } while(!kbhit()); }</pre> |
| 2 | scanf | #include<stdio.h> | Lấy dữ liệu từ bàn phím. | |
| 3 | gotoxy | #include<conio.h> | Di chuyển dấu nháy đến tọa độ (x, y) trên màn hình văn bản. | |
| 4 | textcolor | #include<conio.h> | Đặt màu cho chữ (có giá trị từ 0 đến 15). | |
| 5 | cprintf | #include<stdio.h> | Xuất ra màn hình với màu chữ đã định liền trước đó. | |
| 6 | delay | #include<dos.h> | Dừng thực hiện lệnh tiếp sau một khoảng thời gian. | |
| 7 | kbhit | #include<conio.h> | Kiểm tra xem có nhấn phím. | |

I.5. Cấu trúc rẽ nhánh

a. Cấu trúc if

```
if (biểu thức điều kiện)
{
    <khối lệnh> ;
}
```

Nếu biểu thức điều kiện cho kết quả khác không thì thực hiện khối lệnh.

Ví dụ:

```
#include <conio.h>
#include <stdio.h>
void main ()
{
    float number ;

    printf ( "Nhap mot so trong khoang tu 1 den 10 => " );
    scanf ( "%f", &number );
    if (number >5)
        printf ( "So ban nhap lon hon 5. \n" );
    printf ( "%f la so ban nhap. ", number );
}
```

b. Cấu trúc if ... else

```

if (biểu thức điều kiện)
{
    <khối lệnh 1>;
}
else
{
    <khối lệnh 2>;
}

```

Nếu biểu thức điều kiện cho kết quả khác không thì thực hiện khối lệnh 1, ngược lại thì cho thực hiện khối lệnh thứ 2. Biểu thức điều kiện phải đặt trong cặp dấu ngoặc tròn.

Ví dụ: Giải và biện luận phương trình: $ax+b=0$

```

#include <conio.h>
#include <stdio.h>
void main ()
{
    float a, b;
    printf ( "\n Nhập vào a:");
    scanf ( "%f", &a);
    printf ( " Nhập vào b:");
    scanf ( "%f", &b) ;
    if (a== 0)
        if (b== 0)
            printf ( " \n PTVSN");
        else
            printf ( " \n PTVN");
    else
        printf ( " \n Nghiệm x=%f", -b/a);
    getch ();
}

```

I.6. Cấu trúc lựa chọn switch

```

switch (biểu thức)
{
    case n1:
        các câu lệnh ;
        break ;
    case n2:
        các câu lệnh ;
        break ;
    .....
    case nk:
        <các câu lệnh> ;
        break ;
}

```



```
[default:  các câu lệnh]
```

```
}
```

- n_i là các **hằng số nguyên hoặc ký tự**.
- Phụ thuộc vào giá trị của biểu thức viết sau **switch**, nếu:
 - Giá trị này = n_i thì thực hiện câu lệnh sau case n_i .
 - Khi giá trị biểu thức không thỏa tất cả các n_i thì thực hiện câu lệnh sau **default** nếu có, hoặc thoát khỏi câu lệnh **switch**.
 - Khi chương trình đã thực hiện xong câu lệnh của **case** n_i nào đó thì nó sẽ thực hiện luôn các lệnh thuộc **case** bên dưới nó mà không xét lại điều kiện (do các n_i được xem như các nhãn) → Vì vậy, để chương trình thoát khỏi lệnh **switch** sau khi thực hiện xong một trường hợp, ta dùng lệnh **break**.

Ví dụ: Tạo menu cấp 1 cho phép chọn menu bằng số nhập từ bàn phím.

```
#include<stdio.h>
#include<conio.h>
int ChonTD ()
{
    int chon ;

    printf ("Thuc Don") ;
    printf ("\n1. Lau thai!") ;
    printf ("\n2. Nuoc ngot!") ;
    printf ("\n3. Ca loc hap bau!") ;
    printf ("\n4. Chuot dong!") ;
    printf ("\n Xin moi ban chon mon an!") ;
    scanf ("%d",&chon) ;
    return chon ;
}

void TDchon(int chon)
{
    switch (chon)
    {
        case 1:
            printf ("\nBan chon lau thai!") ;
            break ;
        case 2:
            printf ("\nBan chon nuoc ngot!") ;
            break ;
        case 3:
            printf ("\nBan chon ca loc hap bau!") ;
            break ;
        case 4:
            printf ("\nBan chon chuot dong!") ;
```

```

        break ;
    default:
        printf ("\nBan chon khong dung!");
    }
}

void main()
{
    clrscr() ;
    int c ;
    c=ChonTD() ;
    TDchon(c) ;
    getch() ;
}

```

I.7. Cấu trúc lặp

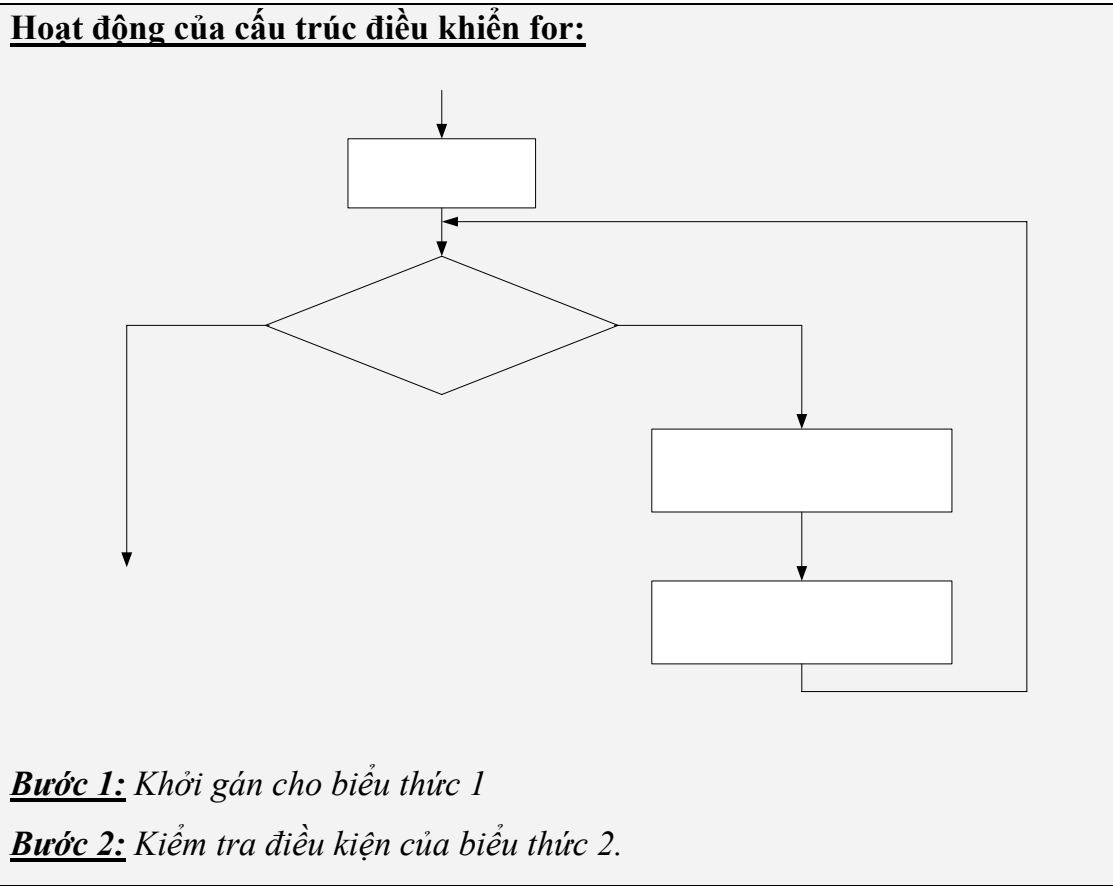
a. for

```

for (<biểu thức khởi gán>; <biểu thức điều kiện>; <biểu thức tăng/giảm>)
{
    <khối lệnh>;
}

```

Bất kỳ biểu thức nào trong 3 biểu thức nói trên đều có thể vắng nhưng phải giữ dấu chấm phẩy (;).



- Nếu **biểu thức 2** $\neq 0$ thì cho thực hiện các lệnh của vòng lặp, thực hiện biểu thức 3. Quay trở lại bước 2.
- Ngược lại thoát khỏi lặp.

Ví dụ: In ra màn hình bảng mã ASCII từ ký tự số 33 đến 255.

```
#include<conio.h>
#include<stdio.h>
void main()
{
    for (int i=33;i<=255;i++)
        printf("Ma ASCII cua %c: %d\t", i, i) ;
    getch () ;
}
```

b. while

```
< Khởi gán>
while ( <biểu thức điều kiện> )
{
    lệnh/ khối lệnh;
    < tăng/giảm chỉ số lặp>;
}
```

⚠ **Lưu ý:** Cách hoạt động của **while** giống **for**

Ví dụ: Tính giá trị trung bình các chữ số của số nguyên n gồm k chữ số.

```
#include<stdio.h>
#include<conio.h>
void main()
{
    long n, tong=0;
    int sochuso=0;
    float tb;

    printf ("Nhap vao gia tri n gom k chu so" );
    scanf ("%ld",&n) ;
    while(n>0)
    {
        tong=tong+n%10 ;
        sochuso++ ;
        n=n/10 ;
    }

    tb=1.0*tong/sochuso ;
    printf ("Gia tri trung binh la: %f", tb) ;
}
```

```

    getch () ;
}

```

c. *do ... while*

```

do
{
    < khối lệnh> ;
} while (biểu thức điều kiện) ;

```

Thực hiện khối lệnh cho đến khi biểu thức có giá trị bằng 0.

Ví dụ: Nhập ký tự từ bàn phím hiển thị lên màn hình mã ASCII của ký tự đó, thực hiện đến khi nhấn phím ESC (Mã ASCII của phím ESC là 27).

```

#include<stdio.h>
#include<conio.h>

void main()
{
    int ma ;
    do {
        ma=getch ();
        if (ma !=27)
            printf ("Ma ASCII %c:%d\t", ma, ma);
    }while (ma!=27) ;

    getch () ;
}

```

⊗ Lặp *while* kiểm tra điều kiện trước khi thực hiện lặp, còn vòng lặp *do...while* thực hiện lệnh lặp rồi mới kiểm tra điều kiện. Do đó vòng lặp *do...while* thực hiện lệnh ít nhất một lần.

I.8. break và continue

a. *break*

Dùng để kết thúc vòng lặp trực tiếp chứa nó khi thỏa điều kiện nào đó.

Ví dụ: Cho phép người dùng nhập liên tục giá trị n cho đến khi nhập âm thì dừng.

```

#include<stdio.h>
#include<conio.h>

void main()
{

```

```

while (1)
{
    printf("\nNhập n: ");
    scanf("%d", &n);
    if(n<0)
        break;
}

getch ();
}

```

b. *continue*

Dùng để bỏ qua một lần lặp khi thỏa điều kiện nào đó.

Ví dụ: In ra màn hình giá trị từ 10 đến 20 trừ đi số 13 và số 17.

```

#include<stdio.h>
#include<conio.h>

void main()
{
    for(int i=10 ; i<=20; i++)
    {
        if(i==13||i==17)
            continue;
        printf("%d\t", i);
    }
    getch ();
}

```

II. BÀI TẬP

II.1. Phương pháp chạy tay từng bước để tìm kết quả chương trình

- ❖ Xác định chương trình có sử dụng những biến nào.
- ❖ **Giá trị ban đầu** của mỗi biến.
- ❖ Những **biến nào sẽ bị thay đổi** trong quá trình chạy chương trình thì lập thành bảng có dạng sau:

| Bước (Hoặc lần thực hiện) | Biến 1 | Biến 2 | ... | Biến n | Kết quả in ra màn hình |
|------------------------------|-----------|-----------|-----|-----------|---------------------------|
| 0 | Giá trị 0 | Giá trị 0 | ... | Giá trị 0 | |
| 1 | Giá trị 1 | Giá trị 1 | ... | Giá trị 1 | |
| 2 | Giá trị 2 | Giá trị 2 | ... | Giá trị 2 | |
| ... | ... | ... | ... | ... | |
| ... | ... | ... | ... | ... | |

☞ *Lưu ý từng lệnh và biểu thức điều kiện trong đoạn chương trình*

Ví dụ: Cho biết kết quả của đoạn chương trình sau:

```
void main()
{
    int i, a = 4;
    clrscr();
    for(i = 0 ; i<a; i++)
        printf("%d\n", i);
}
```

Chương trình gồm 2 biến i và a, chỉ có biến i có giá trị thay đổi trong quá trình chạy chương trình nên ta lập bảng sau:

a có giá trị là 4

| Bước thực hiện | Giá trị của biến i | Kết quả in ra màn hình |
|----------------|--------------------|------------------------|
| 0 | 0 | 0 |
| 1 | 1 | 0 1 |
| 2 | 2 | 0 1 2 |
| 3 | 3 | 0 1 2 3 |
| 4 | 4 | |

Tại bước 4, giá trị của i = 4 vi phạm điều kiện lặp (i<a) nên vòng lặp kết thúc. Do đó kết quả in ra màn hình:

0
1
2
3

II.2. Bài tập cơ bản

a. Cấu trúc if / if..else và switch

1. Cho biết kết quả của đoạn chương trình sau:

```
int a=9, b=6;
a++;
a=a+b--;
a=a+(-b);
```

```
if(a%2==0)
    printf("Gia tri cua a la chan");
printf("Tong cua a va b la: %d", a+b) ;
```

2. Cho biết kết quả của đoạn chương trình sau:

```
int a=7, b=8;
a++;
a=a+(b--);
--b;
a--;
a=(-a)+(-b);
if(a%2!=0)
    printf("\n a la so le");
else
    printf("\n a la so chan");
printf("\na = %d",a);
```

3. Cho biết kết quả của đoạn chương trình sau:

```
int x=5, y;
y=x++ + 5;
printf("x=%d, y=%d\n", x, y);
y*=6;
x=y%7;
printf("x=%d,y=%d,y/x=%d", x, y, y/x);
```

4. Nhập vào hai số nguyên a, b. In ra màn hình giá trị lớn nhất.
5. Cho ba số a, b, c đọc vào từ bàn phím. Hãy tìm giá trị lớn nhất của ba số trên và in ra kết quả.
6. Cho ba số a, b, c đọc vào từ bàn phím. Hãy in ra màn hình theo thứ tự tăng dần các số. (Chỉ được dùng thêm hai biến phụ).
7. Viết chương trình nhập vào một số nguyên n gồm ba chữ số. Xuất ra màn hình chữ số lớn nhất ở vị trí nào?
Ví dụ: $n=291$. Chữ số lớn nhất nằm ở hàng chục (9).
8. Viết chương trình nhập vào số nguyên n gồm ba chữ số. Xuất ra màn hình theo thứ tự tăng dần của các chữ số.

Ví dụ: $n=291$. Xuất ra 129.

9. Nhập vào ngày, tháng, năm. Kiểm tra xem ngày, tháng, năm đó có hợp lệ hay không? In kết quả ra màn hình.
10. Nhập vào giờ, phút, giây. Kiểm tra xem giờ, phút, giây đó có hợp lệ hay không? In kết quả ra màn hình.
11. Viết chương trình nhập vào ngày, tháng, năm hợp lệ. Cho biết năm này có phải là năm nhuận hay không? In kết quả ra màn hình.
12. Viết chương trình tính diện tích và chu vi các hình: tam giác, hình vuông, hình chữ nhật và hình tròn với những thông tin cần được nhập từ bàn phím.
13. Viết chương trình tính tiền cước TAXI. Biết rằng:
 - KM đầu tiên là 5000^d.
 - 200m tiếp theo là 1000^d.
 - Nếu lớn hơn 30km thì mỗi km thêm sẽ là 3000^d.Hãy nhập số km sau đó in ra số tiền phải trả.
14. Nhập vào 3 số nguyên dương a, b, c. Kiểm tra xem 3 số đó có lập thành tam giác không? Nếu có hãy cho biết tam giác đó thuộc loại nào? (Cân, vuông, đều, ...).
15. Viết chương trình nhập vào số nguyên dương n. Kiểm tra xem n có phải là số chính phương hay không? (số chính phương là số khi lấy căn bậc 2 có kết quả là nguyên).

b. Cấu trúc lặp

16. Cho biết kết quả của đoạn chương trình sau:

```
int a=18;  
for(int i=1; i<=a; i++)  
if(a%i==0)  
printf("\t %d", i);
```

17. Cho biết kết quả của đoạn chương trình sau:

```
for(int i=0; i<5; i++)  
{  
for(int j=0; j<=i; j++)  
printf("%d\t", j);  
printf("\n");
```

}

18. Cho biết kết quả của đoạn chương trình sau:

```
int i=10, s=0;
while(i>0)
{
    if(i%2==0)
        s+=i;
    else
        if(i>5)
            s+=2*i;
    i--;
}
printf("s = %d",s);
```

19. Cho biết kết quả của đoạn chương trình sau:

```
int a=18, i=1;
do{
    if(a%i==0)
        printf("\t %d",i);
    i++;
} while(i<=a);
```

20. Cho biết kết quả của đoạn chương trình sau:

```
int a=11, b=16, i=a;
while( i<b )
{
    if(i%2==0)
    {
        printf("\t %d", i);
        break;
    }
    i++;
}
}
```

21. Cho biết kết quả của đoạn chương trình sau:

```
int a=10, s=0, i=0;
```

```

while( i<a )
{
    i++;
    if(i%2==0) continue;
    else s=s+i;
}
printf("s=%d",s);

```

22. Cho biết kết quả của đoạn chương trình sau:

```

int i=1,s=0;
while(1)
{
    s=s+i++;
    if(i%2)
        i=i+2;
    else
        i=i+1;
    if(i>20)
        break;
}
printf("%d",s);

```

23. Viết chương trình in ra màn hình hình chữ nhật đặc kích thước $m \times n$ (m, n nhập từ bàn phím).

Ví dụ: Nhập $m=5, n=4$

```

* * * * *
* * * * *
* * * * *
* * * * *

```

24. Viết chương trình in ra màn hình hình chữ nhật rỗng kích thước $m \times n$ (m, n nhập từ bàn phím).

Ví dụ: Nhập $m=5, n=4$

```

* * * * *
*           *
*           *
* * * * *

```

25. Viết chương trình in ra màn hình tam giác vuông cân đặc có độ cao h (h nhập từ bàn phím).

Ví dụ: Nhập $h=4$

```

*
* *
* * *
* * * *
    
```

26. Viết chương trình in ra màn hình tam giác cân rộng có độ cao h (h nhập từ bàn phím).

Ví dụ: Nhập $h=4$

```

*
* *
*   *
* * * *
    
```

27. Viết chương trình in ra màn hình tam giác cân đặc có độ cao h (h nhập từ bàn phím).

Ví dụ: Nhập $h=4$

```

      *
     * * *
    * * * * *
   * * * * * * *
  * * * * * * * *
    
```

28. Viết chương trình in ra màn hình tam giác cân rộng có độ cao h (h nhập từ bàn phím).

Ví dụ: Nhập $h=4$

```

      *
     *   *
    *     *
   *       *
  *         *
 *           *
*             *
    
```

29. Viết chương trình nhập số nguyên dương n . Liệt kê n số nguyên tố đầu tiên.
 30. Viết chương trình nhập vào hai số nguyên dương a và b . Tìm ước số chung lớn nhất và bội số chung nhỏ nhất của a và b .
 31. Viết chương trình nhập vào một số nguyên n gồm tối đa 10 chữ số (4 bytes). In ra màn hình giá trị nhị phân của số trên. (Hướng dẫn: chia lấy dư cho 2 và xuất theo thứ tự ngược lại dùng hàm gotoxy, wherex, wherey).
 32. Viết chương trình đếm số ước số của số nguyên dương N .

Ví dụ: $N=12$

số ước số của 12 là 6

33. Một số hoàn thiện là một số có tổng các ước số của nó (không kể nó) bằng chính nó. Hãy liệt kê các số hoàn thiện nhỏ hơn 5000.

Ví dụ: số 6 là số hoàn thiện vì tổng các ước số là $1+2+3=6$.

34. Nhập vào ngày, tháng, năm. Cho biết đó là ngày thứ mấy trong năm.
35. In ra dãy số Fibonacci
- $$f_1 = f_0 = 1 ;$$
- $$f_n = f_{n-1} + f_{n-2} ; \quad (n > 1)$$

II.3. Bài tập luyện tập và nâng cao

36. Cài đặt tất cả các lưu đồ đã vẽ ở chương 1.
37. Nhập vào ngày, tháng, năm. Kiểm tra xem ngày, tháng, năm đó có hợp lệ hay không, nếu hợp lệ cho biết ngày sau đó là bao nhiêu.
Ví dụ: *Nhập 31/12/2003*
Ngày sau đó 01/01/2004
38. Nhập vào ngày, tháng, năm. Kiểm tra xem ngày, tháng, năm đó có hợp lệ hay không, nếu hợp lệ cho biết ngày trước đó là bao nhiêu.
Ví dụ: *Nhập 01/01/2003*
Ngày trước đó 31/12/2002
39. (*) Nhập vào ngày, tháng, năm của năm 2003. Hãy kiểm tra xem dữ liệu có hợp lệ hay không? Nếu hợp lệ hãy cho biết đó là ngày thứ mấy trong tuần. (hai, ba, tư, ..., CN). (Hướng dẫn: lấy ngày 01 tháng 01 năm 2003 là ngày thứ tư làm mốc).
40. Nhập vào giờ, phút, giây. Kiểm tra xem giờ, phút, giây đó có hợp lệ hay không, nếu hợp lệ cho biết giờ sau đó 1 giây là bao nhiêu.
Ví dụ: *Nhập 01:59:59*
Giờ sau đó 1 giây 02:00:00
41. Nhập vào giờ, phút, giây. Kiểm tra xem giờ, phút, giây đó có hợp lệ hay không, nếu hợp lệ cho biết giờ trước đó 1 giây là bao nhiêu.
Ví dụ: *Nhập 02:00:00*
Giờ trước đó 1 giây 01:59:59
42. Viết chương trình in ra bảng cửu chương từ 2 đến 9.
43. (*) Vẽ hình cánh quạt sau:

```
* * * * *
* * * * *
* * * * *
* * * * *
* * * * *
* * * * *
* * * * *
* * * * *
* * * * *
* * * * *
```

Sử dụng các hàm `cprintf()`, `textcolor()`, `delay()`, `kbhit()`, ... thay đổi màu để tạo cảm giác cho cánh quạt xoay cho đến khi nhấn một phím bất kỳ.

III. KẾT LUẬN

- ❖ **Cấu trúc lặp và rẽ nhánh** (lựa chọn) là hai cấu trúc chính hình thành nên chương trình. Dựa vào những cấu trúc điều khiển này ta có thể xây dựng thành những chương trình phức tạp hơn. Vì vậy phải nắm rõ cách hoạt động của những cấu trúc điều khiển này để cài đặt đúng yêu cầu bài toán.
- ❖ Khi sử dụng phải **lưu ý điều kiện thực hiện hay kết thúc** của một thao tác nào đó.
- ❖ **Bên trong một phát biểu điều khiển phải là một lệnh hay một khối lệnh** (*khối lệnh được đặt bên trong cặp dấu ngoặc {}*).
- ❖ Những **biến không phụ thuộc vào vòng lặp nên đặt bên ngoài vòng lặp**.
- ❖ Khi sử dụng **cấu trúc điều khiển lồng nhau phải lưu ý vị trí mở ngoặc hay đóng ngoặc cho hợp lý**.

Trình bày cấu trúc của một chương trình, các bước xây dựng cài đặt chương trình theo phương pháp thủ tục hàm và một số kỹ thuật liên quan.

I. TÓM TẮT LÝ THUYẾT

I.1. Khái niệm

Hàm là một đoạn chương trình độc lập **thực hiện trọn vẹn một công việc nhất định** sau đó trả về giá trị cho chương trình gọi nó, hay nói cách khác hàm là sự chia nhỏ của chương trình.

I.2. Ví dụ

```
//Khai báo thư viện hàm
#include<conio.h>
#include<stdio.h>
#include<string.h>
#include<dos.h>
#include<process.h>

//Khai báo biến toàn cục và nguyên mẫu hàm
void ThayThe(char * S, char *St);
void DocISector(int vt);
void GhiISector(int vt);

//Hàm chính
void main()
{
    unsigned char buf[512];
    char S[20], St[20];
    printf("Nhap chuoi can tim: ");
    gets(S);
    printf("Nhap chuoi thay the:");
    gets(St);
    printf("\nXin cho...");
    TimVaThayThe(S,St,buf);
    printf("\n Thanh cong.");
    getch();
}

//Cài đặt các hàm con
void ThayThe(char * S, char *St)
{
```

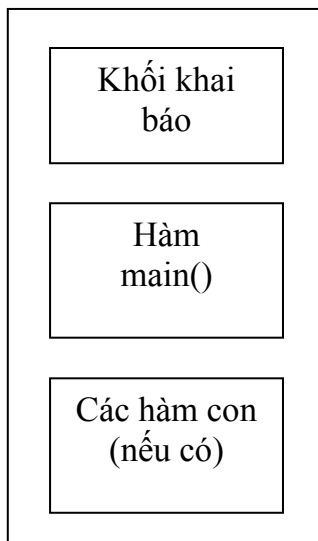
```
        int l=strlen(St);
        for(int i=0;i<l;i++)
            S[i]=St[i];
    }

    void Doc1Sector(int vt, char buf[512])
    {
        if(absread(0,1,vt,buf))
        {
            printf("\n loi doc dia, nhan enter thoat");
            getch();
            exit(1);
        }
    }

    void Ghi1Sector(int vt, char buf[512])
    {
        if(abswrite(0,1,vt,buf))
        {
            printf("\n loi ghi dia, nhan enter thoat");
            getch();
            exit(1);
        }
    }

    void TimVaThayThe(char * S, char *St, unsigned char buf[])
    {
        for(int i=33;i<=500;i++)
        {
            Doc1Sector(i, buf);
            char * p=strstr(buf, S);
            if(p)
            {
                ThayThe(p, St);
                Ghi1Sector(i, buf);
            }
        }
    }
}
```


I.3. Cấu trúc một chương trình C



a. *Khôi khai báo*

Bao gồm các khai báo về sử dụng thư viện, khai báo hằng số, khai báo hàm con (các nguyên mẫu hàm), khai báo các biến toàn cục và khai báo các kiểu dữ liệu tự định nghĩa.

b. *Hàm chính (main())*

Chứa các biến, các lệnh và các lời gọi hàm cần thiết trong chương trình.

c. *Các hàm con*

Được sử dụng nhằm mục đích:

- Khi có một công việc giống nhau cần thực hiện ở nhiều vị trí.
- Khi cần chia một chương trình lớn phức tạp thành các đơn thể nhỏ (hàm con) để chương trình được trong sáng, dễ hiểu trong việc xử lý, quản lý việc tính toán và giải quyết vấn đề.

d. *Nguyên mẫu hàm*

<Kiểu dữ liệu của hàm> Tên hàm ([danh sách các tham số]);

Nguyên mẫu hàm thực chất là dòng đầu của hàm thêm dấu chấm phẩy (;) vào cuối, tuy nhiên tham số trong nguyên mẫu hàm có thể bỏ phần tên.

I.4. Cách xây dựng một hàm con

a. Kiểu dữ liệu của hàm

Xác định dựa vào kết quả của bài toán (Output). Gồm 2 loại :

- **void:** Hàm không trả về giá trị. Những hàm loại này thường rơi vào những nhóm chức năng: **Nhập / xuất dữ liệu , thống kê, sắp xếp, liệt kê.**

```
void Tên_hàm (danh sách các tham số)
{
    Khai báo các biến cục bộ
    Các câu lệnh / khối lệnh hay lời gọi đến hàm khác.
}
```

- **Kiểu dữ liệu cơ bản (rời rạc/ liên tục) hay kiểu dữ liệu có cấu trúc:** Kiểu dữ liệu tùy theo mục đích của hàm cần trả về giá trị gì thông qua việc phân tích bài toán. Những hàm loại này thường được sử dụng trong các trường hợp: **Đếm, kiểm tra, tìm kiếm, tính trung bình, tổng, tích, ...**

```
<Kiểu dữ liệu> Tên_hàm ([danh sách các tham số])
{
    <Kiểu dữ liệu> kq;
    Khai báo các biến cục bộ
    Các câu lệnh / khối lệnh hay lời gọi đến hàm khác.

    return kq;
}
```

☞ Đối với những hàm trả về nhiều loại giá trị cho từng trường hợp cụ thể (chẳng hạn như kiểm tra: đúng hay sai, so sánh: bằng , lớn hơn hay nhỏ hơn, ...) thì cần ghi chú rõ giá trị trả về là gì cho từng trường hợp đó.

b. Tham số

Xác định dựa vào dữ liệu đầu vào của bài toán (Input). Gồm 2 loại :

- **Tham số không là con trỏ (tham trị):** Không thay đổi hoặc không cần lấy giá trị mới của tham số sau lời gọi hàm. Tham số dạng này chỉ mang ý nghĩa là **dữ liệu đầu vào.**
- **Tham số con trỏ (tham biến):** Có sự thay đổi giá trị của tham số trong quá trình thực hiện và cần lấy lại giá trị đó sau khi ra khỏi hàm. Ứng dụng của tham số loại này có thể là **dữ liệu đầu ra** (kết quả) hoặc cũng có thể vừa là **dữ liệu đầu vào** vừa là **dữ liệu đầu ra.**

c. *Tên hàm*

Đặt tên theo **quy ước đặt tên trong C** sao cho tên gọi **đúng với chức năng hay mục đích thực hiện của hàm và gọi nhớ.**

d. *Ví dụ*

Ví dụ 1: Viết chương trình nhập số nguyên dương n và in ra màn hình các ước số của n

Phân tích bài toán:

- **Input:** n (Để xác định tham số)
 - Kiểu dữ liệu: số nguyên dương (*unsigned int*).
 - Giá trị n không bị thay đổi trong quá trình tìm ước số \rightarrow Tham số của hàm *không là con trỏ*.
- **Output:** In ra các ước số của n (Để xác định kiểu dữ liệu hàm)
 - Không trả về giá trị.
 - Kiểu dữ liệu của hàm là *void*.
- **Xác định tên hàm:** Hàm này dùng in ra các ước số của n nên có thể đặt là *LietKeUocSo*

Ta có nguyên mẫu hàm:

```
void LietKeUocSo ( unsigned int n );
```

```
#include<conio.h>
#include<stdio.h>

//Khai bao nguyen mau ham
void LietKeUocSo ( unsigned int n );

void main()
{
        unsigned int n;

        printf("Nhap n = ");
        scanf("%u",&n);
        printf("Cac uoc so cua n : ");
        LietKeUocSo(n);
        getch();
}

void LietKeUocSo ( unsigned int n)
{
```

```

    for(int i=1; i<=n; i++)
        printf("%u\t", i);
}

```

✎ **Lưu ý cách gọi hàm:** Đối với hàm có kiểu dữ liệu hàm là **void** thì khi gọi không cần phải gán giá trị vào biến, ngược lại phải gọi như trong ví dụ 2 (Phải khai báo tương ứng kiểu với kiểu dữ liệu hàm sẽ gọi và gán giá trị trả về vào biến đó).

Ví dụ 2: Viết chương trình nhập số nguyên dương n và tính tổng

$$S = 1 + 2 + 3 + \dots + n, \text{ với } n > 0$$

Phân tích bài toán:

- **Input:** n (Để xác định tham số)
 - Kiểu dữ liệu: số nguyên dương (*unsigned int*).
 - Giá trị n không bị thay đổi trong quá trình tính tổng → Tham số của hàm *không là con trỏ*.
- **Output:** Tổng S (Để xác định kiểu dữ liệu hàm)
 - Trả về giá trị của S.
 - S là tổng các số nguyên dương nên S cũng là số nguyên dương → Kiểu trả về của hàm là *unsigned int* (hoặc *unsigned long* cho trường hợp giá trị của tổng lớn hơn 2 bytes).
- **Xác định tên hàm:** Hàm này dùng tính tổng S nên có thể đặt là *TongS*.

Ta có nguyên mẫu hàm:

```

    unsigned long TongS ( unsigned int n );

```

```

#include<conio.h>
#include<stdio.h>

//Khai bao nguyen mau ham
unsigned long TongS ( unsigned int n );

void main()
{
    unsigned int n;
    unsigned long kq;

    printf("Nhap n = ");

```

```

scanf("%u",&n);
kq = TongS ( n );
printf("Tong can tinh la: %lu ", kq);
getch();
}
unsigned long TongS (unsigned int n)
{
    unsigned long S=0;
    int i=1;
    while(i<=n)
    {
        S+=i;
        i++;
    }
    return S;
}

```

II. BÀI TẬP

II.1. Bài tập cơ bản

1. Cài đặt lại tất cả các bài tập ở chương 2 theo phương pháp hàm.
2. Viết chương trình tính diện tích và chu vi của hình chữ nhật với chiều dài và chiều rộng được nhập từ bàn phím.
3. Viết chương trình tính diện tích và chu vi hình tròn với bán kính được nhập từ bàn phím.
4. Nhập số nguyên dương n ($n > 0$). Liệt kê tất cả các số nguyên tố nhỏ hơn n .
5. Nhập số nguyên dương n ($n > 0$). Liệt kê n số chính phương đầu tiên.
6. Nhập số nguyên dương n ($n > 0$). Đếm xem có bao nhiêu số hoàn thiện nhỏ hơn n .
7. Nhập số nguyên dương n ($0 \leq n < 1000$) và in ra cách đọc của n .
Ví dụ: Nhập $n = 105$. In ra màn hình: *Mot tram le nam.*
8. Viết chương trình tính tiền thuê máy dịch vụ Internet và in ra màn hình kết quả. Với dữ liệu nhập vào là giờ bắt đầu thuê (GBD), giờ kết thúc thuê (GKT), số máy thuê (SoMay).
 - Điều kiện cho dữ liệu nhập: $6 \leq \text{GBD} < \text{GKT} \leq 21$. Giờ là số nguyên.
 - Đơn giá: 2500đ cho mỗi giờ máy trước 17:30 và 3000đ cho mỗi giờ máy sau 17:30.
9. Viết chương trình tính tiền lương ngày cho công nhân, cho biết trước giờ vào ca, giờ ra ca của mỗi người.

Giả sử rằng:

- Tiền trả cho mỗi giờ trước 12 giờ là 6000đ và sau 12 giờ là 7500đ.
- Giờ vào ca sớm nhất là 6 giờ sáng và giờ ra ca trễ nhất là 18 giờ (*Giả sử giờ nhập vào nguyên*).

10. Nhập vào 2 số nguyên p, q và tính biểu thức sau:

$$(-q/2 + (p^3/27 + q^2/4)^{1/2})^{1/3} + (-q/2 - (p^3/27 + q^2/4)^{1/2})^{1/3}$$

11. Nhập vào 3 số thực a, b, c và kiểm tra xem chúng có thành lập thành 3 cạnh của một tam giác hay không? Nếu có hãy tính diện tích, chiều dài mỗi đường cao của tam giác và in kết quả ra màn hình.

- Công thức tính diện tích $s = \sqrt{p(p-a)(p-b)(p-c)}$
 - Công thức tính các đường cao: $h_a = 2s/a, h_b = 2s/b, h_c = 2s/c$.
- (Với p là nửa chu vi của tam giác).

12. Nhập vào 6 số thực a, b, c, d, e, f . Giải hệ phương trình sau :

$$\begin{cases} ax+by=c \\ dx+ey=f \end{cases}$$

13. Viết chương trình nhập 2 số nguyên dương a, b . Tìm USCLN và BSCNN của hai số nguyên đó.

14. Viết chương trình tính tổng nghịch đảo của n giai thừa.

15. Cho 2 số nguyên a, b . Viết hàm hoán vị giá trị 2 số trên.

16. (*) Viết chương trình nhập số nguyên dương n gồm 5 chữ số, kiểm tra xem các chữ số n có phải là số đối xứng hay không.

Ví dụ: *Đối xứng:* 13531

Không đối xứng: 13921

17. Viết chương trình nhập số nguyên dương n gồm k chữ số ($0 < k \leq 5$), đếm xem n có bao nhiêu chữ số chẵn và bao nhiêu chữ số lẻ.

18. Viết chương trình nhập số nguyên dương n gồm k chữ số ($0 < k \leq 5$), đếm xem n có bao nhiêu chữ số là số nguyên tố.

19. Viết chương trình nhập số nguyên dương n gồm k chữ số ($0 < k \leq 5$), tính tổng các ước số dương của n .

Ví dụ: *Nhập $n=6$*

Tổng các ước số từ 1 đến n : $1+2+3+6=12$.

20. Viết chương trình nhập số nguyên dương n gồm k chữ số ($0 < k \leq 5$), tìm ước số lẻ lớn nhất của n .

Ví dụ: Ước số lẻ lớn nhất của 27 là 9.

21. Viết chương trình nhập số nguyên dương n gồm k chữ số ($0 < k \leq 5$), kiểm tra xem các chữ số của n có toàn lẻ hay toàn chẵn không.
22. (*) Viết chương trình nhập số nguyên dương n gồm k chữ số ($0 < k \leq 5$), sắp xếp các chữ số của n theo thứ tự tăng dần.

Ví dụ: Nhập $n=1536$

Kết quả sau khi sắp xếp: 1356.

II.2. Bài tập luyện tập và nâng cao

23. Viết chương trình nhập số nguyên dương n gồm k chữ số ($0 < k \leq 5$), sau đó nhập một số nguyên x , tìm vị trí xuất hiện của chữ số có giá trị x trong n .

Ví dụ: Nhập $n=1526$, $x=2$

Kết quả: Chu số 2 ở vị trí thứ 3.

24. Viết chương trình nhập số nguyên dương n gồm k chữ số ($0 < k \leq 5$), kiểm tra xem các chữ số của n có được sắp thứ tự không.

Ví dụ: Nhập $n=1569$ hoặc $n=8521$

Kết quả: Có thứ tự.

25. Viết chương trình nhập 2 số a, b sao cho: số lớn nhất trong 2 số phải là một số dương và chia hết cho 7. Nếu nhập sai phải yêu cầu nhập lại cho đến khi đúng.
26. Viết chương trình nhập số nguyên dương n gồm k chữ số ($0 < k \leq 5$), tính giá trị trung bình các chữ số chẵn trong n .
27. (*) Viết chương trình in ra màn hình ngày/tháng/năm của ngày hiện tại, cho phép sử dụng các phím mũi tên lên, xuống để tăng hoặc giảm một ngày.
28. (*) Viết chương trình in ra màn hình giờ:phút:giây hiện tại, cho phép sử dụng các phím mũi tên lên, xuống để tăng hoặc giảm một giây.

III. KẾT LUẬN

- ❖ Trước khi xây dựng một hàm ta phải **xác định mục đích của hàm** là dùng để làm gì, trên cơ sở đó, ta mới xác định được các thành phần của hàm và xây dựng nguyên mẫu hàm.
- ❖ Mỗi hàm phải **thực hiện một chức năng độc lập** và tách biệt với các hàm khác (*không được lồng nhau*).

- ❖ Đối với hàm có giá trị trả về phải lưu ý kiểu dữ liệu phải tương ứng kiểu dữ liệu cả giá trị trả về và kiểu dữ liệu của biến được gán khi gọi hàm. Trường hợp hàm trả về từ **hai loại giá trị trở lên thì phải có dòng chú thích cho trường hợp tương ứng** để khi gọi hàm biết được kết quả (*chẳng hạn như tìm kiếm, kiểm tra, so sánh, ... giá trị trả về có 2 trường hợp: Có hoặc không có phần tử cần tìm, thỏa điều kiện kiểm tra hay không? Do vậy ta phải quy ước giá trị cho từng trường hợp*).
- ❖ Nên đặt tên hàm sao cho **gọi nhớ** được chức năng, đặt tên **theo quy tắc nhất định** để **tránh việc gọi sai tên hàm** do lẫn lộn giữa ký tự hoa và thường, có dấu gạch nối giữa các từ trong hàm hay không?
- ❖ Khi gọi hàm phải truyền **đủ tham số, đúng kiểu dữ liệu** và **đúng thứ tự** của tham số.

CHƯƠNG 4 MẢNG MỘT CHIỀU

Cách khai báo dữ liệu kiểu mảng, các thao tác nhập xuất, các kỹ thuật thao tác trên mảng. Ứng dụng các kỹ thuật này trong việc cài đặt các hàm tìm kiếm, kiểm tra, xây dựng mảng, tách và ghép mảng.

I. TÓM TẮT LÝ THUYẾT

I.1. Khái niệm

Mảng thực chất là một biến được cấp phát bộ nhớ liên tục và bao gồm nhiều biến thành phần.

Các thành phần của mảng là tập hợp các biến có cùng kiểu dữ liệu và cùng tên. Do đó để truy xuất các biến thành phần, ta dùng cơ chế chỉ mục.

I.2. Khai báo mảng

Để khai báo một mảng, ta có 2 cách khai báo sau :

❖ Cách 1: Con trỏ hằng

< Kiểu dữ liệu > < Tên mảng > [< Số phần tử tối đa của mảng >] ;

Ví dụ:

int a[100]; // Khai báo mảng số nguyên a gồm 100 phần tử

float b[50]; // Khai báo mảng số thực b gồm 50 phần tử

❖ Cách 2: Con trỏ

Ý nghĩa: Khi ta khai báo một mảng với kiểu dữ liệu bất kì (int, float, char,...) thì tên của mảng thực chất là một **hằng địa chỉ của phần tử đầu tiên**.

< Kiểu dữ liệu > * < Tên mảng >;

Ví dụ :

*int *p; // khai báo con trỏ p*

int b[100];

p = b; // p trỏ vào phần tử 0 của mảng b

Với cách viết như trên thì ta có thể hiểu các cách viết sau là tương đương

p[i] ⇔ *(p + i) ⇔ b[i] ⇔ *(b+i)

➤ **Lưu ý:** Khi sử dụng biến con trỏ để truy xuất mảng, theo cách như trên thì thực chất con trỏ *p* chỉ chiếm 2 byte bộ nhớ để chứa địa chỉ mà thôi. Để tạo mảng chứa dữ liệu thành phần thì ta phải cấp phát vùng nhớ cho con trỏ *p*.
 Dùng hàm : **malloc, calloc** trong thư viện **<stdlib.h>** để cấp phát vùng nhớ.

Ví dụ:

+ Cách 1: dùng malloc

```
int *px; //Khai báo con trỏ px
px = (int *) malloc (100); //Cấp phát 100 ô nhớ kiểu int cho con trỏ px
```

+ Cách 2: dùng calloc

```
int *p; //khai báo con trỏ p
p=(int *) calloc (100,sizeof (int)); //cấp phát 10 ô nhớ mỗi ô chiếm 2bytes
```

Sau khi sử dụng xong thì nên giải phóng vùng nhớ bằng hàm free

Ví dụ : free (p) ; // giải phóng vùng nhớ cho con trỏ p.

I.3. Truy xuất phần tử của mảng

Với khái niệm và cách khai báo như trên ta có hình dạng của mảng một chiều như sau:

Ví dụ : int A[5] // Khai báo mảng A gồm tối đa 5 phần tử nguyên.

Chỉ số 0 1 2 3 4

| | | | | |
|------|------|------|------|------|
| A[0] | A[1] | A[2] | A[3] | A[4] |
|------|------|------|------|------|

Ví dụ minh hoạ:

Khai báo và gán giá trị cho mảng

```
#include <conio.h>
#include <stdio.h>

void main ()
{
    clrscr ();
    int a[4] = {5,9,3,8};
    for (int i = 0; i < 4 ; i++)
        printf (" a [ %d ] = %d \t", i , a[i] );
    getch ();
}
```

Đối với con trỏ: Lấy địa chỉ của phần tử trong mảng ta dùng dấu “&”

Ví dụ:

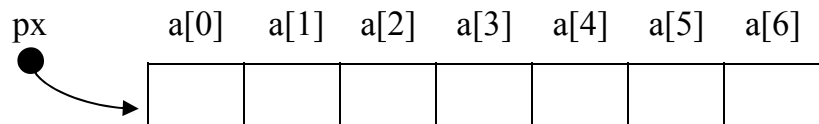
```
int a[7];
```

`int *p = a[3];` //Lấy địa chỉ phần tử thứ 3

Ví dụ :

```
int a[7];
int *px;
px = a; //px trở tới phần tử thứ 0
px = px + 4; //px trở tới phần tử thứ 4
```

Từ ví dụ trên ta có thể mô hình hoá mảng như sau:



Ví dụ minh họa: Viết chương trình nhập vào mảng một chiều 10 phần tử kiểu số nguyên

```
#include <conio.h>
#include <stdio.h>

void main ()
{
    int a[10], i;
    int *p;

    for (i = 0 ; i < 10 ; i++)
    {
        printf (" a [ %d ] = ", i);
        scanf (" %d", &a[i] );
    }
    p = a;
    printf ("\n Noi dung mang vua nhap: ");
    for (i = 0; i < 10 ; i++)
        printf (" %d \t ", *(p + i));
    getch ();
}
```

II. BÀI TẬP

II.1. Một số kĩ thuật cơ bản

a. Kĩ thuật đặt cờ hiệu

Kĩ thuật này thường được áp dụng cho những bài toán “kiểm tra” hay “đánh dấu”.

Viết hàm kiểm tra xem mảng các số nguyên có thứ tự tăng dần không?
(Trả về 1: Nếu mảng tăng dần, ngược lại trả về 0).

```

int KiemTraTang (int a[ ], int n)
{
    int flag = 1;
    for (int i = 0; i < n-1; i++)
        if ( a[i] > a[i+1] ) //Vi phạm điều kiện tăng dần
        {
            flag = 0;
            break;
        }
    return flag;
}

```

Viết hàm kiểm tra xem trong mảng các số nguyên có tồn tại số nguyên lẻ lớn hơn 100 hay không?

(Trả về 1: Nếu có tồn tại số lẻ và lớn hơn 100, ngược lại trả về 0).

```

int KiemTraLe (int a[ ], int n)
{
    int flag = 0;
    for (int i = 0; i < n; i++)
        if ( a[i] % 2 != 0 && a[i]>100 ) //Gặp phần tử thoả
        {
            flag = 1;
            break;
        }
    return flag;
}

```

b. Kỹ thuật đặt lính canh

Kỹ thuật này thường được áp dụng cho những bài tập về “tìm kiếm”, “liệt kê” theo một điều kiện nhất định nào đó.

Viết hàm tìm và trả về giá trị lớn nhất trong mảng một chiều các số nguyên.

```

int TimMax (int a[], int n)
{
    int max, i = 1;

    max = a[0];
    while ( i < n )
    {
        if ( a[i] > max )
            max = a[i] ;
        i++;
    }
    return max;
}

```

II.2. Bài tập cơ bản

a. Nhập xuất mảng một chiều

Phương pháp cơ bản

Viết chương trình nhập xuất mảng một chiều các số nguyên.

```
#include <conio.h>
#include <stdio.h>
#define MAX 100

void NhapMang (int a[], int &n)
{
    printf ("Nhap so phan tu: ");
    scanf ("%d", &n);
    for (int i = 0; i < n; i++)
    {
        printf (" a [%d] = ", i);
        scanf ("%d", &a[i]);
    }
}

void XuatMang (int a[], int n)
{
    printf ("\nNoi dung mang: ");
    for (int i = 0; i < n; i++)
        printf (" %d\t", a[i]);
}

void main ()
{
    clrscr ();
    int a[MAX], n;
    NhapMang (a,n);
    XuatMang (a,n);
    getch ();
}
```

Bài tập

1. Viết chương trình nhập xuất mảng một chiều các số thực.
2. Viết chương trình khởi tạo giá trị các phần tử là 0 cho mảng một chiều các số nguyên gồm n phần tử.
3. Viết chương trình phát sinh ngẫu nhiên mảng một chiều các số nguyên âm.
4. Viết chương trình phát sinh ngẫu nhiên mảng một chiều các số nguyên sao cho mảng có thứ tự tăng dần (Không sắp xếp).

- Viết chương trình nhập mảng các số thực và xuất các phần tử âm trong mảng.
- Viết chương trình nhập mảng các số nguyên và xuất các phần tử lẻ có trong mảng.
- Viết chương trình nhập vào mảng một chiều các số nguyên và xuất ra các phần tử chẵn nhỏ hơn 20.
- Viết chương trình nhập vào mảng một chiều các số nguyên và xuất ra màn hình các phần tử là số nguyên tố.
- Viết chương trình nhập vào số nguyên n và liệt kê các số nguyên tố nhỏ hơn n, nếu mảng không tồn tại số nguyên tố nào nhỏ hơn n thì phải xuất ra một câu thông báo.
- Viết chương trình nhập vào mảng một chiều các số nguyên và xuất ra màn hình các phần tử là số chính phương nằm tại những vị trí lẻ trong mảng.

b. Tìm kiếm trên mảng một chiều

Phương pháp cơ bản

Viết hàm tìm phần tử có giá trị x xuất hiện đầu tiên trong mảng một chiều.

(Nếu tìm thấy trả về vị trí xuất hiện x, ngược lại trả về -1)

```
int TimX (int a[], int n, int x)
{
    for (int i = 0; i < n ; i++)
        if (x==a[i])
            return i;
    return -1;
}
```

Bài tập

- Viết hàm tìm vị trí phần tử có giá trị x xuất hiện cuối cùng trong mảng.
- Viết hàm tìm vị trí của phần tử nhỏ nhất trong mảng các số nguyên.
- Viết hàm tìm vị trí của phần tử lớn nhất trong mảng các số nguyên.
- Viết hàm in vị trí các phần tử nguyên tố trong mảng các số nguyên.
- Viết hàm in vị trí các phần tử nguyên tố lớn hơn 23.
- Viết hàm tìm vị trí phần tử âm đầu tiên trong mảng. Nếu không có phần tử âm trả về -1.
- Viết hàm tìm vị trí phần tử âm lớn nhất trong mảng.

18. Viết hàm tìm vị trí phần tử dương đầu tiên trong mảng. Nếu không có phần tử âm trả về -1.
19. Viết hàm tìm vị trí phần tử dương bé nhất trong mảng.
20. Viết hàm in các phần tử là bội của 3 và 5.
21. Viết hàm tìm số chẵn cuối cùng có trong mảng, nếu không tồn tại số chẵn hàm trả về -1 .
22. Viết hàm tìm số lẻ lớn nhất có trong mảng, nếu không tồn tại số lẻ hàm trả về -1.
23. Viết hàm tìm và đổi chỗ phần tử lớn nhất với phần tử nhỏ nhất trong mảng.
24. Nhập vào X. Viết hàm in ra màn hình những phần tử có giá trị từ 1 đến X có trong mảng.
25. Viết chương trình nhập vào một dãy số a gồm n số thực ($n \leq 100$), nhập vào dãy số b gồm m số thực ($m \leq 100$).
 - In ra những phần tử chỉ xuất hiện trong dãy a mà không xuất hiện trong dãy b.
 - In ra những phần tử xuất hiện ở cả hai dãy.

c. Đếm – Tần suất

Phương pháp cơ bản

Viết hàm đếm các phần tử chia hết cho 5 trong mảng các số nguyên.

```
int Dem (int a[], int n )
{
    int dem = 0;
    for (int i = 0; i < n ; i++)
        if ( a[i] % 5 == 0 )
            dem++;
    return dem;
}
```

Bài tập

26. Viết hàm đếm các phần tử âm, dương trong mảng.
27. Viết hàm đếm các phần tử chẵn, lẻ trong mảng.
28. Viết hàm đếm số lần xuất hiện của phần tử x trong mảng.
29. Viết hàm đếm các phần tử nhỏ hơn x trong mảng.
30. Viết hàm đếm các phần tử là số nguyên tố trong mảng.

31. Viết hàm đếm các phần tử là số hoàn thiện trong mảng.
32. Viết hàm đếm các phần tử là bội của 3 và 5 trong mảng các số nguyên.

d. Tính tổng – Trung bình có điều kiện

Phương pháp cơ bản

Viết hàm tính tổng các phần tử trong mảng.

```
long TinhTong (int a[], int n)
{
    long tong = 0;
    for (int i = 0; i < n; i++)
        tong = tong + a[i];
    return tong;
}
```

Viết hàm tính giá trị trung bình các phần tử có giá trị âm trong mảng.

Đối với hàm tính trung bình có điều kiện phải lưu ý khi chia giá trị (Có thể mảng không có phần tử nào thoả điều kiện, nếu ta chia tức là chia cho 0).

```
float TrungBinhAm (int a[], int n)
{
    long tong = 0;
    int spt=0;
    for (int i = 0; i < n; i++)
        if( a[i]<0 )
        {
            tong = tong + a[i];
            spt++;
        }
    if(spt==0)
        return 0;
    return 1.0*tong/spt;
}
```

Bài tập

33. Viết hàm tính tổng các phần tử chẵn trong mảng.
34. Viết hàm tính tổng các phần tử lẻ trong mảng các số nguyên.
35. Viết hàm tính tổng các phần tử nguyên tố trong mảng.
36. Viết hàm tính tổng các phần tử nằm ở vị trí chẵn trong mảng các số nguyên.
37. Viết hàm tính tổng các phần tử nằm ở vị trí nguyên tố trong mảng.
38. Viết hàm tính tổng các phần tử chia hết cho 5 có trong mảng.
39. Viết hàm tính tổng các phần tử cực đại trong mảng các số nguyên (*phần tử cực đại là phần tử lớn hơn các phần tử xung quanh nó*).

Ví dụ : 1 5 2 6 3 5 1 8 6

40. Viết hàm tính tổng các phần tử cực tiểu trong mảng các số nguyên (*phần tử cực tiểu là phần tử nhỏ hơn các phần tử xung quanh nó*).

Ví dụ : 6 4 2 9 5 3 7 1 5 8

41. Viết hàm tính tổng các phần tử là bội của 3 và 5 trong mảng các số nguyên.
42. Viết hàm tính tổng các phần tử là số hoàn thiện trong mảng các số nguyên.
43. Viết hàm tính giá trị trung bình của các số hoàn thiện trong mảng các số nguyên.

e. Sắp xếp

Kĩ thuật cơ bản

Viết hàm sắp xếp mảng theo thứ tự tăng dần.

```
void HoanVi (int &a, int &b)
{
    int tam = a;
    a = b;
    b = tam;
}

void SapTang (int a[], int n)
{
    for (int i = 0; i < n-1; i++)
        for (int j = i+1; j < n; j++)
            if (a[i] > a [j])
                HoanVi (a[i], a[j]);
}
```

Bài tập

44. Viết hàm sắp xếp mảng theo thứ tự giảm dần.
45. Viết hàm sắp xếp mảng theo thứ tự tăng dần của các phần tử là số nguyên tố.
46. Viết hàm sắp xếp các phần tử lẻ tăng dần.
47. Viết hàm sắp xếp các phần tử chẵn giảm dần.
48. Viết hàm sắp xếp các phần tử chẵn nằm bên trái theo thứ tự tăng dần còn các phần tử lẻ bên phải theo thứ tự giảm dần.
49. Viết hàm sắp xếp các phần tử âm giảm dần từ trái sang phải, phần tử dương tăng dần từ phải sang trái.

f. Xoá

Kĩ thuật cơ bản

Duyệt mảng từ trái sang phải. Xuất phát từ vị trí cần xoá tiến hành dời lần lượt các phần tử về phía trước cho đến khi kết thúc mảng, sau đó giảm kích thước mảng.

Vấn đề đặt ra là tìm vị trí cần xoá theo điều kiện bài toán rồi thực hiện xoá.

Viết hàm xoá phần tử đầu tiên của mảng.

```
void XoaDau (int a[], int &n)
{
    for (int i = 0; i < n-1 ; i++)
        a[i] = a[i+1];
    n--;
}
```

Viết hàm xoá phần tử tại vị trí (vitri) cho trước trong mảng.

```
void XoaTaiViTri (int a[], int &n, int vitri)
{
    for (int i = vitri; i < n-1 ; i++)
        a[i] = a[i+1];
    n--;
}
```

Bài tập

50. Viết hàm xoá phần tử tại vị trí lẻ trong mảng.
51. Viết hàm xoá phần tử có giá trị lớn nhất trong mảng.
52. Nhập vào giá trị X. Viết hàm xoá tất cả các phần tử có giá trị nhỏ hơn X.
53. Nhập vào giá trị X. Viết hàm xoá phần tử có giá trị gần X nhất.

g. Chèn

Kĩ thuật cơ bản

Duyệt mảng từ phải sang trái. Xuất phát từ cuối mảng tiến hành đẩy lần lượt các phần tử về phía sau cho đến vị trí cần chèn, chèn phần tử cần chèn vào vị trí chèn và tăng kích thước mảng.

Trước khi chèn ta phải xác định vị trí cần chèn theo điều kiện bài toán.

Thêm phần tử có giá trị X vào cuối mảng.

```
void ThemCuoi (int a[], int &n, int X)
{
    a[n]=X;
    n++;
}
```

}

Chèn phần tử có giá trị X vào mảng tại vị trí cho trước

```

void ChenX (int a[], int &n, int X, int vitri)
{
    for (int i = n; i > vitri ; i--)
        a[i] = a[i-1] ;
    a[vitri] = X;
    n++;
}

```

Bài tập

54. Viết hàm chèn phần tử có giá trị X vào vị trí đầu tiên của mảng.
55. Viết hàm chèn phần tử có giá trị X vào phía sau phần tử có giá trị lớn nhất trong mảng.
56. Viết hàm chèn phần tử có giá trị X vào trước phần tử có giá trị là số nguyên tố đầu tiên trong mảng.
57. Viết hàm chèn phần tử có giá trị X vào phía sau tất cả các phần tử có giá trị chẵn trong mảng.

h. Tách / ghép mảng**Kĩ thuật tách cơ bản**

Cho mảng a kích thước n (n chẵn). Tách mảng a thành 2 mảng b và c sao cho: b có ½ phần tử đầu của mảng a, ½ phần tử còn lại đưa vào mảng c.

```

void TachMang(int a[], int n, int b[], int &m, int c[], int &l)
{
    int k=n/2;

    m=l=0;
    for(int i=0; i<k; i++)
    {
        b[m++]=a[i];
        c[l++]=a[k+i]
    }
}

```

Kĩ thuật ghép cơ bản

Cho 2 mảng số nguyên a và b kích thước lần lượt là n và m. Viết chương trình nối mảng b vào cuối mảng a.

```
void NoiMang(int a[], int &n, int b[], int m)
{
    for(int i=0; i<m; i++)
        a[n+i]=b[i];
    n=n+m;
}
```

Cho 2 mảng số nguyên a và b kích thước lần lượt là n và m. Viết chương trình nối xen kẽ (*đan xen*) lần lượt các phần tử mảng a và b vào mảng c.

Cách thực hiện: Đưa lần lượt từng phần tử của mảng a và mảng b vào mảng c, tăng chỉ số tương ứng. Nếu một trong hai mảng hết trước thì chép tất cả các phần tử còn lại của mảng chưa hết vào mảng c.

Đặt *i* là chỉ số của mảng a; *j*: chỉ số của mảng b và *k* là chỉ số của mảng c.

```
void NoiMang(int a[], int &n, int b[], int m, int c[], int &k)
{
    int i=0, j=0;
    k=0;
    while(i<n && j<m)
    {
        c[k++]=a[i++];
        c[k++]=b[j++];
    }
    while(i<n)
        c[k++]=a[i++];
    while(j<m)
        c[k++]=b[j++];
}
```

Bài tập

58. Viết chương trình tách 1 mảng các số nguyên thành 2 mảng a và b, sao cho mảng a chứa toàn số lẻ và mảng b chứa toàn số chẵn.

Ví dụ: Mảng ban đầu: 1 3 8 2 7 5 9 0 10

Mảng a: 1 3 7 5 9

Mảng b: 8 2 10

59. Cho 2 mảng số nguyên a và b kích thước lần lượt là n và m. Viết chương trình nối 2 mảng trên thành mảng c theo nguyên tắc chẵn ở đầu mảng và lẻ ở cuối mảng.

Ví dụ: Mảng a: 3 2 7 5 9

Mảng b: 1 8 10 4 12 6

Mảng c: 6 12 4 10 2 8 3 1 7 5 9

II.3. Bài tập luyện tập và nâng cao

60. Viết chương trình nhập vào mảng A gồm n phần tử, trong quá trình nhập kiểm tra các phần tử nhập vào không được trùng, nếu trùng thông báo và yêu cầu nhập lại.

61. Viết hàm tính tổng của từng dãy con giảm có trong mảng.

62. (*) Cho mảng các số nguyên a gồm n phần tử ($n \leq 30000$) và số dương k ($k \leq n$). Hãy chỉ ra số hạng lớn thứ k của mảng.

Ví dụ: Mảng a: 6 3 1 10 11 18

$$k = 2$$

Kết quả: 10

63. (*) Cho 2 dãy A, B các số nguyên (kích thước dãy A nhỏ hơn dãy B). Hãy kiểm tra xem A có phải là con của B hay không?

64. Viết hàm liệt kê các bộ 4 số a, b, c, d trong mảng các số nguyên (có ít nhất 4 phần tử và đôi một khác nhau) sao cho $a + b = c + d$.

65. (*) Viết chương trình tính trung bình cộng của các tổng các dãy tăng dần có trong mảng các số nguyên.

Ví dụ: 1 2 3 4 2 3 4 5 6 4 5 6 $\Rightarrow TB = 15$.

66. Viết chương trình tính tổng tất cả các phần tử xung quanh trên mảng các số nguyên. (Phần tử xung quanh là hai phần tử bên cạnh cộng lại bằng chính nó (Ví dụ: 1 3 2 \rightarrow 1,2 là hai phần tử xung quanh của 3).

Ví dụ: 1 3 2 5 3 9 6 \rightarrow tổng 17

67. (**) Viết chương trình nhập vào hai số lớn a, b nguyên (a, b có từ 20 chữ số trở lên). Tính tổng, hiệu, tích, thương của hai số trên.

68. Viết hàm tính tổng các phần tử là số Armstrong (số Armstrong là số có đặc điểm như sau: số có k ký số, tổng của các lũy thừa bậc k của các ký số bằng chính số đó.

Ví dụ: 153 là số có các ký số $1^3 + 5^3 + 3^3 = 153$ là một số Armstrong).

69. Viết hàm tìm và xóa tất cả các phần tử trùng với x trong mảng một chiều các số nguyên, nếu không tồn tại phần tử x trong mảng thì trả về -1.

70. Viết hàm xoá tất cả những phần tử trùng nhau trong dãy chỉ giữ lại một phần tử trong đó.

Ví dụ: 1 6 2 3 2 4 2 6 5 → 1 6 2 3 4 5

71. (**) Viết hàm xoá những phần tử sao cho mảng kết quả có thứ tự tăng dần và số lần xoá là ít nhất.

72. Cho dãy a gồm n số nguyên có thứ tự tăng dần. Nhập vào một phần tử nguyên X, viết hàm chèn X vào dãy sao cho dãy vẫn có thứ tự tăng dần (*không sắp xếp*).

73. Viết chương trình tìm số lẻ nhỏ nhất lớn hơn mọi số chẵn có trong mảng.

74. Viết hàm tìm giá trị chẵn nhỏ nhất nhỏ hơn mọi giá trị lẻ trong mảng các số nguyên.

75. Viết hàm tìm phần tử xuất hiện nhiều nhất trong mảng các số nguyên.

76. Viết chương trình đếm và liệt kê các mảng con tăng dần trong mảng một chiều các số nguyên.

Ví dụ: 6 5 3 2 3 4 2 7 các dãy con tăng dần là 2 3 4 và 2 7

77. Viết chương trình tìm mảng con tăng dần có tổng lớn nhất trong mảng một chiều.

78. (*) Viết chương trình nhập vào một dãy số a gồm n số nguyên ($n \leq 100$). Tìm và in ra dãy con tăng dài nhất

Ví dụ: Nhập dãy a : 1 2 3 6 4 7 8 3 4 5 6 7 8 9 4 5

Dãy con tăng dài nhất : 3 4 5 6 7 8 9

79. (**) Viết chương trình tách 1 mảng các số nguyên thành 2 mảng a và b, sao cho kết quả thu được là:

- Mảng a chứa toàn số lẻ tăng dần.
- Mảng b chứa toàn số chẵn giảm dần.

(*Không dùng sắp xếp*)

Hướng dẫn: Tìm vị trí chèn thích hợp khi trích phần tử từ mảng ban đầu.

Ví dụ: Mảng ban đầu: 9 3 8 2 7 5 1 0 10

Mảng a: 1 3 5 7 9

Mảng b: 10 8 2

80. (**) **Viết** chương trình in ra tam giác Pascal (dùng mảng một chiều).

81. Viết chương trình nhập vào dãy số a gồm n số thực ($n \leq 100$), nhập vào dãy số b gồm m số thực ($m \leq 100$).

- Hãy sắp xếp hai dãy theo thứ tự tăng dần.

- (*) Trộn 2 dãy trên thành dãy c sao cho dãy c vẫn có thứ tự tăng.
 - Xuất dãy a, b, c ra màn hình.
82. (*) Cho mảng C có n phần tử ($n < 200$), các phần tử là các chữ số trong hệ đếm cơ số 16 (Hexa) (điều kiện mỗi phần tử $\leq n$). Hãy tách mảng C ra các mảng con theo điều kiện sau: các mảng con được giới hạn bởi hai lần xuất hiện thứ hai của con số trong dãy.
- Ví dụ: 123A4518B23 → có các dãy con là 123A451, 23A4518B2, 23A4518B23*
83. (***) Cho hai số nguyên dương A, B. Hãy xác định hai số C, D tạo thành từ hai số A, B sao cho C là số lớn nhất, D là số nhỏ nhất. Khi gạch đi một số chữ số trong C (D), thì các số còn lại giữ nguyên tạo thành A, các chữ số bỏ đi giữ nguyên tạo thành B.
- Ví dụ: A = 52568, B = 462384 -> C = 54625682384, D = 45256236884.*
84. Viết chương trình nhập vào dãy số a gồm n số nguyên ($n \leq 100$).
- Hãy đảo ngược dãy đó.
- Ví dụ: Nhập a: 3 4 5 2 0 4 1
Dãy sau khi đảo: 1 4 0 2 5 4 3*
- (*) Hãy kiểm tra xem dãy đã cho có thứ tự chưa (dãy được gọi là thứ tự khi là dãy tăng hoặc dãy giảm).
85. Cho mảng A có n phần tử hãy cho biết mảng này có đối xứng hay không.
86. (***) Hãy viết chương trình phát sinh ngẫu nhiên mảng các số nguyên gồm 10.000 phần tử, mỗi phần tử có giá trị từ 0 đến 32.000 và xây dựng hàm thống kê số lần xuất hiện các phần tử trong mảng, sau đó cho biết phần tử nào xuất hiện nhiều lần nhất.
- Ví dụ: Mảng: 5 6 11 4 4 5 4
5 xuất hiện 2 lần
6 xuất hiện 1 lần
11 xuất hiện 1 lần
4 xuất hiện 3 lần*
- 4 xuất hiện nhiều lần nhất**
87. Cho mảng A có n phần tử. Nhập vào số nguyên k ($k \geq 0$), dịch phải xoay vòng mảng A k lần.
- Ví dụ: Mảng A: 5 7 2 3 1 9
Nhập k = 2
Dịch phải xoay vòng mảng A: 1 9 5 7 2 3*

III. KẾT LUẬN

- ❖ Dữ liệu kiểu mảng dùng cho việc biểu diễn những thông tin có **cùng kiểu** dữ liệu liên tiếp nhau.
- ❖ Khi cài đặt bài tập mảng một chiều nên **xây dựng thành những hàm chuẩn** để dùng lại cho các bài tập khác.
- ❖ Các thao tác trên mảng đều theo quy tắc nhất định, chúng ta có thể ứng dụng mảng trong việc biểu diễn số lớn, dùng bảng tra, khử đệ qui, ...

CHƯƠNG 5 CHUỖI KÝ TỰ

Chuỗi ký tự là trường hợp đặc biệt của mảng một chiều. Chương này mô tả một số hàm thư viện thao tác trên chuỗi và các kỹ thuật cài đặt xử lý trên chuỗi.

I. TÓM TẮT LÝ THUYẾT

I.1. Khái niệm

Chuỗi ký tự là một dãy các phần tử, mỗi phần tử có kiểu ký tự.

Lưu ý: Chuỗi ký tự được kết thúc bằng ký tự '\0'. Do đó khi khai báo độ dài của chuỗi luôn luôn khai báo dư 1 phần tử để chứa ký tự '\0'.

Ví dụ: char S[5]="CNTT" //khai báo chuỗi có 5 phần tử kiểu char và gán dãy ký tự CNTT và chuỗi.

| | | | | |
|--------------|--------------|--------------|--------------|--------------|
| C | N | T | T | \0 |
| Phần tử S[0] | Phần tử S[1] | Phần tử S[2] | Phần tử S[3] | Phần tử S[4] |

Chuỗi rỗng là chuỗi chưa có ký tự nào trong mảng ký hiệu ""

I.2. Khai báo chuỗi

Để khai báo một chuỗi, ta có 2 cách khai báo sau :

❖ Cách 1: Con trỏ hằng

```
char < Tên chuỗi > [ < Số ký tự tối đa của chuỗi > ] ;
```

Ví dụ: char chuoi[25];

Ý nghĩa khai báo **1 mảng kiểu ký tự tên là chuoi** có 25 phần tử (như vậy tối đa ta có thể nhập 24 ký tự vì **phần tử thứ 25 đã chứa ký tự kết thúc chuỗi '\0'**)

❖ Cách 2: Con trỏ

```
char * < Tên chuỗi >;
```

Ví dụ : char *chuoi;

I.3. Các thao tác trên chuỗi

a. Nhập chuỗi

```
Cú pháp : char *gets(char *s);
```

Nhận các ký tự nhập từ phím cho đến khi nhấn phím Enter và đưa vào s.

Ví dụ:

```
void main()
{
    char chuoi[80];
    printf("Nhap vao chuoi:");
    gets(chuoi);
    printf("Chuoi vua nhap la: %s\n", chuoi);
}
```

b. Xuất chuỗi

Cú pháp : int puts(const char *s);

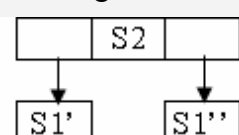
Xuất chuỗi s ra màn hình.

Ví dụ:

```
void main()
{
    char chuoi[] = "Vi du xuat chuoi\n";
    puts(string);
}
```

c. Các hàm thư viện (string.h)

| STT | TÊN HÀM | CHỨC NĂNG | VÍ DỤ |
|-----|---|---|--|
| 1 | int strlen(char s[]); | Trả về độ dài của chuỗi s. | char *s = "Borland International"; printf("Do dai s: %d\n", strlen(s)); Kết quả: Do dai s: 21 |
| 2 | strcpy(char dest[], char src[]); | Sao chép nội dung chuỗi src vào chuỗi dest. | char dest[10]; char *src = "abcdefghi"; strcpy(dest, src); printf("%s\n", dest); Kết quả: abcdefghi |
| 3 | strncpy(char dest[], char src[], int n); | Chép n ký tự từ chuỗi src sang chuỗi dest. Nếu chiều dài src < n thì hàm sẽ điền khoảng trắng cho đủ n ký tự vào dest. | char dest[4]; char *src = "abcdefghi"; strncpy(dest, src, 3); printf("%s\n", dest); Kết quả: abc |
| 4 | strcat(char s1[],char s2[]); | Nối chuỗi s2 vào chuỗi s1. | char *s1 = "Khoa "; char *s2 = "CNTT"; strcat(s1, s2); printf("%s\n", s1); Kết quả: Khoa CNTT |
| 5 | strncat(char s1[],char s2[],int n) | Nối n ký tự đầu tiên của chuỗi s2 vào chuỗi s1. | char *s1 = "Khoa "; char *s2 = "CNTT"; strncat(s1, s2, 2); printf("%s\n", s1); Kết quả: Khoa CN |
| 6 | Int strcmp(char s1[],char s2[]) | So sánh 2 chuỗi s1 và s2 theo nguyên tắc <u>thứ tự từ điển</u> . Phân biệt chữ hoa và thường. | char *s1 = "abcd"; char *s2 = "abCD"; if(strcmp(s1, s2)==0) |

| | | | |
|----|--|---|--|
| | | <p>Trả về:</p> <ul style="list-style-type: none"> • 0 : nếu s1 bằng s2. • >0: nếu s1 lớn hơn s2. • <0: nếu s1 nhỏ hơn s2. | <pre>printf("Giống nhau"); else printf("Khac nhau"); Kết quả: Khac nhau</pre> |
| 7 | int strcmp(char s1[], char s2[], int n) | Tương tự như strcmp(), nhưng chỉ so sánh n ký tự đầu tiên của hai chuỗi. | <pre>char *s1 = "abcd"; char *s2 = "abef"; if(strcmp(s1, s2, 2)==0) printf("Giống nhau"); else printf("Khac nhau"); Kết quả: Giống nhau</pre> |
| 6 | int stricmp(char s1[], char s2[]) | Tương tự như strcmp(), nhưng không phân biệt hoa thường. | <pre>char *s1 = "abcd"; char *s2 = "abCD"; if(stricmp(s1, s2)==0) printf("Giống nhau"); else printf("Khac nhau"); Kết quả: Giống nhau</pre> |
| 7 | int strnicmp(char s1[], char s2[], int n); | Tương tự như strcmp(), nhưng chỉ so sánh n ký tự đầu tiên của hai chuỗi. | <pre>char *s1 = "aBcd"; char *s2 = "Abef"; if(strnicmp(s1, s2, 2)==0) printf("Giống nhau"); else printf("Khac nhau"); Kết quả: Giống nhau</pre> |
| 8 | char *strchr(char s[], char c); | <p>Tìm lần xuất hiện đầu tiên của ký tự c trong chuỗi s. Trả về:</p> <ul style="list-style-type: none"> • NULL: nếu không có. • Địa chỉ c: nếu tìm thấy. | <pre>char s[15]; char *ptr, c = 'm'; strcpy(s, "Vi du tim ky tu"); ptr = strchr(s, c); if (ptr) printf("Ky tu %c tai: %d", c, ptr-s); else printf("Khong tim thay"); t quả: Ky tu m tai: 8</pre> |
| 9 | char *strstr(char s1[], char s2[]); | <p>Tìm sự xuất hiện đầu tiên của chuỗi s2 trong chuỗi s1. Trả về:</p> <ul style="list-style-type: none"> • NULL: nếu không có. • Ngược lại: Địa chỉ bắt đầu chuỗi s2 trong s1. | <pre>char *s1 = "Borland International"; char *s2 = "nation", *ptr; ptr = strstr(s1, s2); printf("Chuoi con: %s\n", ptr); Kết quả: Chuoi con: national</pre> |
| 10 | char *strtok(char s1[], char s2[]); | <ul style="list-style-type: none"> • Nếu s2 có xuất hiện trong s1: Tách chuỗi s1 thành hai chuỗi: Chuỗi đầu là những ký tự cho đến khi gặp chuỗi s2 đầu tiên, chuỗi sau là những ký tự còn lại của s1 sau khi đã bỏ đi chuỗi s2 xuất hiện trong s1. <p>S1: </p> | <pre>char input[16] = "abc,d"; char *p; // Lay chuoi dau p = strtok(input, ","); if (p) printf("S11: %s\n", p); /*Lay chuoi con lai, tham so dau la NULL*/ p = strtok(NULL, ","); if (p) printf("S12: %s\n", p); Kết quả: <u>S11: abc</u> <u>S12: d</u></pre> |

| | | | |
|--|--|---|--|
| | | <ul style="list-style-type: none"> Nếu s2 không xuất hiện trong s1 thì kết quả chuỗi tách vẫn là s1. | |
| <p>🔗 Lưu ý: Cách truy xuất các ký tự tương tự như mảng một chiều.</p> | | | |

d. Ví dụ

Nhập vào một chuỗi ký tự, xuất ra màn hình chuỗi bị đảo ngược thứ tự các ký tự.

Ví dụ: Nhập vào: *Tran minh thai*. Xuất ra màn hình: *iaht hnim narT*

```
#include<stdio.h>
#include<string.h>
#include<conio.h>

void DaoChuoi(char *s1, char *s2)
{
    int l=strlen(s1);
    for(int i=0; i<l; i++)
        s2[i]=s1[l-i-1];

    s2[i]='\0';
}

void main()
{
    char *s1, *s2;
    clrscr();
    printf("\nNhap vao chuoi ky tu: ");
    gets(s1);
    DaoChuoi(s1, s2);
    printf("\nKet qua sau khi dao nguoc chuoi: %s", s2);
}
```

II. BÀI TẬP

II.1. Bài tập cơ bản

- Cho biết kết quả của đoạn chương trình sau:

```
char input[20]="Truong cao dang CNTT", *p, *temp;
strcpy(temp, input);
do
{
    p = strtok(temp, " ");
    printf("%s\n",p);
```

```

    p = strtok(NULL, "");
    strcpy(temp, p);
}while(p!=NULL);
printf("Chuoi temp: %s \n Chuoi input: %s", temp, input);

```

2. Cho biết kết quả của đoạn chương trình sau:

```

char s1[20]="Truong cao dang CNTT", s1[10]="Tp. HCM", *input, *s3;
strcpy(input, s1);    strcpy(s3,"aeiou"); strcat(input, s2);
int n=strlen(input), k=0;
printf("Chuoi: %s",input);
for(int i=0; i<n; i++)
{
    if(strchr(s3, input[i]))
        k++;
}
printf("\nKet qua: %d", k);

```

3. Viết chương trình nhập vào một chuỗi ký tự, đếm số ký tự có trong chuỗi.
4. Viết chương trình đếm có bao nhiêu khoảng trắng trong chuỗi.
5. Viết chương trình nhập vào một chuỗi, hãy loại bỏ những khoảng trắng thừa trong chuỗi.
6. Viết chương trình nhập vào hai chuỗi s1 và s2, nối chuỗi s2 vào s1. Xuất chuỗi s1 ra màn hình.
7. Đổi tất cả các ký tự có trong chuỗi thành chữ thường (không dùng hàm `strlwr`).
8. Đổi tất cả các ký tự trong chuỗi sang chữ in hoa (không dùng hàm `struppr`).
9. Viết chương trình đổi những ký tự đầu tiên của mỗi từ thành chữ in hoa.
10. Viết chương trình đổi chữ xen kẽ 1 chữ hoa và 1 chữ thường.
Ví dụ: nhập ABCDEfgh đổi thành AbCdEfGh
11. Viết chương trình đảo ngược các ký tự trong chuỗi .
Ví dụ: nhập ABCDE, xuất ra màn hình là:EDCBA
12. Viết chương trình tìm kiếm 1 ký tự xem có trong chuỗi hay không, nếu có xuất ra vị trí của từ đó.
13. Viết 1 chương trình đếm một ký tự xuất hiện bao nhiêu lần trong chuỗi.

14. Viết chương trình tìm kiếm tên trong chuỗi họ tên. Nếu có thì xuất ra là tên này đã nhập đúng, ngược lại thông báo là đã nhập sai.
15. Viết chương đảo vị trí của từ đầu và từ cuối.
Ví dụ: nhập “bo an co” xuất ra “co an bo”
16. Viết hàm cắt chuỗi họ tên thành chuỗi họ lót và chuỗi tên.
Ví dụ: chuỗi họ tên là: “Nguyễn Văn A” cắt ra 2 chuỗi là chuỗi họ lót: “Nguyễn Văn”, chuỗi tên là: “A”
17. Nhập một chuỗi bất kỳ, sau đó hỏi người dùng cần tách bắt đầu từ đâu trong chuỗi trở về sau.
Ví dụ: Nhập chuỗi S1: “trường Cao Đẳng Công Nghệ Thông tin”. Người nhập muốn tách bắt đầu từ chữ “Công” thì sẽ xuất ra chuỗi “Công Nghệ Thông Tin” ra màn hình.
18. Viết hàm kiểm tra xem chuỗi có đối xứng hay không?.
19. Viết hàm tra xem trong chuỗi có ký tự số hay không nếu có tách ra thành một mảng số riêng.
20. Nhập một chuỗi bất kì, yêu cầu nhập 1 ký tự muốn xóa. Thực hiện xóa tất cả những ký tự đó trong chuỗi.
21. Viết chương trình tìm kiếm xem ký tự nào xuất hiện nhiều nhất trong chuỗi.
22. Viết 1 chương trình xoá một từ nào đó trong chuỗi.
Ví dụ: Chuỗi ban đầu: “CAO DANG CNTT”
Nhập: “CNTT”, và kết quả xuất ra: “CAO DANG”

II.2. Bài tập luyện tập và nâng cao

23. Đổi các từ ở đầu câu sang chữ hoa và những từ không phải đầu câu sang chữ thường.
Ví dụ: nGuYen vAN a đổi thành: Nguyễn Văn A
24. (*) Viết chương trình đảo ngược thứ tự các từ có trong chuỗi
Ví dụ: Nhập Truong CD CNTT TpHCM
Xuất ra màn hình là: TpHCM CNTT CD Truong
25. Nhập 1 chuỗi bất kì, liệt kê xem mỗi ký tự xuất hiện mấy lần.
26. Viết hàm kiểm tra xem trong 2 chuỗi có bao nhiêu ký tự giống nhau.
27. Viết chương trình mình chạy từ trái qua phải màn hình.

28. Viết 1 chương trình chèn 1 từ ở bất cứ vị trí nào mà người dùng yêu cầu.
29. (*) Viết chương trình nhập vào một chuỗi đếm xem chuỗi có bao nhiêu từ. Các từ cách nhau bằng khoảng trắng, dấu chấm câu: dấu chấm (.), dấu phẩy (,), dấu chấm phẩy (;), dấu hỏi (?) và dấu chấm than (!).
30. (**) Viết chương trình hiển thị một chuỗi ký tự. Chương trình cho phép di chuyển dấu nháy sang trái, sang phải, lên dòng hay xuống dòng bằng phím mũi tên, chèn hay xoá ký tự tại vị trí dấu nháy.

III. KẾT LUẬN

- ❖ Cũng giống như kiểu mảng một chiều, thao tác truy xuất các phần tử trên chuỗi hoàn toàn tương tự. Bên cạnh đó, kiểu dữ liệu này còn được cài đặt sẵn một số hàm thư viện rất hữu ích nên trong quá trình thao tác trên chuỗi nên khi cài đặt ta cố gắng **tận dụng tối đa những hàm liên quan**.
- ❖ **Không nên sử dụng hàm scanf()** để nhập chuỗi trong trường hợp chuỗi dữ liệu nhập vào có chứa khoảng trắng.
- ❖ Nếu **nhập chuỗi phía sau hàm scanf()** nên chèn hàm **fflush(stdin)** hoặc hàm **flushall()** giữa **scanf** và **gets()** để xóa vùng đệm, tránh trường hợp chương trình **bỏ qua hàm gets()** do trong vùng đệm còn lưu ký tự xuống dòng của phím **ENTER**.
- ❖ Khi thao tác trên chuỗi lưu ý phải **đảm bảo chuỗi được kết thúc bằng ký tự kết thúc '\0'**.

CHƯƠNG 6 MẢNG HAI CHIỀU

Đây là kiểu dữ liệu dùng để biểu diễn dữ liệu kiểu bảng, kiểu dữ liệu này rất thích hợp cho các bài toán liên quan đến đồ thị, biểu diễn ảnh, ...

I. TÓM TẮT LÝ THUYẾT

I.1. Khái niệm

Mảng hai chiều thực chất là mảng một chiều trong đó mỗi phần tử của mảng là một mảng một chiều, và được truy xuất bởi hai chỉ số dòng và cột.

Từ khái niệm trên ta có thể đưa ra một khái niệm về mảng nhiều chiều như sau:
mảng có từ hai chiều trở lên gọi là mảng nhiều chiều.

I.2. Khai báo mảng

Từ khái niệm trên ta có cú pháp khai báo mảng hai chiều như sau:

- Cách 1: Con trỏ hằng

< Kiểu dữ liệu > < Tên mảng > [< Số dòng tối đa >][< Số cột tối đa >];

Ví dụ:

```
int A[10][10]; // Khai báo mảng 2 chiều kiểu int gồm 10 dòng, 10 cột
float b[10][10]; // Khai báo mảng 2 chiều kiểu float gồm 10 dòng, 10 cột
```

- Cách 2 : Con trỏ

< Kiểu dữ liệu > **<Tên mảng>;

Ví dụ :

```
int **A ; // Khai báo mảng động 2 chiều kiểu int
float **B ; // Khai báo mảng động 2 chiều kiểu float
```

Tương tự như mảng một chiều, để sử dụng ta phải cấp phát vùng nhớ cho nó bằng malloc hoặc calloc và huỷ sau khi dùng bằng free

Ví dụ : Khai báo mảng các số nguyên A có kích thước 5x6

```
int **A;
A = (int **) malloc (5) ;
for (int i = 0 ; i < 5 ; i ++)
    A[i]=(int *) malloc (6) ;
```

I.3. Truy xuất phần tử của mảng

Để truy xuất các thành phần của mảng hai chiều ta phải dựa vào chỉ số dòng và chỉ số cột.

Ví dụ:

$int\ A[3][4] = \{ \{2,3,9,4\}, \{5,6,7,6\}, \{2,9,4,7\} \};$

Với các khai báo như trên ta có :

$A[0][0] = 2; A[0][1] = 3;$

$A[1][1] = 6; A[1][3] = 6;$

Với ví dụ trên ta có hình dạng của một ma trận như sau

| | | | | |
|---|---|---|---|---|
| | 0 | 1 | 2 | 3 |
| 0 | 2 | 3 | 9 | 4 |
| 1 | 5 | 6 | 7 | 6 |
| 2 | 2 | 9 | 4 | 7 |

⚠ Lưu ý: Khi nhập liệu cho mảng hai chiều, nếu là mảng các số nguyên thì ta nhập liệu theo cách thông thường. Nhưng nếu là mảng các số thực thì ta phải thông qua biến trung gian.

Ví dụ :

```
float a[10][10];           // Mảng số thực a
float tmp;                // Biến trung gian tmp
scanf ("%f", &tmp);       // Nhập liệu cho biến trung gian
a[2][2] = tmp;            // Gán dữ liệu vào phần tử a[2][2]
```

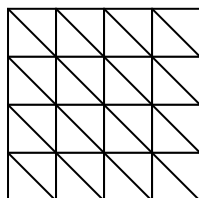
I.4. Ma trận vuông và các khái niệm liên quan

a. Khái niệm

Là ma trận có số dòng và số cột bằng nhau.

b. Tính chất của ma trận vuông

- **Đường chéo loại 1**



- Đường chéo loại 1 bao gồm đường chéo chính và những đường chéo song song với đường chéo chính. Trong đó đường chéo chính là đường chéo có :

chỉ số dòng = chỉ số cột

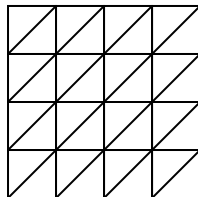
- Truy xuất các phần tử trên đường chéo loại 1 : để truy xuất các phần tử trên các đường chéo loại 1 ta có thể dựa vào chỉ số dòng và chỉ số cột như sau :

$$\text{cột} - \text{dòng} = \text{hằng số}$$

Ví dụ : Cho ma trận vuông $A(n \times n)$. Gọi (i_0, j_0) là tọa độ điểm xuất phát, ta có thể duyệt đường chéo xuất phát từ (i_0, j_0) như sau :

```
for ( i = i0, j = j0; i < n ; i ++, j ++ )
    printf ("%4d", A[i][j]);
```

- **Đường chéo loại 2:**



- Đường chéo loại 2 bao gồm đường chéo phụ và những đường song song với nó. Trong đó đường chéo phụ là đường chéo có:

$$\text{chỉ số cột} + \text{chỉ số dòng} = \text{số dòng (hoặc số cột)}$$

- Truy xuất các phần tử trên đường chéo loại 2 : để truy xuất các phần tử trên các đường chéo loại 1 ta có thể dựa vào chỉ số dòng và chỉ số cột như sau :

$$\text{cột} + \text{dòng} = \text{hằng số}$$

Ví dụ: Cho ma trận vuông $A(n \times n)$. Gọi (i_0, j_0) là tọa độ điểm xuất phát, ta có thể duyệt đường chéo xuất phát từ (i_0, j_0) như sau :

```
for ( i = i0, j = j0; i < n && j >= 0 ; i ++, j -- )
    printf ("%4d", A[i][j]);
```

II. BÀI TẬP

Để đơn giản trong việc khai báo ma trận, ta định nghĩa **kiểu ma trận các phần tử với kiểu dữ liệu bất kỳ** như sau:

```
#define MAX 100
typedef <kiểu dữ liệu> MATRAN[MAX][MAX];
```

Ví dụ: Khai báo ma trận các số nguyên a.

```
#define MAX 100
```

```
typedef int MATRAN[MAX][MAX];
MATRAN a;
```

II.1. Một số kỹ thuật cơ bản

- Phương pháp nhập xuất ma trận

```
void Nhap (MATRAN a, int &d, int &c)
{
    printf ("\nNhap so dong: ");
    scanf ("%d", &d);
    printf ("\nNhap so cot: ");
    scanf ("%d", &c);
    for (int i = 0; i < d; i++)
        for (int j = 0; j < c; j++)
            {
                printf (" a[%d][%d] = ", i, j);
                scanf ("%d", &a[i][j]);
            }
}

void Xuat (MATRAN a, int d, int c)
{
    printf ("\nNoi dung ma tran:\n");
    for (int i = 0; i < d; i++)
        {
            for (int j = 0; j < c; j++)
                printf ("\t %d ", a[i][j]);
            printf ("\n");
        }
}
```

- Kỹ thuật đặt cờ hiệu

Viết hàm kiểm tra xem trong ma trận các số nguyên có tồn tại các số nguyên lẻ lớn hơn 100 không?

```
int KiemTraLe (MATRAN a, int d, int c)
{
    int flag = 0; //tra ve 1 neu co nguoc lai tra ve 0

    for (int i = 0; i < d; i++)
        for (int j = 0; j < c; j++)
            if ( a[i][j] % 2 != 0 && a[i][j] > 100 )
                {
                    flag = 1;
                    break;
                }
    return flag;
}
```

- **Kỹ thuật đặt lính canh**

Viết hàm tìm phần tử nhỏ nhất trong ma trận.

```
int Min (MATRAN a, int d, int c)
{
    int min = a[0][0];
    for (int i = 0; i < d; i++)
        for (int j = 0; j < c; j++)
            if (a[i][j] < min)
                min = a[i][j];
    return min;
}
```

- **Phương pháp tính tổng**

Viết hàm tính tổng các phần tử trong ma trận.

```
long Tong (MATRAN a, int d, int c)
{
    long tong = 0;

    for (int i = 0; i < d; i++)
        for (int j = 0; j < c; j++)
            tong += a[i][j];
    return tong;
}
```

- **Phương pháp sắp xếp**

Viết hàm sắp xếp ma trận tăng dần từ trên xuống dưới và từ trái sang phải không dùng mảng phụ.

```
void SapTang(MATRAN a, int d, int c)
{
    for (int i = 0; i <= d*c-2; i++)
        for (int j = 0; j <= d*c-1; j++)
            if (a[i/c][i%c] < a[j/c][j%c])
            {
                int tmp = a[i/c][i%c];
                a[i/c][i%c] = a[j/c][j%c];
                a[j/c][j%c] = tmp;
            }
}
```

- **Phương pháp đếm**

Viết hàm đếm các phần tử chẵn trong ma trận.

```
int DemChan (MATRAN a, int d, int c)
{
    int dem = 0;
```

```

for ( int i = 0 ; i < d ; i ++ )
    for ( int j = 0 ; j < c ; j ++ )
        if ( a[i][j] % 2 == 0 )
            dem ++;
return dem;
}

```

II.2. Bài tập cơ bản

a. Bài tập nhập xuất

- Viết hàm nhập ma trận các số nguyên dương (*nhập sai báo lỗi và không cho nhập*).
- Viết hàm nhập/ xuất ma trận các số thực.
- Viết hàm in ra những phần tử có ký số tận cùng là 5.
- Viết chương trình in ra các phần tử nằm trên 2 đường chéo.
- Viết hàm in ra các phần tử nằm phía trên đường chéo phụ của ma trận vuông các số nguyên.
- Viết hàm in ra các phần tử nằm phía dưới đường chéo phụ của ma trận vuông các số nguyên.
- Viết hàm in ra các phần tử nằm phía trên đường chéo chính của ma trận vuông các số nguyên.
- Viết hàm in ra các phần tử nằm phía dưới đường chéo chính của ma trận vuông các số nguyên.
- Viết chương trình khởi tạo giá trị các phần tử là ngẫu nhiên cho ma trận các số nguyên kích thước $m \times n$.
- Viết hàm tạo ma trận a các số nguyên gồm 9 dòng 14 cột. Trong đó phần tử $a[i][j] = i * j$
- Viết hàm in tam giác Pascal với chiều cao h.

Ví dụ : h = 5

```

1
1 1
1 2 1
1 3 3 1
1 4 6 4 1

```

b. Bài tập tính tổng

- Viết hàm tính tổng các phần tử trên cùng một dòng.

13. Viết hàm tính tổng các phần tử trên cùng một cột.
14. Viết hàm tính tổng các phần tử chẵn có trong ma trận.
15. Viết hàm tính tổng các phần tử nằm trên đường chéo chính của ma trận vuông.
16. Viết hàm tính tổng các phần tử là số nguyên tố có trong ma trận.
17. Viết hàm tính tổng các số hoàn thiện trong ma trận các số nguyên.
18. Viết hàm tính tổng các giá trị lớn nhất trên mỗi dòng.
19. Viết hàm tính giá trị trung bình của các phần tử nhỏ nhất trên mỗi cột.
20. Viết hàm tính tổng các giá trị nhỏ nhất nằm trên từng đường chéo loại 2.
21. Viết hàm tìm đường chéo có tổng lớn nhất trong các đường chéo loại 1.

c. Bài tập tìm kiếm

22. Viết hàm tìm vị trí phần tử lớn nhất trong ma trận các số nguyên.
23. Viết hàm tìm vị trí phần tử nhỏ nhất trong ma trận các số nguyên.
24. Viết hàm tìm vị trí phần tử chẵn cuối cùng trong ma trận các số nguyên.
25. Viết hàm tìm phần tử âm lẻ lớn nhất trong ma trận.
26. Viết hàm tìm phần tử chẵn dương và nhỏ nhất trong ma trận.
27. Viết hàm tìm số hoàn thiện đầu tiên trong ma trận các số nguyên.
28. Viết hàm tìm số hoàn thiện lớn nhất trong ma trận các số nguyên.
29. Viết hàm tìm vị trí phần tử nguyên tố cuối cùng trong ma trận các số nguyên.
30. Viết hàm tìm phần tử lớn nhất nằm trên đường chéo chính của ma trận vuông.
31. Viết hàm in các số nguyên tố nằm trên đường chéo phụ của ma trận vuông.
32. Viết hàm tìm trong 2 ma trận các số nguyên, những phần tử giống nhau.
33. Viết hàm tìm phần tử nhỏ nhất trên mỗi đường chéo loại 2 của ma trận.
34. Viết hàm tìm và liệt kê những phần tử cực đại trong ma trận (một phần tử được coi là cực đại khi nó lớn hơn các phần tử xung quanh nó).
35. Viết hàm tìm dòng có tổng lớn nhất trong ma trận các số thực.
36. Viết hàm tìm cột có tổng nhỏ nhất trong ma trận các số nguyên.

d. Bài tập đếm

37. Viết hàm đếm các giá trị âm, dương trong ma trận các số thực.
38. Viết hàm đếm các giá trị chẵn, lẻ trong ma trận các số nguyên.

39. Viết hàm đếm số lần xuất hiện của phần tử x trong ma trận các số thực.
40. Viết hàm đếm các giá trị nhỏ hơn x trong ma trận các số thực.
41. Viết hàm đếm các phần tử nguyên tố trong ma trận các số nguyên.
42. Viết hàm đến các phần tử nguyên tố trên đường chéo chính của ma trận vuông các số nguyên.
43. Viết hàm đếm các giá trị chẵn trên đường chéo chính của ma trận vuông các số nguyên.
44. Viết hàm đếm các giá trị là bội của 3 và 5 trên đường chéo chính của ma trận các số nguyên.
45. Viết hàm đếm các giá trị nguyên tố trên 2 đường chéo (chính, phụ) của ma trận vuông các số nguyên.
46. Viết hàm đếm các giá trị cực đại trong ma trận các số nguyên.
47. Viết hàm đếm các giá trị cực tiểu trong ma trận các số nguyên.
48. Viết hàm đếm các cực trị trong ma trận các số nguyên (một phần tử được coi là cực trị khi nó là giá trị cực đại hay cực tiểu).
49. Viết hàm đếm các giá trị là số hoàn thiện trong ma trận các số nguyên.

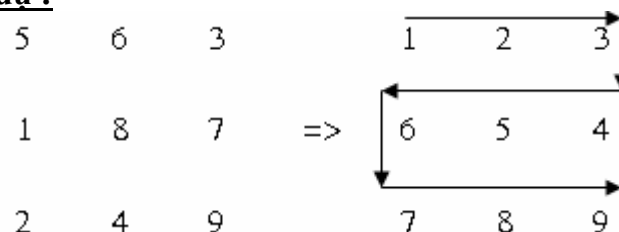
e. Bài tập sắp xếp

50. Viết hàm sắp xếp ma trận theo thứ tự tăng dần từ trên xuống dưới và từ trái qua phải theo phương pháp dùng mảng phụ.

Hướng dẫn: Đổ ma trận sang mảng một chiều, sắp xếp trên mảng một chiều theo thứ tự tăng dần, sau đó chuyển ngược mảng một chiều thành ma trận kết quả.

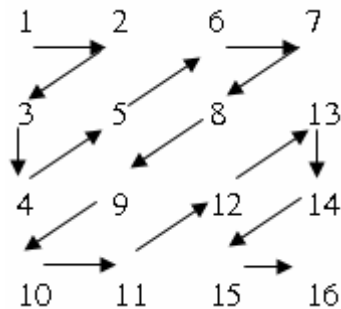
51. Viết hàm sắp xếp ma trận theo thứ tự giảm dần từ trên xuống dưới và từ trái sang phải.
52. Viết hàm sắp xếp các dòng trên ma trận theo thứ tự tăng dần.
53. Viết hàm sắp xếp các cột trên ma trận theo thứ tự giảm dần.
54. Viết hàm sắp xếp ma trận theo đường ziczắc ngang.

Ví dụ :



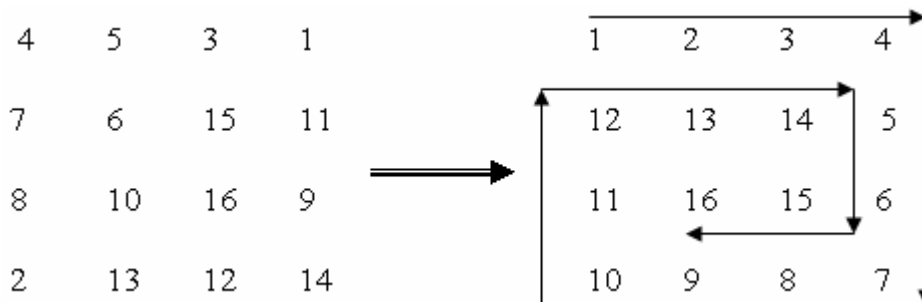
55. Viết hàm sắp xếp ma trận theo đường ziczắc chéo

Ví dụ :



56. Viết hàm sắp xếp ma trận theo đường xoắn ốc từ ngoài vào trong theo chiều kim đồng hồ.

Ví dụ :



57. Cho ma trận vuông, viết hàm sắp xếp tăng dần các phần tử nằm trên các đường chéo song song với đường chéo chính.
58. Viết chương trình nhập một ma trận vuông các số nguyên, và thực hiện những công việc sau :
- Sắp xếp các phần tử nằm trên các đường chéo loại 1 tăng dần
 - Sắp xếp các phần tử nằm trên các đường chéo loại 2 giảm dần.
 - Sắp xếp với điều kiện: các phần tử trên đường chéo chính tăng, các phần tử trên các đường chéo song song với đường chéo chính giảm.

f. Bài tập Thêm – Xoá – Thay thế

59. Viết hàm xoá một dòng i trên ma trận.
60. Viết hàm xoá một cột j trên ma trận.
61. Viết hàm xoá dòng có tổng lớn nhất trên ma trận.
62. Viết hàm hoán vị dòng có tổng lớn nhất với dòng có tổng nhỏ nhất.
63. Viết hàm tìm và thay thế các phần tử chẵn trong ma trận bằng ước số nhỏ nhất của nó.
64. Viết hàm thay thế những phần tử có giá trị x thành phần tử có giá trị y trong ma trận (x , y nhập từ bàn phím).

II.3. Bài tập luyện tập và nâng cao

65. Viết chương trình tính tổng, tích của hai ma trận các số nguyên.
66. Viết hàm kiểm tra xem ma trận vuông các số nguyên có đối xứng qua đường chéo chính hay không.
67. Viết hàm kiểm tra xem trong ma trận vuông cấp n có hàng nào trùng nhau hay không, nếu có thì chỉ rõ những hàng nào. (Trùng giá trị và vị trí).
68. Viết chương trình nhập vào ma trận vuông kích thước $n \times n$ ($2 \leq n \leq 100$).
Hãy viết hàm thực hiện những công việc sau :
 - In ra các phần tử trên 4 đường biên của ma trận.
 - Tính tổng các phần tử trên biên.
69. (*) Viết chương trình xoay ma trận các số thực 90^0 ngược chiều kim đồng hồ.

Ví dụ:

| | | | | | | | | |
|----|----|----|----|----|---|---|----|----|
| 1 | 2 | 3 | 4 | | 4 | 8 | 12 | 16 |
| 5 | 6 | 7 | 8 | => | 3 | 7 | 11 | 15 |
| 9 | 10 | 11 | 12 | | 2 | 6 | 10 | 14 |
| 13 | 14 | 15 | 16 | | 1 | 5 | 9 | 13 |

70. Viết chương trình dịch phải xoay vòng một cột trong ma trận các số thực.
71. Viết chương trình dịch xuống xoay vòng một dòng trong ma trận các số thực.
72. (*) Cho ma trận A ($m \times n$) các số nguyên hãy phát sinh ma trận B sao cho B là ma trận lật ngược của ma trận A .

Ví dụ :

| | | | | | | | | |
|----|----|----|----|----|----|----|----|----|
| 1 | 2 | 3 | 4 | | 4 | 3 | 2 | 1 |
| 5 | 6 | 7 | 8 | => | 8 | 7 | 6 | 5 |
| 9 | 10 | 11 | 12 | | 12 | 11 | 10 | 9 |
| 13 | 14 | 15 | 16 | | 16 | 15 | 14 | 13 |

73. (**) Cho ma trận A ($m \times n$) hãy phát sinh ma trận B ($m \times n$) sao cho phần tử $B(i, j)$ là trung bình cộng của các phần tử trong hình vuông 3×3 tâm tại (i, j) của A .

Ví dụ :

| | | | | | | | | |
|----|---|----|----|----|---|---|---|---|
| 1 | 5 | 2 | 6 | | 3 | 2 | 4 | 4 |
| 4 | 2 | 3 | 6 | => | 4 | 4 | 4 | 4 |
| 8 | 7 | 9 | 1 | | 5 | 6 | 6 | 7 |
| 10 | 2 | 12 | 13 | | 6 | 8 | 7 | 8 |

78. Nhập vào mảng hai chiều gồm n dòng và m cột các số nguyên. Hãy tìm phần tử lớn nhất trên mỗi dòng và đồng thời nhỏ nhất trên mỗi cột, hoặc lớn nhất trên mỗi cột và đồng thời nhỏ nhất trên mỗi dòng. Có bao nhiêu phần tử như thế?

Ví dụ:

| | | | |
|----------|----------|---|---|
| 3 | 6 | 2 | 1 |
| 4 | 7 | 6 | 9 |
| 5 | 15 | 8 | 7 |

79. Viết chương trình tạo ngẫu nhiên một ma trận các số nguyên (0 -> 50), tìm những phần tử cực đại (là phần tử lớn hơn các phần tử xung quanh).

Ví dụ :

| | | | |
|----------|---|----------|---|
| 2 | 6 | 8 | 4 |
| 9 | 7 | 5 | 3 |
| 6 | 2 | 8 | 1 |

80. (***) Cho ma trận các số nguyên $A_{m \times n}$ ($n \geq 3, m \geq 3$). Hãy tìm ma trận con (3x3) có tổng lớn nhất.

Ví dụ :

| | | | | | | | |
|----|----|----|----|----|----|----|----|
| 1 | 2 | 3 | 4 | | 6 | 7 | 8 |
| 5 | 6 | 7 | 8 | => | 10 | 11 | 12 |
| 9 | 10 | 11 | 12 | | 14 | 15 | 16 |
| 13 | 14 | 15 | 16 | | | | |

81. Nhập ma trận vuông cấp $n \times n$ ($n < 10$). In ra các phần tử của ma trận này theo hướng của đường chéo chính.

Ví dụ : $n = 4$

| | | | | | | | | |
|---|---|---|---|----|---|---|---|---|
| | | | | | 1 | | | |
| | | | | | 9 | 3 | | |
| 1 | 3 | 7 | 4 | | 3 | 5 | 7 | |
| 9 | 5 | 6 | 2 | => | 2 | 4 | 6 | 4 |
| 3 | 4 | 7 | 5 | | 3 | 7 | 2 | |
| 2 | 3 | 1 | 6 | | 1 | 5 | | |
| | | | | | 6 | | | |

82. (***) Hãy điền các số từ 1 đến n^2 vào ma trận cấp n ($n > 2$), chỉ xét trường hợp n là số lẻ với tính chất P là tổng các số bằng nhau.

Hướng dẫn : Ma phương của một bảng vuông cấp n , trong mỗi ô nhận một giá trị sao cho, mỗi hàng, mỗi cột và mỗi đường chéo đều thỏa mãn một tính chất P nào đó cho trước.

Ví dụ : Với $n = 5$

| | | | | |
|----|----|----|----|----|
| 1 | 18 | 25 | 2 | 9 |
| 10 | 12 | 19 | 21 | 3 |
| 4 | 6 | 13 | 20 | 22 |
| 23 | 5 | 7 | 14 | 16 |
| 17 | 24 | 1 | 8 | 15 |

83. (*) Viết hàm in ma trận các số nguyên dương theo qui luật được mô tả như sau : các phần tử phía trên đường chéo phụ là giá trị bình phương của các giá trị $1 \rightarrow n \times 2$, các giá trị từ đường chéo phụ trở xuống là các số nguyên tố. Ma trận được sắp xếp như ví dụ bên dưới.

Ví dụ : $n = 5$

| | | | | |
|----|----|----|-----|----|
| 1 | 9 | 36 | 100 | 31 |
| 4 | 25 | 81 | 37 | 17 |
| 16 | 64 | 41 | 19 | 7 |
| 49 | 43 | 23 | 11 | 3 |
| 47 | 29 | 13 | 5 | 2 |

84. Cho ma trận vuông a cấp n (n lẻ, $3 \leq n \leq 15$), mỗi phần tử đều có giá trị nguyên dương. Hãy xây dựng hàm kiểm tra xem ma trận a có phải là ma phương hay không?
85. (***) Viết chương trình giải bài toán 8 hậu. Hãy đặt 8 con hậu trên bàn cờ 8×8 sao cho chúng không ăn nhau (2 hậu ăn nhau khi cùng hàng, cùng cột và cùng nằm trên đường chéo).

Hướng dẫn:

Dùng ma trận 8×8 để lưu bàn cờ. Mỗi ô có 3 trạng thái :

- Có hậu 1
- Ô trống 0
- Ô không được đi -1

86. (***) Viết chương trình giải bài toán mã đi tuần. Hãy đi con mã 64 lượt đi trên bàn cờ 8×8 sao cho mỗi ô chỉ đi qua một lần (xuất phát từ một ô bất kỳ)

Hướng dẫn :

Đứng tại một ô trên bàn cờ con mã có thể đi được 1 trong 8 hướng sau .

| | | | | | | | |
|--|---|---|---|---|---|--|--|
| | | | | | | | |
| | | | | | | | |
| | | 1 | | 2 | | | |
| | 8 | | | | 3 | | |
| | | | • | | | | |
| | 7 | | | | 4 | | |
| | | 6 | | 5 | | | |
| | | | | | | | |

Khai báo 8 hướng đi của mã như sau:

```
typedef struct DIEM
```

```
{
```

```
    int x, y;
```

```
};
```

DIEM huongdi[8]={{-2,-1},{-2,1},{-1,2},{1,2},{2,1},{2,-1},{1,-2},{-1,-2}};

Trong đó mỗi thành phần của huongdi là độ lệch của dòng và cột so với vị trí của con mã.

Ví dụ: huongdi[0] (tức đi đến vị trí 1 như hình vẽ) có độ lệch 2 dòng và 1 cột. (Giá trị âm biểu thị độ lệch về bên trái cột hay hướng lên của dòng).

Chọn vị trí đi kế tiếp sao cho vị trí đó phải gần với biên hay góc nhất (tức số đường đi có thể đi là ít nhất).

87. Viết chương trình giải bài toán Taci. Cho ma trận vuông 3x3 gồm các số nguyên từ 0 -> 8 trong đó 0 là ô trống. Bài toán đặt ra là hãy đưa ma trận ở một trạng thái đầu về trạng thái đích, mỗi lần chỉ dịch chuyển được 1 ô.

Ví dụ: Trạng thái đầu

| | | |
|---|---|---|
| 1 | 3 | 0 |
| 8 | 2 | 5 |
| 7 | 4 | 6 |

Trạng thái đích

| | | |
|---|---|---|
| 1 | 2 | 3 |
| 8 | 0 | 4 |
| 7 | 6 | 5 |

=>

III. KẾT LUẬN

- ❖ Kiểu dữ liệu mảng hai chiều được ứng dụng rộng rãi trong các bài toán về tìm đường đi trong đồ thị, xử lý ảnh, xử lý những dữ liệu dạng bảng, ...
- ❖ Lưu ý khi nhập mảng hai chiều **các số thực** phải thông qua 1 biến trung gian.

CHƯƠNG 7 KIỂU DỮ LIỆU CÓ CẤU TRÚC

Cung cấp cơ chế cho phép khai báo các kiểu dữ liệu mới để giải quyết theo yêu cầu của bài toán dựa vào những kiểu dữ liệu cơ bản được cài đặt sẵn trong ngôn ngữ lập trình.

I. TÓM TẮT LÝ THUYẾT

I.1. Khái niệm

Cấu trúc (struct) thực chất là một kiểu dữ liệu do người dùng định nghĩa bằng cách gom nhóm các kiểu dữ liệu cơ bản có sẵn trong C thành một kiểu dữ liệu phức hợp nhiều thành phần.

I.2. Định nghĩa kiểu dữ liệu

Cú pháp

```
struct < tên cấu trúc >
{
    Các kiểu dữ liệu thành phần ;
};
```

Ngoài ra ta có thể dùng từ khoá **typedef** để định nghĩa một tên mới cho kiểu dữ liệu đã có.

Cú pháp

```
typedef struct < tên cấu trúc > < tên mới >;
```

Ví dụ 1: Kiểu dữ liệu DATE gồm các thành phần:

- Thứ (thu): chuỗi có tối đa 4 ký tự.
- Ngày (ngay): số nguyên 1 byte.
- Tháng (thang): số nguyên 1 byte.
- Năm (nam): số nguyên 2 bytes.

Ta định nghĩa DATE như sau:

```
struct DATE
{
    char thu[5];
    unsigned char ngay;
    unsigned char thang;
    int nam;
};
typedef struct DATE d;
```

Kiểu dữ liệu có cấu trúc có thể lồng vào nhau.

Ví dụ 2: Định nghĩa kiểu dữ liệu của học sinh HOCSINH gồm:

- Mã số học sinh (MSHS): chuỗi có tối đa 5 ký tự.
- Họ tên (hoten): chuỗi có tối đa 30 ký tự.
- Ngày tháng năm sinh (ngaysinh): kiểu DATE.
- Địa chỉ (diachi): chuỗi có tối đa 50 ký tự.
- Giới tính (phai): chuỗi có tối đa 3 ký tự.
- Điểm trung bình (diemtb): số thực.

Ta định nghĩa kiểu HOCSINH như sau:

```
struct DATE
{
    char thu[5];
    unsigned char ngay;
    unsigned char thang;
    int nam;
};

typedef struct HOCSINH
{
    char MSHS[6];
    char hoten[31];
    struct DATE ngaysinh;
    char diachi[51];
    unsigned char phai[4];
    float diemtb;
};
```

⚠ Khi định nghĩa kiểu dữ liệu struct lồng nhau, ta cần lưu ý: Kiểu dữ liệu được sử dụng phải khai báo phía trên.

I.3. Khai báo

Khi ta định nghĩa kiểu dữ liệu tức là ta có một kiểu dữ liệu mới, muốn sử dụng ta phải khai báo biến. Cú pháp khai báo kiểu dữ liệu cũng giống như cách khai báo của các kiểu dữ liệu chuẩn.

```
struct < tên cấu trúc > < tên biến > ;
```

Ví dụ :

```
struct DATE x ; // Khai bao bien x co kieu du lieu DATE
```

Tuy nhiên nếu ta định nghĩa struct có dùng từ khoá **typedef** thì ta có thể khai báo trực tiếp mà không cần từ khoá “**struct**”.

Ví dụ :

DATE x ; // Khai báo biến x có kiểu DATE

***Biến con trỏ kiểu cấu trúc:** Ngoài cách khai báo như trên ta có thể khai báo theo kiểu con trỏ như sau

struct < tên cấu trúc > * < tên biến > ;

Để sử dụng ta cũng phải cấp phát vùng nhớ giống như kiểu dữ liệu chuẩn.

Ví dụ :

DATE *y; // Khai báo con trỏ y kiểu cấu trúc DATE

y = (DATE *) malloc (sizeof (DATE)) ;

I.4. Truy xuất

Để truy xuất một thành phần dữ liệu nào đó bên trong cấu trúc ta có 2 trường hợp truy xuất như sau :

- Biến x là một biến cấu trúc thông thường, ta dùng toán tử dấu chấm “.”

Cú pháp :

< Tên cấu trúc > . < Biến thành phần > ;

Ví dụ :

DATE x ; // khai báo biến x kiểu DATE

x.ngay = 5 ; // gan ngay bang 5

- Biến x là một biến con trỏ, ta dùng toán tử mũi tên “->” (Gồm dấu trừ ‘-’ và dấu lớn hơn ‘>’).

Cú pháp :

< Tên cấu trúc > -> < Biến thành phần > ;

Ví dụ :

DATE *x ; // khai báo biến x kiểu con trỏ DATE

x -> ngay = 5 ; // gan ngay bang 5

⊗ Đối với kiểu dữ liệu có struct lồng nhau phải truy cập đến thành phần cuối cùng có kiểu dữ liệu cơ bản.

Ví dụ: Giả sử, có kiểu HOCSINH như trên

HOCSINH hs; // khai báo biến hs kiểu HOCSINH

Muốn in học sinh A sinh vào tháng mấy ta phải truy cập như sau:


```
printf("Thang sinh cua hoc sinh A la: %d", (hs.ngaysinh).thang);
```

I.5. Ví dụ minh họa

Viết chương trình nhập vào tọa độ hai điểm trong mặt phẳng và tính tổng hai tọa độ này.

```
#include <conio.h>
#include <stdio.h>

typedef struct DIEM //khai bao mot kieu du lieu DIEM gom toa do x va y
{
    int x;
    int y;
};

void Nhap (DIEM &d)
{
    printf ("\nNhap vao tao do diem\n");
    printf ("Tung do : ");
    scanf ("%d", & d.x);
    printf ("Hoanh do : ");
    scanf ("%d", & d.y);
}

void Xuat (DIEM d)
{
    printf ("\nToa do diem : (%d , %d)", d.x, d.y);
}

DIEM Tong (DIEM d1, DIEM d2)
{
    DIEM temp;
    temp.x = d1.x + d2.x ;
    temp.y = d1.y + d2.y ;
    return Temp;
}

void main ()
{
    DIEM A , B, AB; //khai bao 3 diem A, B, AB;
    clrscr ();
    Nhap ( A );
    Xuat ( A );
    Nhap ( B );
    Xuat ( B );
    printf ("\n Tong cua hai diem vua nhap la : ");
    AB = Tong ( A, B );
    Xuat ( AB );
}
```

```

    getch ();
}

```

I.6. Mảng cấu trúc

- Cách khai báo tương tự như mảng một chiều hay ma trận (Kiểu dữ liệu bây giờ là kiểu dữ liệu có cấu trúc).
- Cách truy cập phần tử trong mảng cũng như truy cập trên mảng một chiều hay ma trận. Nhưng do từng phần tử có kiểu cấu trúc nên phải chỉ định rõ cần lấy thành phần nào, tức là phải truy cập đến thành phần cuối cùng có kiểu là dữ liệu cơ bản (xem lại bảng các kiểu dữ liệu cơ bản) .

I.7. Nguyên tắc viết chương trình có mảng cấu trúc

Do kiểu dữ liệu có cấu trúc thường chứa rất nhiều thành phần nên khi viết chương trình loại này ta cần lưu ý:

- Xây dựng hàm xử lý cho một kiểu cấu trúc.
- Muốn xử lý cho mảng cấu trúc, ta gọi lại hàm xử lý cho một kiểu cấu trúc đã được xây dựng bằng cách dùng vòng lặp.

Ví dụ 1: Cho một lớp học gồm n học sinh ($n \leq 50$). Thông tin của một học sinh được mô tả ở ví dụ 2, mục I.2. Hãy viết chương trình nhập và xuất danh sách học sinh sau đó đếm xem có bao nhiêu học sinh được lên lớp (Điều kiện được lên lớp là điểm trung bình ≥ 5.0).

Cách làm:

- Trước hết ta phải xây dựng hàm nhập và xuất cho 1 học sinh.
- Xây dựng hàm nhập và xuất ngày tháng năm (Kiểu dữ liệu DATE).
- Sau đó mới xây dựng hàm nhập và xuất cho danh sách học sinh.

```

#define MAX 50
struct DATE
{
    char thu[5];
    unsigned char ngay;
    unsigned char thang;
    int nam;
};
typedef struct HOCSINH
{
    char MSHS[6];
    char hoten[31];
    struct DATE ngaysinh;
}

```

```

        char diachi[51];
        unsigned char phai[4];
        float diemt;
    };

    void NhapNamSinh(DATE &d);
    void XuatNamSinh(DATE d);
    void Nhap1HS (HOCSINH &hs);
    void Xuat1HS (HOCSINH hs);
    void NhapDSHS(HOCSINH lh[], int &n);
    void XuatDSHS(HOCSINH lh[], int n);
    int DemHSLenLop(HOCSINH lh[], int n);

    void main()
    {
        HOCSINH lh[MAX]; // Khai báo mảng lh gồm có tối đa 50 học sinh
        int n, sohsdau;
        NhapDSHS(lh, n);
        XuatDSHS(lh, n);
        sohsdau = DemHSLenLop(lh, n);
        printf("\nSố lượng học sinh được lên lớp là: %d", sohsdau);
        getch();
    }

    void NhapNamSinh(DATE &d)
    {
        printf("\nNhập vào ngày: ");
        scanf("%u", &d.ngay);
        printf("\nNhập vào tháng: ");
        scanf("%u", &d.thang);
        printf("\nNhập vào năm: ");
        scanf("%d", &d.nam);
    }

    void XuatNamSinh(DATE d)
    {
        printf("%02u / %02u / %4d", d.ngay, d.thang, d.nam);
    }

    void Nhap1HS(HOCSINH &hs)
    {
        float d;

        lushall(); // Xóa vùng đệm
        printf("\nNhập mã số học sinh: ");
        gets(hs.MSHS);
        printf("\nNhập họ tên học sinh: ");
        gets(hs.hoten);
        printf("\nNhập ngày tháng năm sinh: ");
    }

```

```

    flushall(); //Xoa vung dem
    NhapNamSinh(hs.ngaysinh);
    printf("\nNhap vao dia chi: ");
    flushall(); //Xoa vung dem
    gets(hs.diachi);
    printf("\nPhai: ");
    gets(hs.phai);
    printf("\nNhap vao diem trung binh: ");
    flushall(); //Xoa vùng đệm
    scanf("%f", &d); //Nhập vào biến tạm d sau đó gán vào hs.diemtb
    hs.diemtb=d;
}

void NhapDSHS(HOCSINH lh[], int &n)
{
    printf("\nNhap vao so luong hoc sinh: ");
    scanf("%d", &n);
    for(int i=0; i<n; i++)
    {
        printf("\nNhap vao thong tin cua hoc sinh thu %d:\n", i+1);
        Nhap1HS(lh[i]); //Goi ham nhap thong tin 1 hoc sinh
    }
}

void Xuat1HS(HOCSINH hs)
{
    printf("\nMa so hoc sinh: %s", hs.MSHS);
    printf("\nHo ten hoc sinh: %s", hs.hoten);
    printf("\nNgay thang nam sinh: ");
    XuatNamSinh(hs.ngaysinh);
    printf("\nDia chi: %s", hs.diachi);
    printf("\nPhai: %s", hs.phai);
    printf("\nDiem trung binh: %.2f", hs.diemtb);
}

void XuatDSHS(HOCSINH lh[], int n)
{
    for(int i=0; i<n; i++)
    {
        printf("\n\nThong tin hoc sinh thu %d:", i+1);
        Xuat1HS(lh[i]); //Goi ham xuat thong tin 1 hoc sinh
    }
}

int DemHSLenLop(HOCSINH lh[], int n)
{
    int d=0;
    for(int i=0; i<n; i++)
        if(lh[i].diemtb>=5.0)

```

```
        d++;  
    return d;  
}
```

Kết quả ví dụ khi chạy chương trình:

Nhap vao thong tin cua hoc sinh thu 1:

Nhap ma so hoc sinh: 02313

Nhap ho ten hoc sinh: Nguyen Van A

Nhap ngay thang nam sinh:

Nhap vao ngay: 12

Nhap vao thang: 03

Nhap vao nam: 1980

Nhap vao dia chi: 60 Phan Dang Luu Q.Phu Nhuan

Phai: Nam

Nhap vao diem trung binh: 6.5

Nhap vao thong tin cua hoc sinh thu 2:

Nhap ma so hoc sinh: 03852

Nhap ho ten hoc sinh: Ly Thi B

Nhap ngay thang nam sinh:

Nhap vao ngay: 05

Nhap vao thang: 12

Nhap vao nam: 1981

Nhap vao dia chi: 24 Ly Tu Trong Q.1

Phai: Nu

Nhap vao diem trung binh: 3.5

Thong tin hoc sinh thu 1:

Ma so hoc sinh: 02313

Ho ten hoc sinh: Nguyen Van A

Ngay thang nam sinh: 12 / 03 / 1980

Dia chi: 60 Phan Dang Luu Q.Phu Nhuan

Phai: Nam

Diem trung binh: 6.50

Thong tin hoc sinh thu 2:

Ma so hoc sinh: 03852

Ho ten hoc sinh: Ly Thi B

Ngay thang nam sinh: 05 / 12 / 1981

Dia chi: 24 Ly Tu Trong Q.1

Phai: Nu

Diem trung binh: 3.50

So luong hoc sinh duoc len lop la: 1

Ví dụ 2: Cho một mảng các phân số (**PHANSO**) gồm n phần tử ($n \leq 50$). Hãy viết chương trình nhập và xuất danh sách các phân số sau đó tìm phân số có giá trị lớn nhất, tổng và tích các phân số và nghịch đảo giá trị các phân số trong mảng.

Cách làm:

- Trước hết ta phải xây dựng hàm nhập và xuất cho 1 phân số.
- Xây dựng hàm tính tổng, hiệu, tích, thương, rút gọn, so sánh và nghịch đảo cho 2 phân số.
- Sau đó mới xây dựng hàm nhập, xuất, tính tổng, tích cho mảng các phân số.

```
#define MAX 100
```

```
typedef struct PHANSO
```

```
{  
    int tu, mau;  
};
```

```
void NhapPS(PHANSO &ps);
```

```
void XuatPS(PHANSO ps);
```

```
void NhapMangPS(PHANSO dsps[], int &n);
```

```
void XuatMangPS(PHANSO dsps[], int n);
```

```
PHANSO TimMax(PHANSO dsps[], int n);
```

```
int KiemTra(PHANSO ps);
```

```
//Tra ve 1: Neu hop le
```

```
int USCLN(int a, int b);
```

```
PHANSO RutGon(PHANSO ps);
```

```
PHANSO NghichDao(PHANSO ps);
```

```
PHANSO Nhan(PHANSO ps1, PHANSO ps2);
```

```
PHANSO Chia(PHANSO ps1, PHANSO ps2);
```

```
PHANSO Tru(PHANSO ps1, PHANSO ps2);
```

```
PHANSO Cong(PHANSO ps1, PHANSO ps2);
```

```
int SoSanh(PHANSO ps1, PHANSO ps2);
```

```
//Tra ve 0: ps1=ps2
```

```
//Tra ve 1: ps1>ps2
```

```
//Tra ve -1: ps1<ps2
```

```
PHANSO TongCacPS(PHANSO dsps[], int n);
```

```
PHANSO TichCacPS(PHANSO dsps[], int n);
```

```
void NghichDaoCacPS(PHANSO dsps[], int n);
```

```
void main()
```

```
{
```

```
    int n;
```

```
    PHANSO a[MAX], max, s, p;
```

```
    clrscr();
```

```
    NhapMangPS(a, n);
```

```
    printf("\nMang cac phan so vua nhap: ");
```

```
    XuatMangPS(a, n);
```

```
    max=TimMax(a, n);
```

```
    printf("\nPhan so co gia tri lon nhat: ");
```

```
    XuatPS(max);
```

```
    s=TongCacPS(a, n);
```

```

    printf("\nTong gia tri cac phan so co trong mang: ");
    XuatPS(s);

    p=TichCacPS(a, n);
    printf("\nTich gia tri cac phan so co trong mang: ");
    XuatPS(p);

    NghichDaoCacPS(a, n);
    printf("\nMang phan so sau khi nghich dao cac phan tu: ");
    XuatMangPS(a, n);

    getch();
}

void NhapPS(PHANSO &ps)
{
    do{
        printf("\nNhap tu so: ");
        scanf("%d", &ps.tu);
        printf("\nNhap mau so: ");
        scanf("%d", &ps.mau);
        if(!KiemTra(ps))
            printf("\nMau so khong duoc bang 0, nhap lai phan so\n");
        else
            break;
    } while(1);
    ps=RutGon(ps);
}

void XuatPS(PHANSO ps)
{
    printf("%d", ps.tu);
    if(ps.tu&&ps.mau!=1)
        printf("/%d", ps.mau);
}

void NhapMangPS(PHANSO dsps[], int &n)
{
    printf("\nNhap so luong phan so: ");
    scanf("%d", &n);
    for(int i=0; i<n; i++)
    {
        printf("\nNhap vao phan so thu %d: ", i+1);
        NhapPS(dsps[i]);
    }
}

void XuatMangPS(PHANSO dsps[], int n)
{

```

```
for(int i=0; i<n; i++)
{
    XuatPS(dsps[i]);
    printf("t");
}

int KiemTra(PHANSO ps)
{
    if(ps.mau==0)
        return 0;
    return 1;
}

int USCLN(int a, int b)
{
    a=abs(a);
    b=abs(b);
    while(a!=b)
    {
        if(a>b)
            a=a-b;
        else
            b=b-a;
    }
    return a;
}

PHANSO RutGon(PHANSO ps)
{
    int us;
    if(ps.tu==0)
        return ps;

    us=USCLN(ps.tu, ps.mau);
    ps.tu=ps.tu/us;
    ps.mau=ps.mau/us;
    return ps;
}

PHANSO NghichDao(PHANSO ps)
{
    PHANSO kq;
    kq.tu=ps.mau;
    kq.mau=ps.tu;
    return kq;
}

PHANSO Nhan(PHANSO ps1, PHANSO ps2)
```



```

{
    PHANSO kq;
    kq.tu=ps1.tu*ps2.tu;
    kq.mau=ps1.mau*ps2.mau;
    kq=RutGon(kq);
    return kq;
}

PHANSO Chia(PHANSO ps1, PHANSO ps2)
{
    PHANSO kq;
    kq=Nhan(ps1, NghichDao(ps2));
    return kq;
}

PHANSO Tru(PHANSO ps1, PHANSO ps2)
{
    PHANSO kq;
    kq.tu=ps1.tu*ps2.mau-ps1.mau*ps2.tu;
    kq.mau=ps1.mau*ps2.mau;
    kq=RutGon(kq);
    return kq;
}

PHANSO Cong(PHANSO ps1, PHANSO ps2)
{
    PHANSO kq;
    kq.tu=ps1.tu*ps2.mau+ps1.mau*ps2.tu;
    kq.mau=ps1.mau*ps2.mau;
    kq=RutGon(kq);
    return kq;
}

int SoSanh(PHANSO ps1, PHANSO ps2)
{
    ps1=RutGon(ps1);
    ps2=RutGon(ps2);
    if(ps1.tu==ps2.tu&&ps1.mau==ps2.mau)
        return 0;
    if(ps1.tu*ps2.mau>ps2.tu*ps1.mau)
        return 1;
    return -1;
}

PHANSO TimMax(PHANSO dsps[], int n)
{
    PHANSO max;
    max=dsps[0];
    for(int i=1; i<n; i++)

```

```

        if(SoSanh(dsps[i], max)==1)
            max=dsps[i];
        return max;
    }

    PHANSO TongCacPS(PHANSO dsps[], int n)
    {
        PHANSO s=dsps[0];
        for(int i=1; i<n; i++)
        {
            s=Cong(s, dsps[i]);
        }
        return s;
    }

    PHANSO TichCacPS(PHANSO dsps[], int n)
    {
        PHANSO p=dsps[0];
        for(int i=1; i<n; i++)
        {
            p=Nhan(p, dsps[i]);
        }
        return p;
    }

    void NghichDaoCacPS(PHANSO dsps[], int n)
    {
        for(int i=0; i<n; i++)
        {
            dsps[i]=NghichDao(dsps[i]);
        }
    }

```

Kết quả ví dụ khi chạy chương trình:

Nhap so luong phan so: 5

Nhap vao phan so thu 1:

Nhap tu so: 1

Nhap mau so: 3

Nhap vao phan so thu 2:

Nhap tu so: 7

Nhap mau so: 4

Nhap vao phan so thu 3:

Nhap tu so: 9

Nhap mau so: 7

Nhap vao phan so thu 4:

Nhap tu so: 5

Nhap mau so: 6

Nhap vao phan so thu 5:

Nhap tu so: 4

Nhap mau so: 7

Mang cac phan so vua nhap: 1/3 7/4 9/7 5/6 4/7

Phan so co gia tri lon nhat: 7/4

Tong gia tri cac phan so co trong mang: 401/84

Tich gia tri cac phan so co trong mang: 5/14

Mang phan so sau khi nghich dao cac phan tu: 3 4/7 7/9 6/5 7/4

II. BÀI TẬP

II.1. Bài tập cơ bản

1. Viết chương trình sử dụng con trỏ cấu trúc để hiển thị giờ, phút, giây ra màn hình, và tính khoảng cách giữa 2 mốc thời gian.
2. Viết chương trình sử dụng con trỏ cấu trúc thể hiện ngày, tháng, năm ra màn hình, và tính khoảng cách giữa 2 ngày.
3. Viết chương trình khai báo kiểu dữ liệu thể hiện một số phức. Sử dụng kiểu này để viết hàm tính tổng, hiệu, tích của hai số phức.
4. Viết chương trình khai báo kiểu dữ liệu để biểu diễn một phân số. Hãy viết hàm thực hiện những công việc sau:
 - Tính tổng, hiệu, tích, thương hai phân số.
 - Rút gọn phân số.
 - Quy đồng hai phân số.
 - So sánh hai phân số.
5. Viết chương trình khai báo kiểu dữ liệu để biểu diễn một hỗn số. Hãy viết hàm thực hiện những công việc sau :
 - Đổi hỗn số sang phân số
 - Tính tổng, tích hai hỗn số
6. Viết chương trình khai báo kiểu dữ liệu để biểu diễn một điểm trong hệ tọa độ Oxy . Hãy viết hàm thực hiện các công việc sau:
 - Tìm những điểm đối xứng của nó qua tung độ, hoành độ, tọa độ tâm.
 - Hãy tính tổng, hiệu, tích của hai điểm trong mặt phẳng tọa độ Oxy.
 - Tính khoảng cách giữa hai điểm.

7. Cho một hình trụ có các thông tin sau: BanKinh (bán kính hình trụ kiểu số thực), ChieuCao (chiều cao hình trụ kiểu số thực). Hãy thực hiện các công việc sau.
- Nhập dữ liệu cho hình trụ trên.
 - Tính diện tích xung quanh, diện tích toàn phần, thể tích hình trụ.

II.2. Bài Tập Luyện Tập

8. Viết chương trình tạo một mảng các số phức. Hãy viết hàm tính tổng, tích các số phức có trong mảng.
9. Viết chương trình tạo một mảng các phân số. Hãy viết hàm thực hiện các công việc sau :
- Tính tổng tất cả các phân số (kết quả dưới dạng phân số tối giản)
 - Tìm phân số lớn nhất, phân số nhỏ nhất.
 - Sắp xếp mảng tăng dần.
10. Viết chương trình khai báo kiểu dữ liệu STACK (cơ chế LIFO). Viết hàm làm những công việc sau :
- Kiểm tra STACK rỗng
 - Kiểm tra STACK đầy
 - Thêm phần tử vào STACK
 - Lấy phần tử ra khỏi STACK
11. Tổ chức dữ liệu để quản lí sinh viên bằng cấu trúc mẫu tin trong một mảng N phần tử, mỗi phần tử có cấu trúc như sau:
- Mã sinh viên.
 - Tên.
 - Năm sinh.
 - Điểm toán, lý, hoá, điểm trung bình.

Viết chương trình thực hiện những công việc sau:

- Nhập danh sách các sinh viên cho một lớp học.
- Xuất danh sách sinh viên ra màn hình.
- Tìm sinh viên có điểm trung bình cao nhất.
- Sắp xếp danh sách lớp theo thứ tự tăng dần của điểm trung bình.
- Sắp xếp danh sách lớp theo thứ tự giảm dần của điểm toán.

- Tìm kiếm và in ra các sinh viên có điểm trung bình lớn hơn 5 và không có môn nào dưới 3.
- Tìm sinh viên có tuổi lớn nhất.
- Nhập vào tên của một sinh viên. Tìm và in ra các thông tin liên quan đến sinh viên đó (nếu có).

12. Tổ chức dữ liệu quản lý danh mục các bộ phim VIDEO, các thông tin liên quan đến bộ phim này như sau:

- Tên phim (tựa phim).
- Thể loại (3 loại : hình sự, tình cảm, hài).
- Tên đạo diễn.
- Tên diễn viên nam chính.
- Tên diễn viên nữ chính.
- Năm sản xuất.
- Hãng sản xuất

Viết chương trình thực hiện những công việc sau :

- Nhập vào bộ phim mới cùng với các thông tin liên quan đến bộ phim này.
- Nhập một thể loại: In ra danh sách các bộ phim thuộc thể loại này.
- Nhập một tên nam diễn viên. In ra các bộ phim có diễn viên này đóng.
- Nhập tên đạo diễn. In ra danh sách các bộ phim do đạo diễn này dàn dựng.

13. Một thư viện cần quản lý thông tin về các đầu sách. Mỗi đầu sách bao gồm các thông tin sau : MaSSach (mã số sách), TenSach (tên sách), TacGia (tác giả), SL (số lượng các cuốn sách của đầu sách). Viết chương trình thực hiện các chức năng sau:

- Nhập vào một danh sách các đầu sách (tối đa là 100 đầu sách)
- Nhập vào tên của quyển sách. In ra thông tin đầy đủ về các sách có tên đó, nếu không có thì tên của quyển sách đó thì báo là :Không Tìm Thấy.
- Tính tổng số sách có trong thư viện.

14. Viết chương trình tạo một mảng danh sách các máy tính của một cửa hàng, thông tin của một máy tính bao gồm :

- Loại máy
- Nơi sản xuất

- *Thời gian bảo hành*

- Viết hàm nhập một dãy các loại máy tính có thông tin như trên.
- Hãy viết hàm thống kê xem có bao nhiêu máy có thời gian bảo hành là 1 năm.
- In ra danh sách các máy tính có xuất xứ từ Mỹ.

15. Để lắp ráp một máy vi tính hoàn chỉnh cần phải có tối thiểu 10 linh kiện loại A và có thể lắp bổ sung thêm vào khoảng tối đa 8 linh kiện loại B. Tại một cửa hàng vi tính cần quản lý bán hàng các loại linh kiện tại cửa hàng. Thông tin về một loại linh kiện gồm có: Tên linh kiện, quy cách, loại, đơn giá loại 1 (chất lượng tốt – số nguyên), đơn giá loại 2 (chất lượng thường – số nguyên). Viết chương trình thực hiện những công việc sau :

- Nhập vào thông tin về các linh kiện có ở cửa hàng.
- Xuất danh sách các linh kiện đã nhập theo thứ tự tăng dần của loại linh kiện và tên linh kiện.
- Cho biết đã có đủ 10 linh kiện loại A cần thiết lắp ráp máy hay chưa?

16. Một cửa hàng cần quản lý các mặt hàng, thông tin một mặt hàng bao gồm:

- *Mã hàng.*
- *Tên mặt hàng.*
- *Số lượng.*
- *Đơn giá.*
- *Số lượng tồn.*
- *Thời gian bảo hành (tính theo đơn vị tháng).*

- Hãy nhập vào một danh sách các mặt hàng.
- Tìm mặt hàng có số lượng tồn nhiều nhất.
- Tìm mặt hàng có số lượng tồn ít nhất.
- Tìm mặt hàng có giá tiền cao nhất.
- In ra những mặt hàng có thời gian bảo hành lớn hơn 12 tháng.
- Sắp xếp các mặt hàng theo thứ tự tăng dần của số lượng tồn.

17. Viết chương trình quản lý hồ sơ nhân viên trong một công ty, chương trình thực hiện những công việc sau :

- *Họ và tên.*
- *Phái.*

- Ngày sinh.
 - Địa chỉ.
 - Lương cơ bản.
 - Bảo hiểm xã hội.
 - Thưởng.
 - Phạt.
 - Lương thực lĩnh = lương cơ bản + thưởng – BH xã hội – phạt.
- Nhập vào hồ sơ của các nhân viên trong công ty.
 - Xuất danh sách các nhân viên theo lương thực lĩnh giảm dần bằng 2 cách sau :
 - Cấp phát vùng nhớ tĩnh.
 - Cấp phát vùng nhớ động.
18. (*) Viết chương trình quản lý lớp học của một trường. Các thông tin của một lớp học như sau :
- Tên lớp.
 - Sĩ số.
 - Danh sách các sinh viên trong lớp.
- Nhập vào danh sách các lớp với thông tin yêu cầu như trên.
 - In danh sách các lớp có trên 5 sinh viên có điểm trung bình loại giỏi.
 - Tìm lớp có nhiều sinh viên nhất.
 - Tìm lớp có ít sinh viên nhất.
 - Tìm sinh viên có điểm trung bình cao nhất.
 - Tìm lớp có số lượng sinh viên đạt điểm trung bình loại giỏi nhiều nhất.
19. Viết chương trình quản lý vé tàu, thông tin một vé tàu như sau :
- Ngày giờ khởi hành, ngày giờ đến.
 - Ga đi, ga đến.
 - Loại tàu, loại chỗ ngồi (ngồi, nằm, cứng, mềm).
 - Số toa, số ghế.
- Viết hàm nhập vào danh sách các vé tàu.
 - In danh sách các vé tàu có ga đến là Huế.
 - In danh sách các vé tàu có ga đến là Hà Nội và đi ngày 8/6/2005.
 - Đếm xem có bao nhiêu khách đi tàu loại chỗ ngồi là nằm cứng.

20. Viết chương trình tính tiền điện hàng tháng của các hộ gia đình, thông tin các khách hàng như sau :
- *Kỳ thu, từ ngày.....đến ngày.*
 - *Tên khách hàng, mã khách hàng.*
 - *Địa chỉ.*
 - *Điện năng tiêu thụ (Kwh).*
 - Nhập vào danh sách các khách hàng.
 - Xuất danh sách hoá đơn theo thứ tự tăng dần của điện năng tiêu thụ.
 - Tính tiền điện của các khách hàng theo quy định sau.
 - *100 kw đầu tiên là 550 đ / kw*
 - *50 kw tiếp theo là 900 đ / kw*
 - *50 kw tiếp theo là 1210 đ / kw*
 - *Thuế 10 % trên tổng số tiền phải trả*
 - Tính tổng số tiền thu được của các khách hàng.

III. KẾT LUẬN

- ❖ Kiểu dữ liệu có cấu trúc cho phép ta định nghĩa những kiểu dữ liệu bất kỳ trên cơ sở là những kiểu dữ liệu cơ bản có sẵn trong ngôn ngữ lập trình.
- ❖ Khi xây dựng xong kiểu dữ liệu mới ta phải **định nghĩa những thao tác** cho kiểu dữ liệu đó.
- ❖ Những kiểu dữ liệu tự định nghĩa này thông thường có rất nhiều thành phần, mỗi thành phần cũng có thể là một kiểu dữ liệu tự định nghĩa, vấn đề là ta chọn kiểu dữ liệu cơ bản nào để xây dựng nên chúng sao cho **phù hợp về mặt kiểu dữ liệu và phù hợp về kích thước lưu trữ** (vừa đủ).
- ❖ Các sử dụng những kiểu dữ liệu tự định nghĩa cũng giống như các kiểu dữ liệu cơ bản. Muốn sử dụng phải khai báo biến, khi truy cập các thành phần phải truy cập theo quy ước.
- ❖ Nếu thành phần cấu trúc có kiểu dữ liệu là số thực thì khi sử dụng hàm **scanf()** phải thông qua biến trung gian rồi gán lại cho thành phần cấu trúc đó.
- ❖ Đối với mảng các kiểu dữ liệu có cấu trúc ta nên **xử lý cho từng thành phần cấu trúc** rồi mới xử lý cho mảng cấu trúc bằng cách dùng vòng lặp.

Trong chương này, chúng ta sẽ tìm hiểu cấu trúc tập tin, cài đặt các thao tác, một số hàm thư viện và ứng dụng trong việc tổ chức dữ liệu trên tập tin.

I. TÓM TẮT LÝ THUYẾT

I.1. Khái niệm

Trong các chương trình trước thì các dữ liệu đưa vào chương trình chỉ được tồn tại trong RAM, khi thoát chương trình thì tất cả dữ liệu đều bị mất. Để khắc phục tình trạng này Borland C cung cấp cho ta các hàm để lưu trữ và truy xuất tập tin, đó là kiểu **FILE**. Và ở đây ta chỉ đề cập đến 2 loại tập tin :

- Tập tin văn bản: là tập tin dùng để ghi các ký tự lên đĩa theo các dòng.
- Tập tin nhị phân: là tập tin dùng để ghi các cấu trúc dạng nhị phân (được mã hoá).

I.2. Thao tác với tập tin

Quá trình thao tác trên tập tin thông qua 4 bước:

Bước 1: Khai báo con trỏ trỏ đến tập tin.

Bước 2: Mở tập tin.

Bước 3: Các xử lý trên tập tin.

Bước 4: Đóng tập tin.

a. Khai báo

FILE * < tên biến >;

Ví dụ : `FILE *f; // Khai bao bien con tro file f`

b. Mở tập tin

fopen (< đường dẫn tên tập tin > , < kiểu truy nhập >);

Ví dụ : `FILE *f; // Khai bao bien con tro f`
`f = fopen ("C:\\VD1.txt" , "rt");`

Các kiểu truy nhập tập tin thông dụng:

- t** là kiểu truy nhập tập tin đối với dạng tập tin văn bản (text).
- b** là kiểu truy nhập tập tin đối với dạng tập tin nhị phân (binary).
- r** mở ra để đọc (ready only).
- w** mở ra để ghi (create / write).
- a** mở ra để thêm vào (append).
- r+** mở ra để đọc và ghi (modify).

c. Các hàm đọc ghi nội dung tập tin

- Tập tin văn bản

| STT | TÊN HÀM | Ý NGHĨA SỬ DỤNG | VÍ DỤ |
|--------------------|--|---|---|
| ĐỌC TẬP TIN | | | |
| 1 | fscanf(<FILE *>, <định dạng>, <các tham biến>); | Đưa dữ liệu từ một tập tin theo định dạng. | <i>fscanf(f, "%d", &x);</i> |
| 2 | fgets(<vùng nhớ>, <kích thước tối đa>, <FILE *>); | Đọc một chuỗi ký tự từ một tập tin với kích thước tối đa cho phép, hoặc gặp ký tự xuống dòng. | <i>char s[80]; fgets(s, 80, f);</i> |
| 3 | getc(<FILE *>); | Đọc một ký tự từ tập tin đang mở. | <i>char c=getc(f);</i> |
| GHI TẬP TIN | | | |
| 1 | fprintf(<FILE *>, <định dạng>[, <các tham biến>]); | Ghi dữ liệu theo một định dạng nào đó vào tập tin. | <i>fprintf(f, "%d", x);</i> |
| 2 | fputs(<chuỗi ký tự>, <FILE *>); | Ghi một chuỗi ký tự vào tập tin đang mở. | <i>fputs("Giao trình BT", f);</i> |

- Tập tin nhị phân

| STT | TÊN HÀM | Ý NGHĨA SỬ DỤNG | VÍ DỤ |
|--------------------|--|--|--|
| ĐỌC TẬP TIN | | | |
| 1 | fread(&ptr, <size>, <len>, <FILE *>); | <ul style="list-style-type: none"> • ptr: vùng nhớ để lưu dữ liệu đọc. • size: kích thước mỗi ô nhớ (tính bằng byte). • len: độ dài dữ liệu cần đọc. FILE: đọc từ tập tin nhị phân nào. | <i>int a[30], b, n; fread(a, sizeof(int), n, f); Fread(&b, sizeof(int), 1, f);</i> |
| GHI TẬP TIN | | | |
| 1 | fwrite(&prt, <size>, <len>, <FILE *>); | Tham số tương tự như hàm fread. | <i>fwrite(a, sizeof(int), n, f);</i> |

d. Đóng tập tin

Sau khi không còn làm việc với tập tin, để đảm bảo an toàn cho dữ liệu thì nhất thiết ta phải đóng tập tin lại.

```
fclose ( < biến con trỏ tập tin > );
hoặc fcloseall ( );
```

Ví dụ : fclose (f);

e. Các thao tác khác trên tập tin*** Xoá tập tin :**

```
remove ( < đường dẫn tập tin > );
```

*** Đổi tên tập tin :**

```
rename ( < tên tập tin cũ > , < tên tập tin mới > );
```

*** Di chuyển con trỏ tập tin :**

```
fseek ( < FILE * > , < độ dời > , < mốc > );
```

Các mốc :

SEEK_SET dời đến đầu tập tin (giá trị 0).

SEEK_END dời đến cuối tập tin (giá trị 2).

SEEK_CUR dời vị trí hiện hành (giá trị 1).

Ví dụ :

```
fseek ( f , +5 , SEEK_CUR ); // dời vị trí hiện hành về cuối 5 bytes
```

```
fseek ( f , -4 , SEEK_CUR ); // dời vị trí hiện hành về trước 4 bytes
```

*** Cho biết vị trí con trỏ file:**

```
ftell ( < FILE * > );
```

Ví dụ :

```
int size = ftell ( f );
```

```
/*size: khoảng cách từ đầu tập tin đến vị trí hiện hành (tính bằng
byte)*/
```

f. Ví dụ minh hoạ

```
void KiemTra (FILE *f, char duongdan [])
{
    f = open ( duongdan , "rt");
    if ( f == NULL )
    {
        printf ("Khong mo duoc tap tin %s", duongdan);
        perror ("\nLy do");
    }
}
```

```

        getch ();
        return ;
    }
    printf ("Tập tin %s da duoc mo", duongdan);
    fclose (f);
}

```

I.3. Các ví dụ minh hoạ

a. Tập tin văn bản

Ví dụ 1: Viết chương trình tạo tập tin văn bản SO.OUT gồm n số nguyên, các số của dãy được tạo ngẫu nhiên có giá trị tuyệt đối không vượt quá M (n, M đọc từ tập tin SO.INP). Kết quả chương trình là 1 tập tin văn bản có dòng thứ nhất ghi số n ; n dòng tiếp theo ghi các số tạo được, mỗi số trên một dòng.

| SO.INP |
|--------|
| 3 |
| 10 |

| SO.OUT |
|--------|
| 3 |
| 5 |
| 7 |
| 2 |

```

#include < conio.h >
#include < stdio.h >

#define in "SO.INP"
#define out "SO.OUT"
int n, M ;

void Nhap ()
{
    FILE *fi;
    fi = fopen ( in , "rt" );
    fscanf ( fi, "%d %d ", &n, &M );
    fclose ( fi );
}

void Xuat ()
{
    FILE *fo;
    fo = fopen ( out , "wt " );
    fprintf ( fo , "%d\n", n );
    randomize ();
    for ( ; n > 0 ; n -- )
        fprintf ( fo , "%d\n", random ( ( 2 * M + 1 ) - M ) );
    fclose ( fo );
}

```

```

}
void main ()
{
    clrscr ();
    Nhap ();
    Xuat ();
}

```

Ví dụ 2: Viết chương trình phát sinh ngẫu nhiên ma trận a kích thước 5×6 , lưu ma trận này vào file **test.inp**. Đọc lại file **test.inp** đưa dữ liệu vào ma trận b và xuất ra màn hình xem kết quả lưu đúng không? Cấu trúc của file **test.inp** như sau:

- Dòng đầu lưu 2 số nguyên: m, n thể hiện số dòng và số cột của ma trận.
- m dòng tiếp theo, mỗi dòng gồm n phần tử là giá trị các phần tử trên một dòng của ma trận.

```

#include<stdio.h>
#include<conio.h>
#include<stdlib.h>

#define MAX 100
#define dl "test.inp"

void LuuFile(int a[MAX][MAX], int m, int n)
{
    FILE *f;
    f=fopen(dl, "wt");
    if(f==NULL)
    {
        printf("\nKhong tao duoc file.");
        getch();
        exit(0);
    }
    fprintf(f, "%d %d\n", m, n);
    for(int i=0; i<m; i++)
    {
        for(int j=0; j<n; j++)
            fprintf(f, "%d\t", a[i][j]);
        fprintf(f, "\n");
    }
    fclose(f);
}

void DocFile(int a[MAX][MAX], int &m, int &n)
{

```

```

FILE *f;
f=fopen(dl, "rt");
if(f==NULL)
{
    printf("\nKhong doc duoc file.");
    getch();
    exit(0);
}
fscanf(f, "%d%d", &m, &n);
for(int i=0; i<m; i++)
{
    for(int j=0; j<n; j++)
        fscanf(f, "%d", &a[i][j]);
}
fclose(f);
}

void main()
{
    int a[MAX][MAX], m=5, n=6, i, j;
    int b[MAX][MAX], x, y;

    randomize();
    for(i=0; i<m; i++)
        for(j=0; j<n; j++)
            a[i][j]=random(1000);
    LuuFile(a, m, n);
    DocFile(b, x, y);
    for(i=0; i<x; i++)
    {
        for(j=0; j<y; j++)
            printf("%d\t", b[i][j]);
        printf("\n");
    }
}

```

Kết quả ví dụ sau khi chạy chương trình, file test.inp có dạng sau:

| | | | | | |
|-----|-----|-----|-----|-----|-----|
| 5 | 6 | | | | |
| 480 | 661 | 395 | 736 | 998 | 987 |
| 31 | 414 | 211 | 801 | 774 | 416 |
| 166 | 191 | 454 | 830 | 508 | 72 |
| 121 | 382 | 35 | 365 | 567 | 726 |
| 159 | 309 | 1 | 275 | 870 | 378 |

b. Tập tin nhị phân

Viết hàm đọc/ ghi một danh sách sinh viên của một lớp vào tập tin SV.DAT

```
SINHVIEN ds[100];
```

```

int siso;

void nhap ()
{
    FILE *fi;
    fi = fopen ( "SV.DAT", "rb" );
    fseek ( fi, 0, SEEK_END );
    siso = ( ftell ( fi ) + 1 ) / sizeof ( SINHVIEN );
    fseek ( fi, 0, SEEK_SET );
    fread ( ds, sizeof ( SINHVIEN ), siso, fi );
    fclose ( fi );
}

void xuat ()
{
    FILE *fo;
    fo = fopen ( "SV.DAT", "wb" );
    fwrite ( ds, sizeof ( SINHVIEN ), siso, fo );
    fclose ( fo );
}

```

II. BÀI TẬP

II.1. Bài tập cơ bản

- Viết chương trình tạo tập tin văn bản chứa 1 dãy số nguyên bất kỳ.
- Viết chương trình tạo tập tin nhị phân chứa 10000 số nguyên bất kỳ ghi vào file SONGUYEN.INP. Mỗi dòng 10 số, sau đó viết chương trình đọc file SONGUYEN.INP, sắp xếp theo thứ tự tăng dần và lưu kết quả vào file SONGUYEN.OUT.
- Viết chương trình tạo một file chứa 10000 số nguyên ngẫu nhiên đôi một khác nhau trong phạm vi từ 1 đến 32767 và đặt tên là "SONGUYEN.INP".
- Viết chương trình tạo một file chứa các số nguyên có tên SONGUYEN.INP. Sau đó đọc file SONGUYEN.INP và ghi các số chẵn vào file SOCHAN.OUT và những số lẻ vào file SOLE.OUT.
- Viết chương trình ghi vào tập tin SOCHAN.DAT các số nguyên chẵn từ 0 đến 100.
- Viết chương trình đọc tập tin SOCHAN.DAT và xuất ra màn hình, mỗi dòng 30 số.
- Viết chương trình giả lập lệnh **COPY CON** để tạo tập tin văn bản. Khi kết thúc tập tin nhấn phím **F6** để lưu.

8. Viết chương trình giả lập lệnh **TYPE** để in nội dung của tập tin văn bản ra màn hình.
9. Viết chương trình kiểm tra một tập tin nào đó có trong một thư mục được chỉ định hay không?
10. Viết chương trình giả lập lệnh **DEL** để xoá tập tin. Yêu cầu nhập đường dẫn và tên tập tin, kiểm tra sự tồn tại của tập tin, nếu có thì xoá tập tin được chỉ định.
11. Viết chương trình giả lập lệnh **RENAME** để đổi tên một tập tin.
12. Viết chương trình tạo file văn bản có tên là “MATRIX.INP” có cấu trúc như sau:
 - Dòng đầu ghi hai số m, n .
 - Trong m dòng tiếp theo mỗi dòng ghi n số và các số các nhau một khoảng cách.

Hãy kiểm tra xem trong file đó có bao nhiêu số nguyên tố.

Kết quả cần ghi vào file “MATRIX.OUT” có nội dung là một số nguyên đó là số lượng các số nguyên tố trong file “MATRIX.INP”.

13. Cho số nguyên n , hãy in tam giác PASCAL gồm n dòng
Dữ liệu vào: tập tin văn bản PAS.INP gồm 1 dòng chứa giá trị n .
Kết quả: đưa ra tập tin văn bản PAS.OUT thể hiện một tam giác PASCAL n dòng.
14. Cho mảng các số nguyên , hãy sắp xếp mảng theo thứ tự tăng dần.
Dữ liệu vào : tập tin văn bản ARRAY.INP gồm 2 dòng
 - Dòng 1 chứa số nguyên n ($n \leq 100$).
 - Dòng 2 chứa n số nguyên.Kết quả : Đưa ra tập tin văn bản ARRAY.OUT gồm hai dòng
 - Dòng 1 chứa n phần tử của mảng các số nguyên.
 - Dòng 2 chứa n số nguyên được xếp tăng dần.
15. Cho mảng các số nguyên, tìm phần tử lớn nhất của mảng.
Dữ liệu vào: tập tin văn bản ARRAY.INP gồm hai dòng:
 - Dòng 1 chứa số nguyên n ($n \leq 100$).
 - Dòng 2 chứa n số nguyên.Kết quả: Đưa ra tập tin văn bản ARRAY.OUT gồm 1 dòng ghi 2 giá trị x, y trong đó x là giá trị lớn nhất, y là vị trí của x trong mảng.

II.2. Bài tập luyện tập và nâng cao

16. Cho mảng các số nguyên, tính tổng các phần tử của mảng.
 Dữ liệu vào : tập tin văn bản ARRAY.INP gồm hai dòng
 - Dòng 1 chứa số nguyên n ($n \leq 10$)
 - Dòng 2 chứa n số nguyên
 Kết quả : Đưa ra tập tin văn bản ARRAY.OUT gồm một dòng ghi tổng các phần tử trong mảng.
17. Cho mảng các số nguyên, hãy liệt kê các phần tử là số nguyên tố
 Dữ liệu vào : tập tin văn bản NT.INP gồm hai dòng
 - Dòng 1 chứa số nguyên n ($n \leq 100$)
 - Dòng 2 chứa n số nguyên
 Kết quả : đưa ra tập tin văn bản NT.OUT gồm hai dòng:
 - Dòng 1 chứa số lượng các phần tử nguyên tố trong mảng.
 - Dòng 2 liệt kê các số nguyên tố đó.
18. (*) Tạo file văn bản có tên là "INPUT.TXT" có cấu trúc như sau:
 - Dòng đầu tiên ghi N (N là số nguyên dương nhập từ bàn phím).
 - Trong các dòng tiếp theo ghi N số nguyên ngẫu nhiên trong phạm vi từ 0 đến 100, mỗi dòng 10 số (các số cách nhau ít nhất một khoảng trắng).

Hãy đọc dữ liệu của file "INPUT.TXT" và lưu vào mảng một chiều A.

Thực hiện các công việc sau :

- Tìm giá trị lớn nhất của mảng A.
- Đếm số lượng số chẵn, số lượng số lẻ của mảng A.
- Hãy sắp xếp các phần tử theo thứ tự tăng dần.

Hãy ghi các kết quả vào file văn bản có tên OUTPUT.TXT theo mẫu sau:

| INPUT.TXT | OUTPUT.TXT |
|---------------------------|----------------------------|
| 18 | Cau a: 99 |
| 87 39 78 19 89 4 40 98 29 | Cau b: 9 9 |
| 65 | Cau c: |
| 20 43 1 99 38 34 58 4 | 1 4 4 19 20 29 34 38 39 40 |
| | 43 58 65 78 87 89 98 99 |

19. (*) Viết chương trình nhập và lưu hồ sơ của sinh viên vào một file có tên là "DSSV.TXT". Sau đó đọc file "DSSV.TXT" và cất vào mảng, hãy sắp xếp các hồ sơ sinh viên theo thứ tự giảm dần theo điểm trung bình môn

học rồi in ra màn hình hồ sơ các sinh viên theo thứ tự đó ra màn hình có thông tin như sau :

- Mã số sinh viên.
- Họ và tên sinh viên.
- Điểm trung bình kiểm tra.
- Điểm thi hết môn.
- Điểm trung bình môn học (tính bằng (điểm TBKT+điểm thi)/2).

20. (*) Tạo một file text có tên là “INPUT.TXT” có cấu trúc như sau :

- Dòng đầu tiên ghi hai số M và N (M, N là hai số nguyên dương nhập từ bàn phím).
- Trong M dòng tiếp theo mỗi dòng ghi N số nguyên ngẫu nhiên trong phạm vi từ 0 đến 100 (các số này cách nhau ít nhất một khoảng trắng).

Hãy đọc dữ liệu từ file trên và lưu vào mảng hai chiều. Rồi thực hiện các công việc sau:

- Tìm giá trị lớn nhất của ma trận.
- Đếm số lượng số chẵn, lẻ, nguyên tố có trong ma trận.
- Hãy tính tổng các phần tử trên mỗi dòng của ma trận.

Hãy ghi kết quả này vào filetext có tên là “OUTPUT.TXT”

| INPUT.TXT | OUTPUT.TXT |
|-------------------|----------------------------|
| 6 6 | Cau a: 49 |
| 41 17 33 23 12 1 | Cau b: 17 19 |
| 44 24 23 49 5 24 | Cau c: 127 169 147 214 132 |
| 33 20 17 25 33 19 | 146 |
| 0 48 45 48 41 32 | |
| 10 24 36 19 19 24 | |
| 30 4 23 26 27 36 | |

21. (**) Xét dãy số a_1, a_2, \dots, a_N . Một đoạn con của dãy là dãy các phần tử liên tiếp nhau được xác định bởi chỉ số của số bắt đầu (L) và chỉ số của số cuối cùng (R). Tổng các số trên đoạn được gọi là tổng đoạn.

Yêu cầu : Cho dãy (a_N), hãy tìm đoạn con có tổng đoạn lớn nhất (T)

Dữ liệu được cho trong tập tin văn bản SUMMAX.INP

- Dòng thứ nhất chứa số nguyên N ($0 < N \leq 30000$)

- *N* dòng tiếp theo, mỗi dòng chứa một số là các số của dãy đã cho theo đúng thứ tự. Giá trị tuyệt đối của mỗi số không vượt quá 30000

Kết quả tìm được ghi vào tập tin văn bản SUMMAX.OUT gồm 1 dòng ghi 3 số T, L, R.

Ví dụ :

| SUMMAX.INP | SUMMAX.OUT |
|------------|------------|
| 5 | 8 2 5 |
| -1 | |
| 5 | |
| -3 | |
| 2 | |
| 4 | |

22. (*) Cho dãy (a_N), hãy tìm đoạn con tăng dần có tổng lớn nhất
 Dữ liệu : được cho trong tập tin AMAX.INP
- Dòng 1 chứa số nguyên N ($0 < N \leq 30000$).
 - N dòng tiếp theo, mỗi dòng chứa một số là các số của dãy đã cho theo đúng thứ tự. Giá trị tuyệt đối của mỗi số không vượt quá 30000.
- Kết quả tìm được ghi vào tập tin văn bản AMAX.OUT gồm hai dòng:
- Dòng 1 ghi tổng của dãy con.
 - Dòng 2 ghi mảng con tăng dần có tổng lớn nhất.
23. Viết chương trình nhập lý lịch một nhân viên vào danh sách các nhân viên.
 Khi không nhập nữa bấm phím **Esc** và ghi vào tập tin NHANVIEN.DAT sau đó :
- Đọc từ tập tin NHANVIEN.DAT vừa tạo và in danh sách các nhân viên lên màn hình.
 - Tìm và in lý lịch một nhân viên bằng các nhập và họ tên hoặc mã số nhân viên.
24. (***) Để lắp ráp một máy vi tính hoàn chỉnh cần phải có tối thiểu 10 linh kiện loại A và có thể lắp bổ sung thêm vào khoảng tối đa 8 linh kiện loại B. Tại một cửa hàng vi tính cần quản lý bán hàng các loại linh kiện tại cửa hàng. Thông tin về một loại linh kiện gồm có: Tên linh kiện, quy cách , loại, đơn giá loại 1 (chất lượng tốt – số nguyên), đơn giá loại 2 (chất lượng thường – số nguyên). Viết chương trình thực hiện những công việc sau :

- Nhập vào thông tin của các loại linh kiện có ở cửa hàng. Xuất danh sách các linh kiện đã nhập theo thứ tự tăng dần của loại linh kiện và tên linh kiện. Cho biết đã có đủ 10 linh kiện loại A cần thiết để lắp ráp máy tính hay chưa?
- Với giả định là cửa hàng đã có đủ 10 linh kiện loại A để lắp ráp máy. Nhập vào một số tiền để lắp ráp một máy tính. Có thể lắp được một máy tính hoàn chỉnh với các linh kiện toàn bộ theo đơn giá loại 1 hay đơn giá loại 2 hay không? Nếu số tiền trong khoảng giữa thì hãy tìm một phương án gồm những linh kiện theo đơn giá 1 và linh kiện theo đơn giá 2 để lắp?
- Tất cả dữ liệu phải lưu ở tập tin.

III. KẾT LUẬN

- ❖ Mục đích của kiểu dữ liệu tập tin cho phép chúng ta lưu lại những thông tin cần thiết tương đối lớn: những dữ liệu đầu vào, những kết quả của chương trình hoặc những dữ liệu dùng để test chương trình, ...
- ❖ Khi thao tác trên tập tin phải thông qua 4 bước: ***Khai báo con trỏ trỏ đến tập tin, Mở tập tin, Xử lý trên tập tin và cuối cùng là Đóng tập tin.***
- ❖ Lưu ý khi mở tập tin để ghi thì phải cẩn thận với thao tác tạo mới hay chỉnh sửa nội dung tập tin, di chuyển con trỏ hợp lý để tránh mất thông tin.
- ❖ Sử dụng hàm thao tác trên tập tin phải dùng **đúng loại hàm** cho tập tin kiểu nhị phân hay kiểu văn bản.

Giới thiệu phương pháp lập trình theo kỹ thuật đệ quy, phân loại, cách hoạt động và cách cài đặt các hàm đệ quy.

I. TÓM TẮT LÝ THUYẾT

I.1. Khái niệm

Một hàm được gọi có tính đệ qui nếu trong thân của hàm đó có lệnh gọi lại chính nó một cách tường minh hay tiềm ẩn.

I.2. Phân loại đệ qui

- Đệ qui tuyến tính.
- Đệ qui nhị phân.
- Đệ qui phi tuyến.
- Đệ qui hỗ tương.

a. **Đệ qui tuyến tính**

Trong thân hàm có duy nhất một lời gọi hàm gọi lại chính nó một cách tường minh.

```
<Kiểu dữ liệu hàm> TenHam (<danh sách tham số>)
{
    if (điều kiện dừng)
    {
        ...
        //Trả về giá trị hay kết thúc công việc
    }
    //Thực hiện một số công việc (nếu có)
    ... TenHam (<danh sách tham số>);
    //Thực hiện một số công việc (nếu có)
}
```

Ví dụ 1: Tính $S(n) = 1 + 2 + 3 + \dots + n$

Trước khi cài đặt hàm đệ qui ta xác định:

- Điều kiện dừng: $S(0) = 0$.

- Quy tắc (công thức) tính: $S(n) = S(n-1) + n$.

Ta cài đặt hàm đệ qui như sau:

```

long TongS (int n)
{
    if(n==0)
        return 0;
    return ( TongS(n-1) + n );
}

```

Ví dụ 2: Tính $P(n) = n!$

Trước khi cài đặt hàm đệ qui ta xác định:

- Điều kiện dừng: $P(0) = 0! = 1$.
- Quy tắc (công thức) tính: $P(n) = P(n-1) * n$.

Ta cài đặt hàm đệ qui như sau:

```

long GiaiThua (int n)
{
    if(n==0)
        return 1;
    return ( GiaiThua(n-1) * n );
}

```

b. Đệ qui nhị phân

Trong thân của hàm có hai lời gọi hàm gọi lại chính nó một cách tường minh.

```

<Kiểu dữ liệu hàm> TenHam (<danh sách tham số>)
{
    if (điều kiện dừng)
    {
        ...
        //Trả về giá trị hay kết thúc công việc
    }
    //Thực hiện một số công việc (nếu có)
    ... TenHam (<danh sách tham số>); //Giải quyết vấn đề nhỏ hơn
    //Thực hiện một số công việc (nếu có)
    ... TenHam (<danh sách tham số>); //Giải quyết vấn đề còn lại
    //Thực hiện một số công việc (nếu có)
}

```

Ví dụ 1: Tính số hạng thứ n của dãy Fibonacci được định nghĩa như sau:

$$f_1 = f_0 = 1 ;$$

$$f_n = f_{n-1} + f_{n-2} ; \quad (n > 1)$$

Trước khi cài đặt hàm đệ qui ta xác định:

- Điều kiện dừng: $f(0) = f(1) = 1$.

Ta cài đặt hàm đệ qui như sau:

```

long Fibonacci (int n)
{
    if(n==0 || n==1)
        return 1;
    return Fibonacci(n-1) + Fibonacci(n-2);
}

```

Ví dụ 2: Cho dãy số nguyên a gồm n phần tử có thứ tự tăng dần. Tìm phần tử có giá trị x có xuất hiện trong mảng không?

Trước khi cài đặt hàm đệ qui ta xác định:

- Điều kiện dừng: Tìm thấy x hoặc xét hết các phần tử.

- Giải thuật:

Do dãy số đã có thứ tự tăng nên ta có thể áp dụng cách tìm kiếm theo phương pháp nhị phân. Ý tưởng của phương pháp này là tại mỗi bước ta tiến hành so sánh x với phần tử nằm ở vị trí giữa của dãy để thu hẹp phạm vi tìm.

Goi: l: biên trái của dãy (ban đầu $l=0$).

r: biên phải của dãy (ban đầu $r = n-1$).

m: vị trí ở giữa ($m = (l+r)/2$).

| | | |
|-------------------|-------------|---------------|
| <i>l</i> | <i>m</i> | <i>R</i> |
| ↓ | ↓ | ↓ |
| <i>a[0]</i> | <i>a[1]</i> | ... |
| <i>a[(l+r)/2]</i> | ... | <i>a[n-2]</i> |
| <i>a[n-1]</i> | | |

Thu hẹp dựa vào giá trị của phần tử ở giữa, có hai trường hợp:

i. Nếu x lớn hơn phần tử ở giữa thì x chỉ có thể xuất hiện ở bên phải vị trí này. (từ $m+1$ đến r).

ii. Ngược lại nếu x nhỏ hơn phần tử ở giữa thì x chỉ có thể xuất hiện ở bên trái vị trí này. (từ l đến $m-1$).

Quá trình này thực hiện cho đến khi gặp phần tử có giá trị x, hoặc đã xét hết các phần tử.

Ta cài đặt hàm đệ qui như sau:

```

int TimNhiPhan(int a[], int l, int r, int x)
{
    int m = (l+r)/2;
    if(l>r)
        return -1; // Không có phần tử x
    if(a[m]>x) return TimNhiPhan(a, l, m-1, x);
    if(a[m]<x) return TimNhiPhan(a, m+1, r, x);
    return m; // Trả về vị trí tìm thấy
}

```

}

Ví dụ 3: Bài toán tháp Hà Nội:

Bước 1: Di chuyển $n - 1$ đĩa nhỏ hơn từ cọc A sang cọc B.

Bước 2: Di chuyển đĩa còn lại từ cọc A sang cọc C.

Bước 3: Di chuyển $n - 1$ đĩa nhỏ hơn từ cọc B sang cọc C.

Ta cài đặt hàm đệ qui như sau:

```
void ThapHaNoi (int n, char A, char B, char C)
{
    if (n == 1)
        printf("Di chuyen dia tren cung tu %d den %d\n", A, C);
    else
    {
        ThapHaNoi(n-1, A, C, B);
        ThapHaNoi(1, A, B, C);
        ThapHaNoi(n-1, B, A, C);
    }
}
```

c. Đệ qui phi tuyến

Trong thân của hàm có lời gọi hàm gọi lại chính nó được đặt bên trong vòng lặp.

```
<Kiểu dữ liệu hàm> TenHam (<danh sách tham số>)
{
    for (int i = 1; i<=n; i++)
    {
        //Thực hiện một số công việc (nếu có)
        if (điều kiện dừng)
        {
            ...
            //Trả về giá trị hay kết thúc công việc
        }
        else
        {
            //Thực hiện một số công việc (nếu có)
            TenHam (<danh sách tham số>);
        }
    }
}
```

Ví dụ: Tính số hạng thứ n của dãy $\{X_n\}$ được định nghĩa như sau:

$$X_0 = 1 ;$$

$$X_n = n^2 X_0 + (n-1)^2 X_1 + \dots + 1^2 X_{n-1} ; \quad (n \geq 1)$$

Trước khi cài đặt hàm đệ qui ta xác định:

- Điều kiện dừng: $X(0) = 1$.

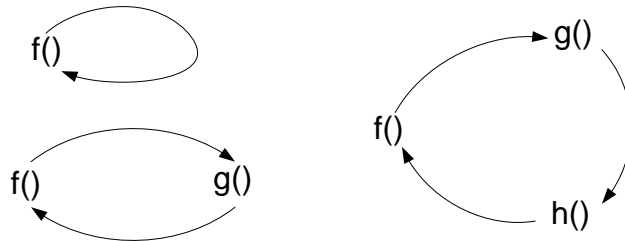
Ta cài đặt hàm đệ qui như sau:

```

long TinhXn (int n)
{
    if(n==0)
        return 1;
    long s = 0;
    for (int i=1; i<=n; i++)
        s = s + i * i * TinhXn(n-i);
    return s;
}
    
```

d. Đệ qui hỗ tương

Trong thân của hàm này có lời gọi hàm đến hàm kia và trong thân của hàm kia có lời gọi hàm tới hàm này.



```

<Kiểu dữ liệu hàm> TenHam2 (<danh sách tham số>);
<Kiểu dữ liệu hàm> TenHam1 (<danh sách tham số>)
{
    //Thực hiện một số công việc (nếu có)
    ...TenHam2 (<danh sách tham số>);
    //Thực hiện một số công việc (nếu có)
}

<Kiểu dữ liệu hàm> TenHam2 (<danh sách tham số>)
{
    //Thực hiện một số công việc (nếu có)
    ...TenHam1 (<danh sách tham số>);
    //Thực hiện một số công việc (nếu có)
}
    
```

Ví dụ: Tính số hạng thứ n của hai dãy $\{X_n\}$, $\{Y_n\}$ được định nghĩa như sau:

$$\begin{aligned}
 X_0 &= Y_0 = 1 ; \\
 X_n &= X_{n-1} + Y_{n-1}; & (n > 0) \\
 Y_n &= n^2 X_{n-1} + Y_{n-1}; & (n > 0)
 \end{aligned}$$

Trước khi cài đặt hàm đệ qui ta xác định:

- Điều kiện dừng: $X(0) = Y(0) = 1$.

Ta cài đặt hàm đệ qui như sau:

```

long TinhXn(int n);
long TinhYn(int n)
{
    if(n==0)
        return 1;
    return TinhXn(n-1) + TinhYn(n-1);
}

long TinhXn (int n)
{
    if(n==0)
        return 1;
    return n*n*TinhXn(n-1) + TinhYn(n-1);
}

```

I.3. Tìm hiểu cách hoạt động của hàm đệ qui

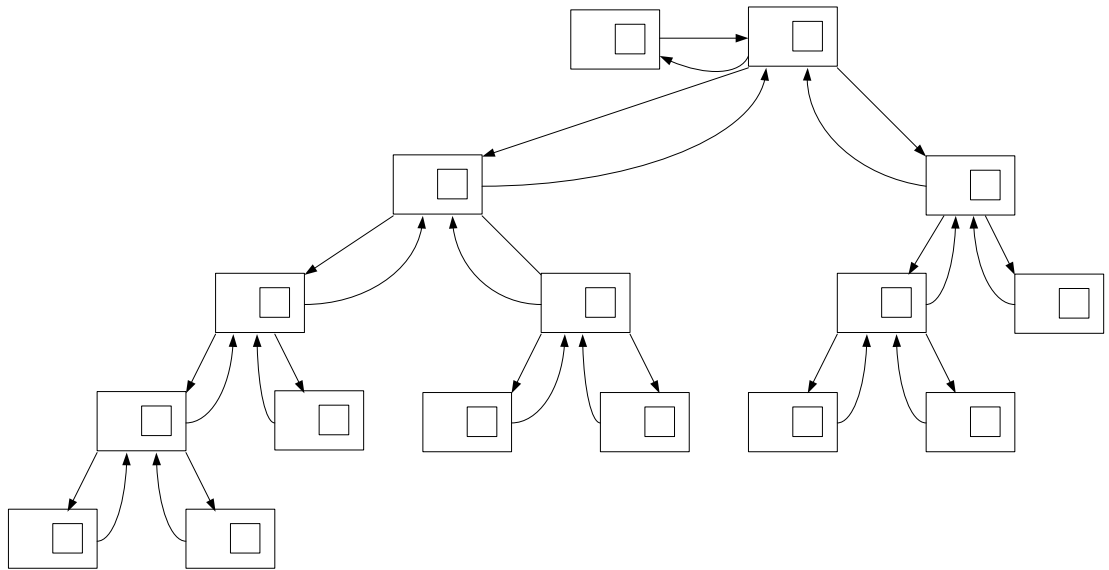
Phục vụ cho công việc kiểm chứng kết quả thực thi của chương trình bằng tay.

Ví dụ 1: Lấy lại ví dụ tính $P(n) = n!$ bằng phương pháp đệ qui như đã mô tả cài đặt ở trên với $n = 5$



Lệnh gọi khởi đầu trong hàm main(), truyền đến hàm GiaiThua(). Ở đó, giá trị của tham số n là 5, do đó nó gọi GiaiThua(4), truyền 4 đến hàm GiaiThua(). Ở đó giá trị của tham số n là 4, do đó nó gọi GiaiThua(3), truyền 3 đến hàm GiaiThua(). Tiến trình này tiếp tục (đệ quy) đến khi gọi GiaiThua(1) được thực hiện từ bên trong lệnh gọi GiaiThua(2). Ở đó, giá trị của tham số n là 1, do đó nó trả về giá trị 1, mà không thực hiện thêm bất kì lệnh gọi nào. Sau đó lần ngược về lệnh gọi GiaiThua(2) trả $2*1=2$ trở về lệnh gọi GiaiThua(3). Sau đó lệnh gọi GiaiThua(3) trả $3*2=6$ trở về lệnh gọi GiaiThua(4). Sau đó lệnh gọi GiaiThua(4) trả $4*6=24$ trở về lệnh gọi GiaiThua(5). Sau cùng, lệnh gọi GiaiThua(5) trả về giá trị 120 cho hàm main().

Ví dụ 2: Lấy lại ví dụ tính số hạng thứ n của dãy Fibonacci như đã mô tả cài đặt ở trên với $n = 5$, quá trình thực hiện tương tự như trong ví dụ trước, ta có sơ đồ sau:



I.4. Ví dụ

Viết chương trình nhập vào mảng một chiều số nguyên a, xuất ra màn hình và tính tổng các phần tử có giá trị chẵn bằng phương pháp đệ qui.

```

#define MAX 100
void Nhap(int a[], int n)
{
    if(n==0)
        return;
    Nhap(a, n-1);
    printf("\nNhap phan tu thu %d: ", n);
    scanf("%d", &a[n-1]);
}

void Xuat(int a[], int n)
{
    if(n==0)
        return;
    Xuat(a, n-1);
    printf("%d\t", a[n-1]);
}

long TongChan(int a[], int n)
{
    if(n==0)
        return 0;
    int s = TongChan(a, n-1);
    if(a[n-1]%2==0)
        s+=a[n-1];
    return s;
}
    
```

```

void main()
{
    int a[MAX], n;
    long s;

    printf("\nNhap so phan tu cua mang: ");
    scanf("%d", &n);
    Nhap(a, n);
    Xuat(a, n);
    s=TongChan(a, n);
    printf("\nTong cac so chan trong mang la: %ld", s);
    getch();
}

```

II. BÀI TẬP

Viết hàm đệ qui thực hiện các yêu cầu sau:

II.1. Bài tập cơ bản

1. Cài đặt lại những bài tập ở chương mảng một chiều.
2. Tìm chữ số có giá trị lớn nhất của số nguyên dương n .
3. Hãy xây dựng một dãy gồm N số có giá trị từ 1 đến K cho trước, sau cho không có hai dãy con liên tiếp đứng kề nhau.

Ví dụ: $N = 6$

$K = 3$

Kết quả: 121312

4. Tìm ước số chung lớn nhất của hai số nguyên dương a và b .
5. Tìm chữ số đầu tiên của số nguyên dương n .
6. Tìm dãy nhị phân dài nhất sao cho trên dãy này không có hai bộ k bất kỳ trùng nhau. Bộ k là dãy con có k số liên tiếp nhau trên dãy tìm được.

Ví dụ: $k = 3$

Kết quả: 000 101 110 0

7. Tính $P(n) = 1.3.5 \dots (2n+1)$, với $n \geq 0$
8. Tính $S(n) = 1 + 3 + 5 + \dots + (2 \times n + 1)$, với $n \geq 0$
9. Tính $S(n) = 1 - 2 + 3 - 4 + \dots + (-1)^{n+1} n$, với $n > 0$
10. Tính $S(n) = 1 + 1.2 + 1.2.3 + \dots + 1.2.3 \dots n$, với $n > 0$
11. Tính $S(n) = 1^2 + 2^2 + 3^2 + \dots + n^2$, với $n > 0$

12. Tính $S(n) = 1 + \frac{1}{2} + \frac{1}{3} + \dots + \frac{1}{n}$, với $n > 0$
13. Tính $S(n) = 1 + \frac{1}{1+2} + \frac{1}{1+2+3} + \dots + \frac{1}{1+2+3+\dots+n}$, với $n > 0$
14. Tính $P(x, y) = x^y$.
15. Tính $S(n) = 1 + (1+2) + (1+2+3) + \dots + (1+2+3+\dots+n)$, với $n > 0$

II.2. Bài tập luyện tập và nâng cao

16. Cho số nguyên dương n . In ra biểu diễn nhị phân của n .
17. (*) Cài đặt và minh họa bài toán tháp Hà Nội.
18. (**) Cài đặt bài toán mã đi tuần.
19. (**) Cài đặt bài toán tám hậu.
20. (*) Tính $S(n) = \sqrt{n + \sqrt{(n-1) + \sqrt{(n-2) + \dots + \sqrt{1}}}}$, với $n > 0$
21. (*) Tính $S(n) = \sqrt{1 + \sqrt{2 + \sqrt{3 + \dots + \sqrt{n}}}}$, với $n > 0$
22. (*) Tính $S(n) = \frac{1}{1 + \frac{1}{1 + \frac{1}{1 + \frac{1}{\ddots \frac{1}{1 + \frac{1}{1+1}}}}}}$ có n dấu phân số.

III. KẾT LUẬN

- ❖ Đệ qui cung cấp cho ta cơ chế giải quyết các bài toán phức tạp một cách đơn giản hơn.
- ❖ Xây dựng hàm đệ qui thông qua việc **xác định điều kiện dừng** và **bước thực hiện tiếp theo**.
- ❖ Chỉ nên cài đặt bằng phương pháp đệ qui khi không còn cách giải quyết bằng cách lặp thông thường.

CHƯƠNG 10 LẬP TRÌNH THEO PHƯƠNG PHÁP PROJECT

I. MỤC TIÊU

Chia một chương trình lớn thành các tập tin nhỏ hơn, mỗi tập tin chứa các khai báo nguyên mẫu hàm, cài đặt các hàm và dữ liệu thực hiện một số chức năng nhất định.

Việc phân chia này giúp quá trình lập trình:

- ❖ Dễ kiểm soát các lệnh và kiểm lỗi.
- ❖ Tránh được giới hạn kích thước tập tin quá lớn của ngôn ngữ lập trình.

II. PHƯƠNG PHÁP

II.1. Tạo một project mới

Bước 1: Tạo thư mục sẽ chứa toàn bộ chương trình sẽ được cài đặt.

Bước 2: Khởi động Borland C++ 3.1.

Bước 3: Thay đổi đường dẫn đến thư mục vừa tạo.

Vào menu **File\Change Dir ...** sau đó chọn đường dẫn thư mục và chọn OK.

Bước 4: Tạo Project.

Vào menu **Project\Open Project** sau đó đặt tên cho project tương ứng, chọn OK.

(Lưu ý: Xem đường dẫn của file Project có nằm đúng thư mục vừa tạo ở bước 1 hay không. Nếu cần có thể chỉnh sửa lại đường dẫn).

Bước 5: Thêm file vào Project.

Chọn menu **Window\Project** sau đó nhấn phím **Insert** hoặc vào menu **Project\Add Item ...** đặt tên file và chọn OK, muốn loại file khỏi project thì chọn **Project>Delete Item** (Hoặc khi đang trong cửa sổ project vừa tạo nhấn phím **insert** để thêm file, muốn xóa chọn file rồi nhấn **delete**).

Lưu ý: Chỉ Insert các file chứa cài đặt lớp và hàm main (.cpp), không insert file header do người dùng định nghĩa (*.h).*

II.2. Mở project có sẵn

Bước 1: Đóng project trước (nếu có).

Vào menu **Project\Close Project**.

Bước 2: Mở project.

Vào menu **Project\Open Project** chọn đường dẫn đến file project cần thực hiện, chọn OK.

Bước 3: Hiệu chỉnh đường dẫn thư viện của BC++ 3.1.

Việc tạo project ở các máy với thông số cài đặt BC++3.1 khác nhau sẽ dẫn đến đường dẫn thư viện hàm của các máy cũng khác nhau, do vậy khi biên dịch sẽ gặp lỗi về thư viện hàm trong BC++3.1.

Vào menu **Options\Directories...** sau đó hiệu chỉnh lại đường dẫn đến thư mục chứa thư viện hàm trong các ô **Include** và **Library** cho đúng với đường dẫn cài BC++3.1 (*Đường dẫn đến thư mục INCLUDE và thư mục BIN của BC++3.1 trên máy đang sử dụng*).

II.3. Một số lưu ý

Nên chia từng file theo từng nhóm hàm. Mỗi một project phải có tối thiểu **3 file** như sau:

- **File header (*.h):** Tạo thư viện tự định nghĩa. Chứa các khai báo nguyên mẫu hàm, kiểu dữ liệu, ...
- **File cài đặt hàm (*.cpp):** Chứa các cài đặt hàm theo nhóm. Nếu có sử dụng thư viện tự định nghĩa thì phải include file chứa thư viện đó vào.
- **File chứa hàm main() (m*.cpp):** Chứa hàm chính (hàm **main()**).

⌘ *Khi cài đặt hay chỉnh sửa một hàm nào đó trước hết phải xem xét hàm đó thuộc nhóm hàm nào và sau đó mở file của nhóm tương ứng để hiệu chỉnh.*

II.4. Ví dụ minh họa

Viết chương trình nhập thông tin của học sinh gồm: họ tên học sinh, điểm văn và toán, xuất thông tin và tính điểm trung bình cho học sinh đó.

Ta chia chức năng chương trình theo các nhóm chức năng để dễ quản lý, gồm các file sau:

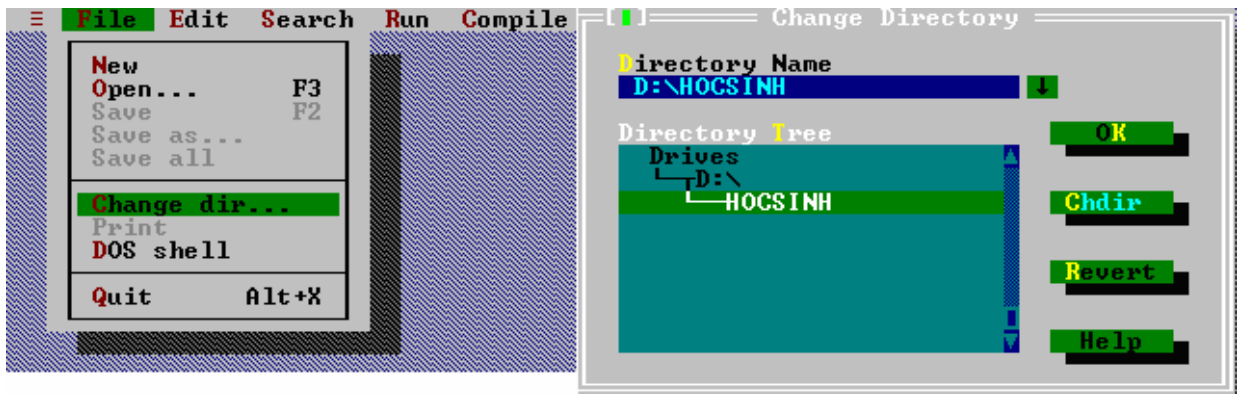
- ❖ *File **hocsinh.h**: Chứa các khai báo biến và nguyên mẫu hàm.*
- ❖ *File **mhocsinh.cpp**: Chứa hàm main().*
- ❖ *File **xuat.cpp**: Chứa các thao tác xuất thông tin học sinh, ...*
- ❖ *File **nhap.cpp**: Chứa các thao tác nhập thông tin học sinh, ...*
- ❖ *File **tinhtoan.cpp**: Chứa các thao tác tính điểm trung bình, ...*

Bước 1: Tạo thư mục **HOCSINH** sẽ chứa toàn bộ các file của chương trình sẽ được cài đặt (Ví dụ tạo ở ổ đĩa D:).

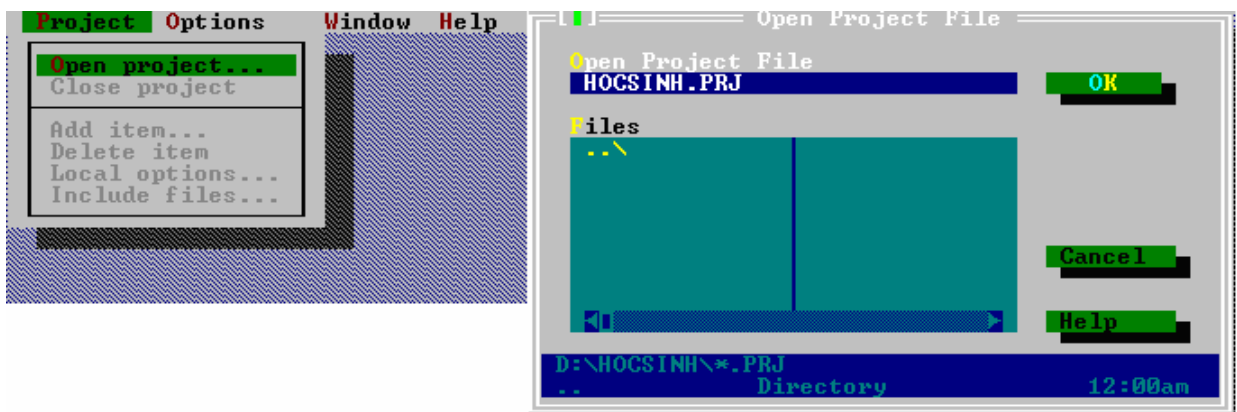
Bước 2: Khởi động Borland C++ 3.1.

Bước 3: Thay đổi đường dẫn đến thư mục **HOCSINH** vừa tạo.

❖ Chọn thư mục **HOCSINH**



Bước 4: Tạo Project: Đặt tên file project là **hocsinh**



❖ Cài đặt file **hocsinh.h** trong thư mục **HOCSINH**.

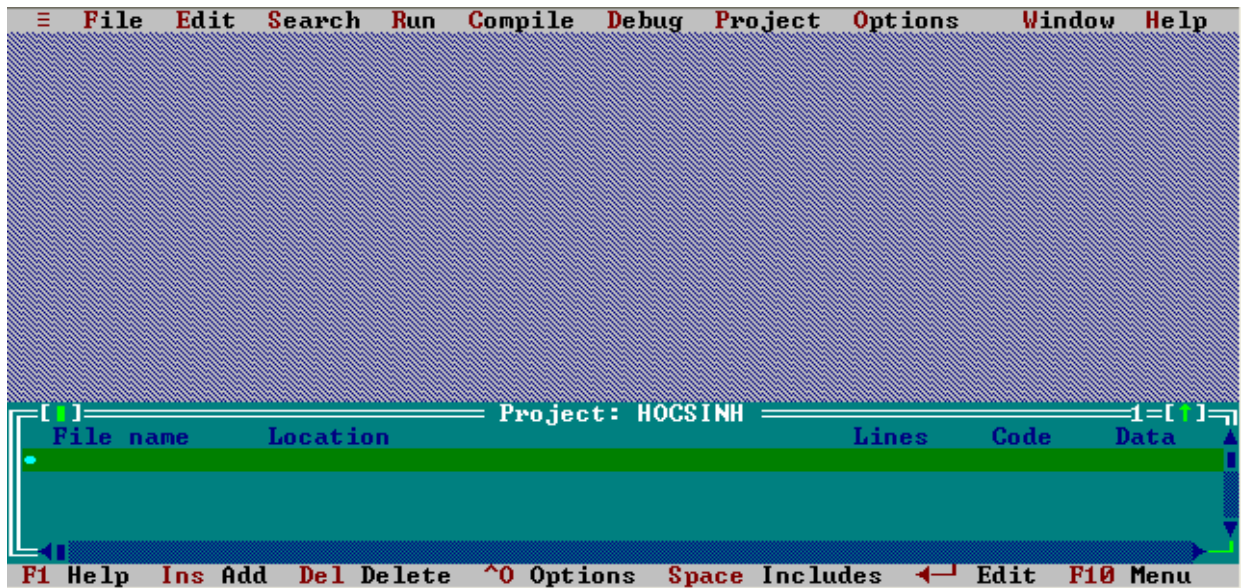
```
#ifndef _HOCSINH_H
#define _HOCSINH_H
```

```
typedef struct HOCSINH
{
    char hoten[30];
    int toan, van;
};
```

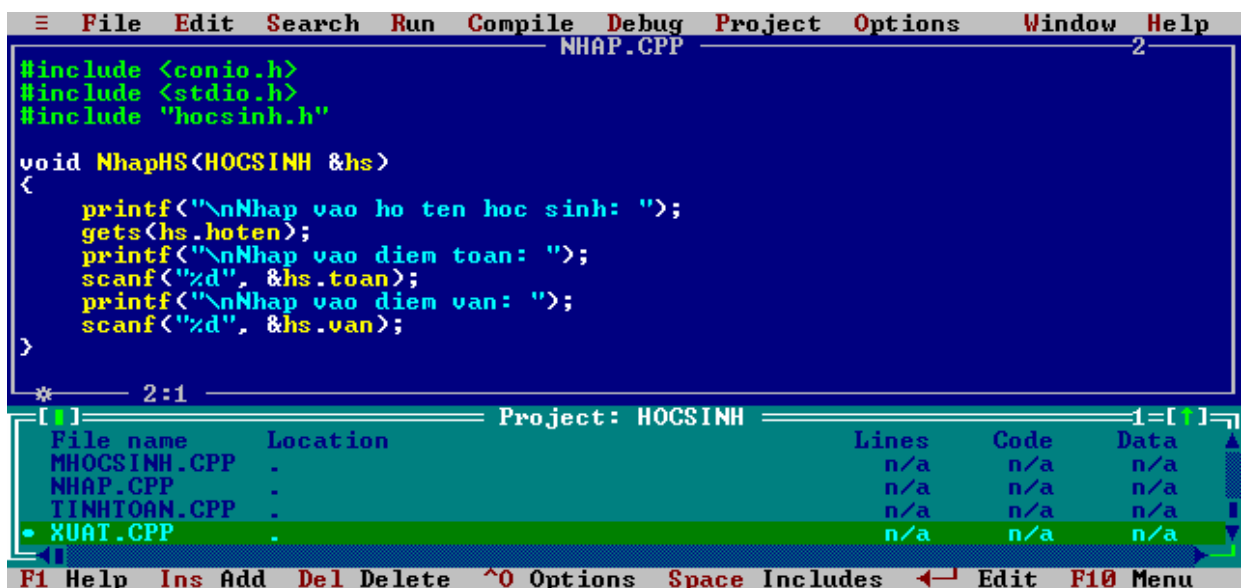
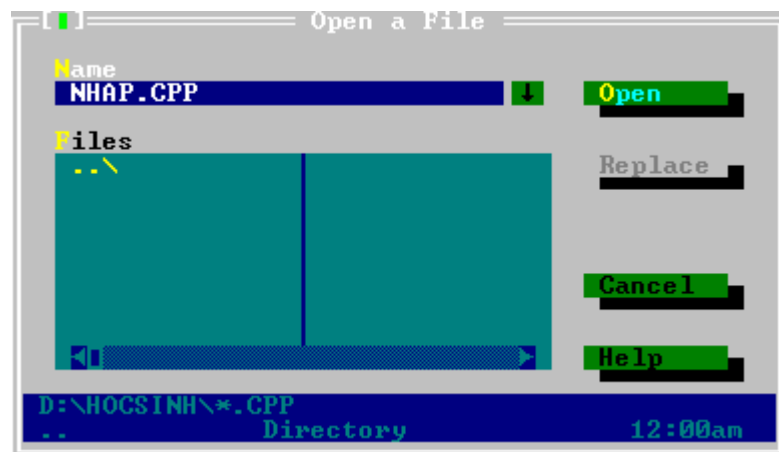
```
void NhapHS(HOCSINH &hs);
float TinhDTB(HOCSINH hs);
void XuatHS(HOCSINH hs);
```

```
#endif
```


Bước 5: Thêm file vào Project.



- ❖ Nhấn F3, đặt tên file mới là **nhap.cpp** và viết hàm nhập. Tương tự cho những file: **xuat.cpp**, **tinhtoan.cpp** và file **mhocsinh.cpp**.



- ❖ Nội dung file **nhap.cpp**.

```
#include <conio.h>
#include <stdio.h>

// Sử dụng kiểu dữ liệu HOCSINH và khai báo nguyên mẫu hàm
#include "hocsinh.h"

void NhapHS(HOCSINH &hs)
{
    printf("\nNhập vào họ tên học sinh: ");
    gets(hs.hoten);
    printf("\nNhập vào điểm toán: ");
    scanf("%d", &hs.toan);
    printf("\nNhập vào điểm văn: ");
    scanf("%d", &hs.van);
}
```

- ❖ Nội dung file **xuat.cpp**.

```
#include <conio.h>
#include <stdio.h>

// Sử dụng kiểu dữ liệu HOCSINH và khai báo nguyên mẫu hàm
#include "hocsinh.h"

void XuatHS(HOCSINH hs)
{
    printf("\nHọ tên học sinh: %s", hs.hoten);
    printf("\nĐiểm toán: %d \nĐiểm văn: %d", hs.toan, hs.van);
    printf("\nĐiểm trung bình: %.2f", TinhDTB(hs));
}
```

- ❖ Nội dung file **tinhtoan.cpp**.

```
// Sử dụng kiểu dữ liệu HOCSINH và khai báo nguyên mẫu hàm
#include "hocsinh.h"

float TinhDTB(HOCSINH hs)
{
    return (hs.toan + hs.van)/2.0;
}
```

- ❖ Nội dung file **mhocsinh.cpp**.

```
#include <conio.h>
#include <stdio.h>

// Sử dụng kiểu dữ liệu HOCSINH và khai báo nguyên mẫu hàm
#include "hocsinh.h"
```

```
void main()  
{  
    clrscr();  
    HOCSINH hs;  
  
    NhapHS(hs);  
    printf("\nKet qua:\n");  
    XuatHS(hs);  
  
    getch();  
}
```

- ❖ Nhấn F9 để biên dịch và kiểm lỗi.
- ❖ Nhấn Ctrl + F9 để thực thi chương trình.

Ví dụ kết quả chạy chương trình

Nhap vào họ tên học sinh: Nguyen Van A
Nhap vào điểm toán: 6
Nhap vào điểm văn: 5

Ket qua:

Ho ten học sinh: Nguyen Van A
Diem toán: 6
Diem văn: 5
Diem trung bình: 5.50

III. BÀI TẬP

Cài đặt các bài tập ở chương mảng cấu trúc bằng phương pháp tạo **project**.

PHỤ LỤC 1 **ĐỀ THI MẪU**

ĐỀ SỐ 01

Thời gian: 120 phút
(Không tham khảo tài liệu)
⊕ ⊕ ⊕ ⊕

Câu 1: Viết chương trình tính tổng: $S(n) = 1! + 2! + \dots + n!$

Câu 2: Viết chương trình thực hiện các yêu cầu sau:

- Nhập mảng một chiều các số nguyên.
- Đếm số lượng giá trị chẵn âm trong mảng.
- Tìm số lẻ cuối cùng trong mảng.

Câu 3: Cho ma trận các số thực. Viết hàm tìm giá trị trong ma trận xa giá trị x nhất.

float **xanhat**(float a[][100], int m, int n, float x);

Câu 4: Hãy khai báo kiểu dữ liệu biểu diễn khái niệm điểm trong mặt phẳng Oxy (DIEM).

- Viết hàm nhập tọa độ điểm.
void **nhap**(DIEM &P);
- Viết hàm xuất tọa độ điểm.
void **xuat**(DIEM P);
- Viết hàm tính khoảng cách giữa 2 điểm.

float **khoangcach**(DIEM P, DIEM Q);

ĐỀ SỐ 02

Thời gian: 120 phút
(Không tham khảo tài liệu)
⊕ ⊕ ⊕ ⊕

Câu 1: Viết chương trình tính tổng: $S(x, n) = x + x^2 + \dots + x^n$

Câu 2: Viết chương trình thực hiện các yêu cầu sau:

- Nhập mảng một chiều các số nguyên.

- b. Đếm số lượng giá trị lẻ dương trong mảng.
- c. Tìm số chẵn cuối cùng trong mảng.

Câu 3: Cho ma trận các số thực. Viết hàm tìm giá trị trong ma trận gần giá trị x nhất.

float **gannhat**(float a[][100], int m, int n, float x);

Câu 4: Hãy khai báo kiểu dữ liệu biểu diễn khái niệm phân số (**PHANSO**)

- a. Viết hàm nhập phân số.

void **nhap**(PHANSO &x);

- b. Viết hàm xuất phân số.

void **xuat**(PHANSO x);

- c. Viết hàm tính tổng hai phân số.

PHANSO **tong**(PHANSO x, PHANSO y);

ĐỀ SỐ 03

Thời gian: 120 phút

(Không tham khảo tài liệu)

⊕ ⊕ ⊕ ⊕

Câu 1:

$$S_n = \frac{1}{2} + \frac{3}{4} + \frac{5}{6} + \dots + \frac{2n+1}{2n+2} \quad \text{Với } n \text{ nguyên dương } (n > 0)$$

1. Vẽ lưu đồ thuật toán (Flowchart) tính tổng trên.
2. Viết hàm tính tổng trên bằng **phương pháp đệ quy**.

Câu 2:

Cho mảng một chiều các số thực A kích thước n ($0 < n \leq 100$). Hãy xây dựng hàm thực hiện các yêu cầu sau:

1. Nhập giá trị các phần tử vào mảng.
2. Tìm và trả về vị trí của phần tử có giá trị **âm đầu tiên** trong mảng. Nếu không có giá trị âm thì trả về -1.
3. Tìm và trả về giá trị phần tử **âm lớn nhất** trong mảng a. Nếu mảng không có phần tử chứa giá trị âm thì trả về 0.

Câu 3:

Cho ma trận vuông các số nguyên A kích thước $n \times n$ ($3 < n < 10$). Hãy xây dựng các hàm cho phép thực hiện các yêu cầu sau:

1. Nhập giá trị các phần tử vào ma trận.
2. Đếm và trả về số lượng các phần tử là **số nguyên tố** trong ma trận.
3. Tính **trung bình cộng** các phần tử trên đường chéo chính.

Câu 4:

Hãy khai báo kiểu dữ liệu để biểu diễn thông tin của một nhân viên (NHANVIEN).

Biết một nhân viên gồm:

- Mã nhân viên (MaNV): Chuỗi tối đa 5 ký tự.
- Tên nhân viên (TenNV): Chuỗi tối đa 30 ký tự.
- Chức vụ (ChucVu): Chuỗi tối đa 20 ký tự
(gồm các chức vụ: “Truong phong”, “Nhan vien”, “Giam doc”, “Pho giam doc”, ...).
- Số năm làm việc (SoNam): Số nguyên 1 byte.
- Hệ số lương (HeSo): Kiểu số thực.

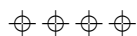
Cho danh sách gồm n ($n > 0$) nhân viên. Viết các hàm sau:

1. Liệt kê các nhân viên có số năm làm việc từ **3 năm trở lên**.
 2. Đếm số nhân viên có chức vụ là **“Truong phong”**.
- Sắp xếp danh sách nhân viên tăng dần theo **hệ số lương** nhân viên.
-

ĐỀ SỐ 04

Thời gian: 120 phút

(Không tham khảo tài liệu)



Câu 1:

$S_n = 1 - 2 + 3 - 4 + \dots + (-1)^{n+1} n$ Với n nguyên dương ($n > 0$)

1. Vẽ lưu đồ thuật toán (Flowchart) tính tổng trên.
2. Viết hàm tính tổng trên bằng **phương pháp đệ quy**.

Câu 2:

Cho mảng một chiều các số nguyên A kích thước n ($0 < n \leq 100$). Hãy xây dựng hàm thực hiện các yêu cầu sau:

1. Nhập giá trị các phần tử vào mảng.

2. Tìm và trả về vị trí của phần tử có giá trị là **số nguyên tố đầu tiên** trong mảng. Nếu không có giá trị là số nguyên tố thì trả về -1.
3. Tìm và trả về giá trị phần tử là **số nguyên tố lớn nhất** trong mảng a. Nếu mảng không có phần tử là số nguyên tố thì trả về 0.

Câu 3:

Cho ma trận vuông các số thực A kích thước $n \times n$ ($3 < n < 10$). Hãy xây dựng các hàm cho phép thực hiện các yêu cầu sau:

1. Nhập giá trị các phần tử vào ma trận.
2. Liệt kê những phần tử tại những dòng lẻ trong ma trận.
3. Tính và trả về giá trị trung bình cộng của những phần tử âm trong ma trận.

Câu 4:

Hãy khai báo kiểu dữ liệu để biểu diễn thông tin của một mặt hàng (**MATHANG**).

Biết một mặt hàng gồm:

- Mã hàng (**MaHang**): Chuỗi tối đa 5 ký tự.
- Tên hàng (**TenHang**): Chuỗi tối đa 30 ký tự.
- Số lượng (**SoLuong**): Số nguyên 2 byte.
- Đơn vị tính (**DonViTinh**): Chuỗi tối đa 5 ký tự.
- Đơn giá (**DonGia**): Kiểu số thực.

Cho danh sách gồm n ($n > 0$) mặt hàng. Viết các hàm sau:

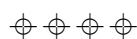
1. Liệt kê các mặt hàng có số lượng **lớn hơn 100**.
2. Tìm và trả về mặt hàng có **thành tiền lớn nhất**
(*thành tiền = số lượng * đơn giá*).

Sắp xếp danh sách các mặt hàng theo thứ tự giảm dần của **đơn giá**.

ĐỀ SỐ 05

Thời gian: 120 phút

(Không tham khảo tài liệu)



Bài 1: Nhập số nguyên n ($0 < n \leq 20$). Viết chương trình xuất n phần tử đầu tiên của hai mảng A và B, cho biết các giá trị được xác định như sau:

$$\begin{cases} A_1 = 1, B_1 = 1 \\ A_i = \sqrt{A_{i-1}^2 + B_{i-1}^2} \\ B_i = 2A_i + \frac{B_{i-1}}{2} \end{cases}$$

Bài 2: Viết chương trình nhập vào ma trận vuông cấp n với n nhập từ bàn phím. Hãy kiểm tra ma trận này có phải là ma trận tam giác dưới hoặc tam giác trên theo đường chéo phụ không?

Ví dụ:

```
1 0 1 0
3 2 0 0
4 0 0 0
5 0 0 0
```

Ma trận tam giác trên

```
0 0 0 4
0 0 5 1
0 1 2 3
1 0 1 0
```

Ma trận tam giác dưới

Bài 3: Mỗi hồ sơ nhân viên gồm:

- họ tên
- năm sinh
- lương cơ bản

Viết chương trình thực hiện các công việc sau:

- Nhập n hồ sơ với n nhập từ bàn phím.
- In ra họ tên và lương cơ bản của nhân viên có lương cơ bản thấp nhất và nhân viên có lương cơ bản cao nhất.
- Ghi xuống file văn bản (với tên file là hoso.txt) danh sách gồm họ tên, lương cơ bản, phụ cấp và thực lãnh của các nhân viên (mỗi nhân viên một dòng) biết rằng:

Phụ cấp = 30% lương cơ bản

Thực lãnh = lương cơ bản + phụ cấp

ĐỀ SỐ 06

Thời gian: 120 phút

(Không tham khảo tài liệu)

⊕ ⊕ ⊕ ⊕

Bài 1: Nhập vào một dãy số thực kết thúc bởi 0 hoặc đã đủ 20 phần tử

- Sắp xếp dãy theo thứ tự tăng dần.

- b. Cho biết dãy có hội tụ không? (Dãy được hội tụ khi có nửa phần tử trở lên nhỏ hơn trung bình cộng của dãy).

Bài 2: Nhập vào ma trận cấp $m \times n$ với m và n nhập từ bàn phím. Hãy kiểm tra xem ma trận có cân bằng theo cột hay không? (Ma trận cân bằng theo cột khi tổng các giá trị của các cột bên trái bằng tổng các giá trị của các cột bên phải, nếu số cột lẻ thì không tính cột giữa).

Ví dụ:

| | | | | |
|---|---|---|---|---|
| 8 | 4 | 5 | 8 | 9 |
| 3 | 5 | 7 | 4 | 6 |
| 4 | 9 | 7 | 5 | 1 |

Tổng bên trái = 33 Tổng bên phải = 33

Kết luận: Ma trận cân bằng theo cột.

Bài 3: Một Album ca nhạc MP3 gồm tối đa 150 ca khúc. Thông tin mỗi ca khúc gồm:

- Tên ca khúc
- Tên nhạc sỹ
- Tên ca sỹ
- Thời gian (tính bằng giây)

Viết chương trình thực hiện các công việc sau:

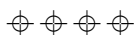
- Nhập n ca khúc với n nhập từ bàn phím.
- Xuất tổng thời gian của các ca khúc (hiển thị theo dạng hh:mm:ss) và cho biết tên ca khúc nào có thời gian dài nhất.

Ghi xuống file văn bản (với tên file là mp3.txt) danh sách gồm tên ca khúc, tên nhạc sỹ, tên ca sỹ và thời gian (hiển thị theo dạng hh:mm:ss), mỗi ca khúc chiếm một dòng.

ĐỀ SỐ 07

Thời gian: 120 phút

(Không tham khảo tài liệu)



Câu 1. Tính số hạng thứ n của hệ thức truy hồi như sau

$$f(0)=1, f(1)=2$$

$$f(n)=3f(n-1)+2f(n-2) \quad (n \geq 2)$$

bằng hai cách

- Dùng đệ qui
- Khử đệ qui, dùng vòng lặp

Câu 2. Xây dựng một cấu trúc có các thành phần sau

- Mã số học sinh
- Họ và tên học sinh
- Điểm Toán
- Điểm Văn
- Điểm trung bình=(Điểm Toán+Điểm Văn)/2

Viết chương trình nhập dữ liệu của n học sinh và lưu vào một tập tin có tên là HOSOHS.DOC (hay mảng 1 chiều có cấu trúc). Sau đó đọc dữ liệu từ tập tin HOSOHS.DOC (hay mảng 1 chiều có cấu trúc), sắp xếp theo thứ tự Điểm trung bình giảm dần và xuất dữ liệu của từng học sinh ra màn hình

Câu 3. Cho n là một số nguyên dương, tính giá trị biểu thức sau bằng cách viết chương trình sử dụng vòng lặp và tối ưu vòng lặp.

$$1 + \frac{1}{2!} + \frac{1}{3!} + \frac{1}{4!} + \dots + \frac{1}{n!} \quad (n = 1, 2, 3, \dots)$$

ĐỀ SỐ 08

Thời gian: 120 phút

(Không tham khảo tài liệu)

⊕ ⊕ ⊕ ⊕

PHẦN I: Chọn câu trả lời đúng nhất (5 điểm)

Đánh dấu chéo vào câu trả lời đúng nhất trong các câu trả lời cho mỗi câu hỏi.

Câu 1. Đoạn chương trình sau sẽ cho giá trị của t:

```
for (t=i=0; (i<10) && (t<100); i++, t += 2*i);
```

- a. 90 b. 100 c. 110 d. 120

Câu 2. Cho dãy gồm 12 phần tử $a_0, a_1, a_2, \dots, a_{11}$ như sau:

-9 -9 -5 -2 0 3 7 7 10 15

Dùng thuật toán tìm nhị phân để tìm vị trí phần tử $x = -9$, vị trí tìm được sẽ là:

- a. -1 b. 0 c. 1 d. 2

Câu 3. Chương trình sau:

```
#include<conio.h>  
#include<stdio.h>  
int a=1, b=2, c=3;  
int A(int &a, int b)  
{  
          a += c + 2;  
          b -= a;  
}
```

```
        return a;
    }
    void main()
    {
        printf(" %d %d", A(b, c), a+c);
    }
```

Sẽ in ra:

- a. 7 5 b. 7 4 c. 7 3 d. 7 2

4. Chương trình sau:

```
#include <conio.h>
#include <stdio.h>
int A(int a, int &b)
{
    a += b + 2;
    b -= a;
    return a;
}
void main()
{
    int x = 5;
    printf(" %d %d", A(A(3, x), x), x);
}
```

Sẽ in ra:

- a. 5 5 b. 5 7 c. 7 7 d. 7 5

5. Đoạn chương trình dưới đây khi thực thi sẽ:

```
char buf1[100], buf2[100], *strptr1, *strptr2;
strcpy(buf1, "abcdefghijklmnopqrstuvwxyz");
strcpy(buf2, "Hello");
strptr1 = buf1 + 6;
strcpy(strptr1, buf2);
strptr2 = (strptr1 + 4);
strncpy(strptr2, buf2, 4);
printf("%s\n", buf1);
```

Sẽ in ra màn hình:

- a. abcdefHellHelloworldpqrstuvwxyz b. ghijklmnHellotuvwxyz
c. abcdefghijklmnopqrstuvwxyz d. abcdefHelloImnopqrstuvwxyz

PHẦN II: Lập trình (5 điểm)

Câu 1. Hãy viết hàm kiểm tra một số nguyên không n có phải là số nguyên tố hay không, hàm thực hiện sẽ trả về: 1 nếu n là số nguyên tố, 0 nếu n không là số nguyên tố

```
int LaSNT(unsigned int n);
```

Câu 2. Hãy viết hàm tìm tổng các số nguyên tố nằm trong mảng một chiều a có n phần tử (unsigned int a[100], int n).

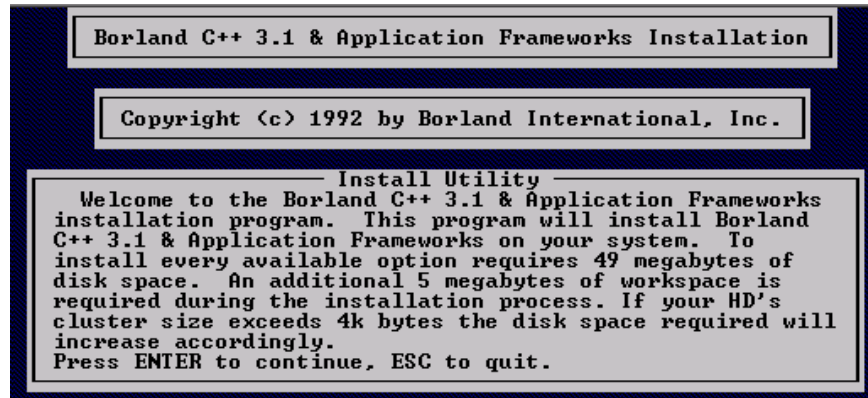
Câu 3. Viết hàm xác định vị trí của số nguyên tố lớn nhất trên mảng a có n phần tử (unsigned int a[100], int n)

(Lưu ý: Có thể làm các câu 2 và 3 mà không cần làm câu 1).

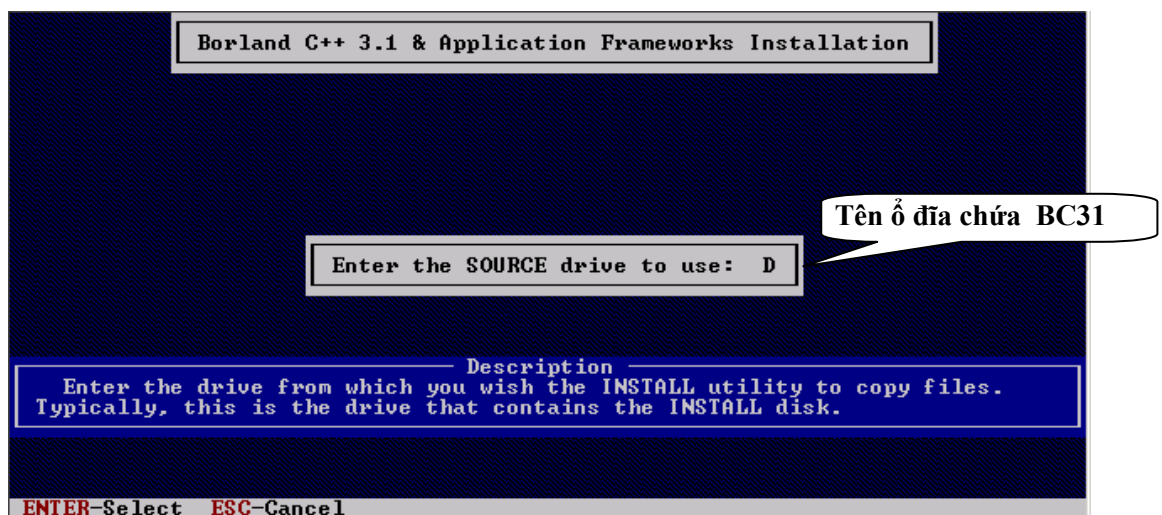
PHỤ LỤC 2 HƯỚNG DẪN VIẾT CHƯƠNG TRÌNH TRÊN MÔI TRƯỜNG BORLAND C++ 3.1 (BC31)

I. CÀI ĐẶT BC3.1

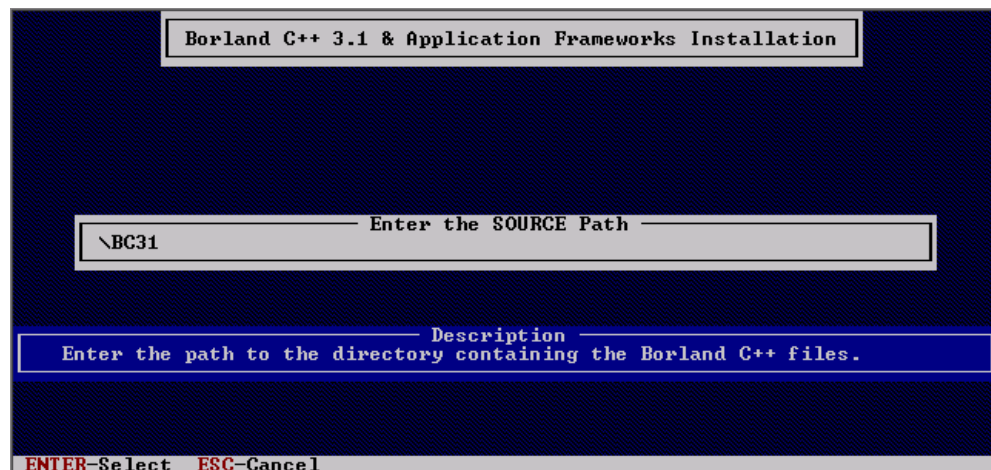
- ❖ Vào thư mục BC3.1 trên đĩa CD chạy file install.exe.



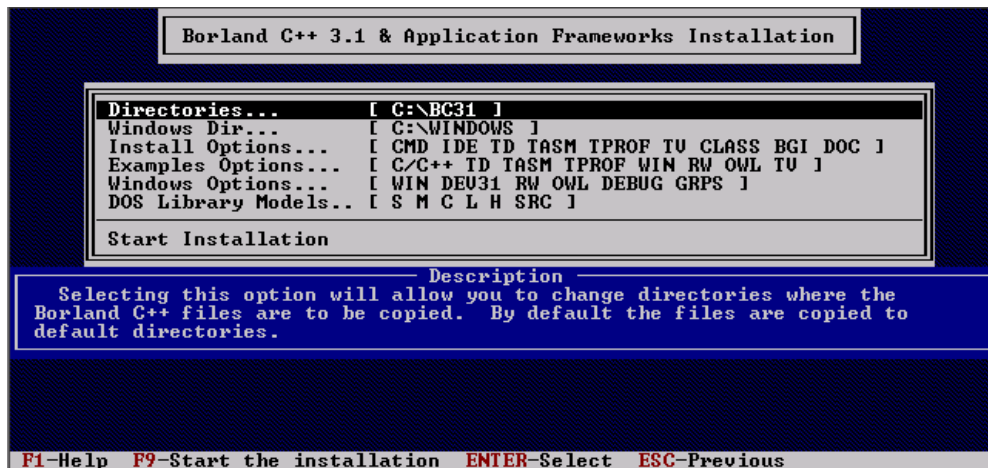
- ❖ Nhấn Enter.



- ❖ Gõ vào ô chứa thư mục nguồn chứa BC3.1 (Ví dụ trên:Giả sử ổ đĩa chứa thư mục BC3.1 trên đĩa CD là D:) → Nhấn Enter.



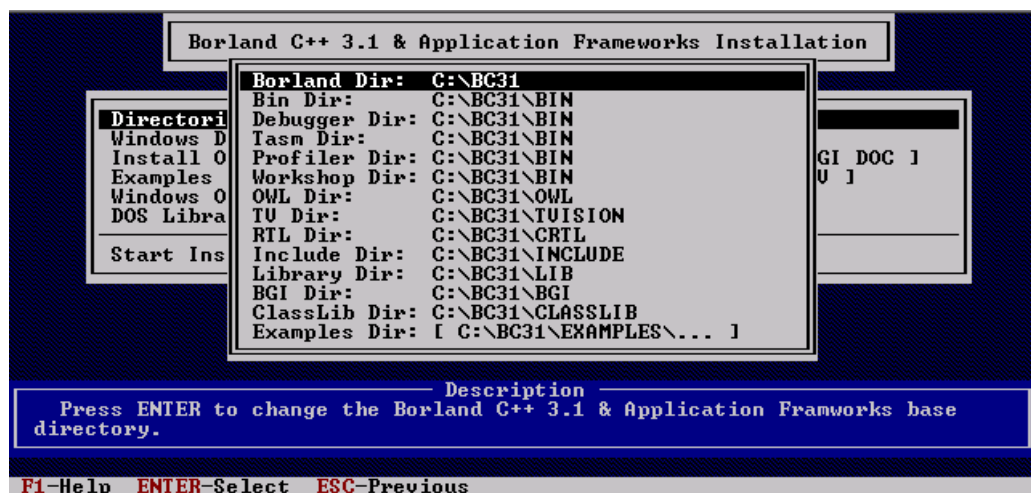
- ❖ Enter tiếp (có thể gỡ lại đường dẫn chứa BC3.1 nếu bước trên gõ sai).



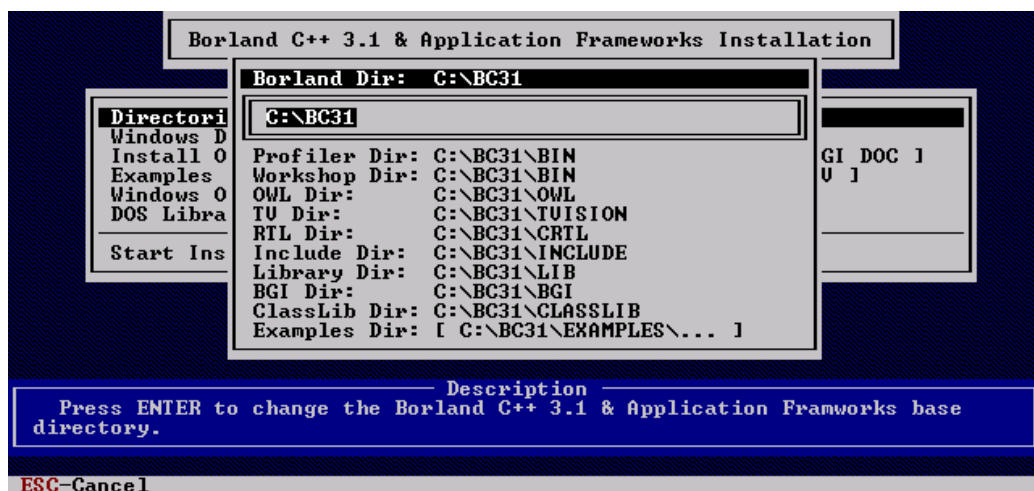
- ❖ Chọn đường dẫn và tên thư mục cần cài đặt BC3.1 lên đĩa cứng.

Ví dụ: Cần cài BC3.1 lên ổ đĩa C: tên thư mục là BC31.

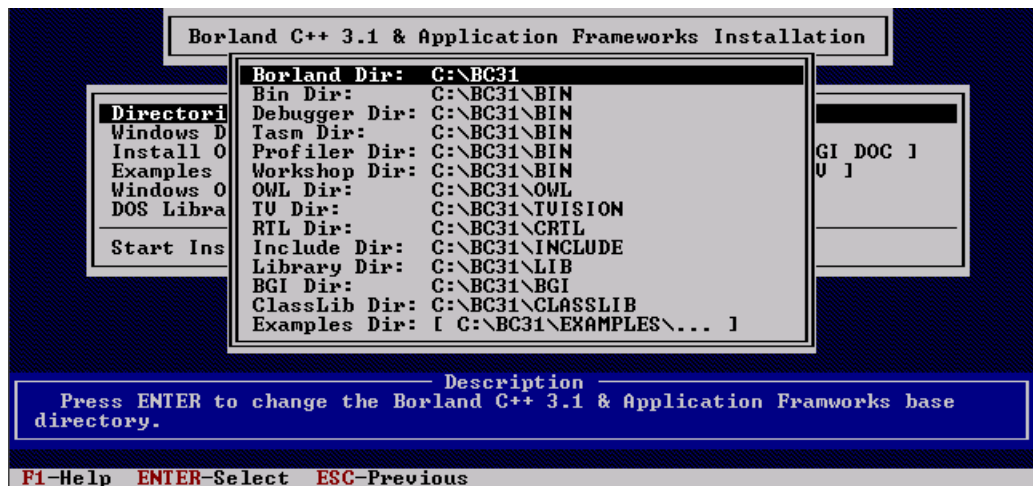
- Để thay đổi thư mục và ổ đĩa cài đặt → di chuyển vệt sáng (dùng phím mũi tên) đến dòng Directories ... như hình trên → sau đó nhấn Enter.



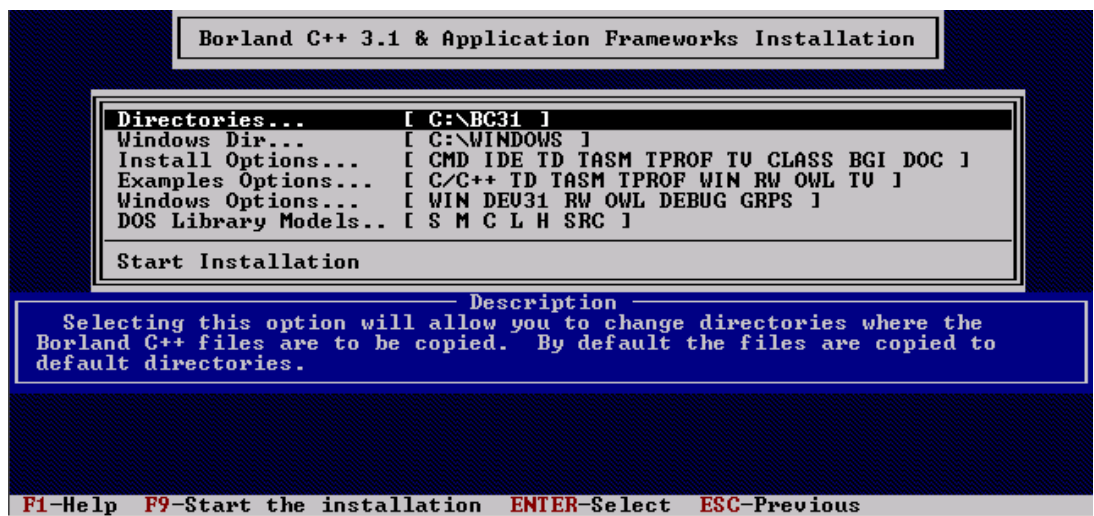
- Nhấn tiếp Enter.



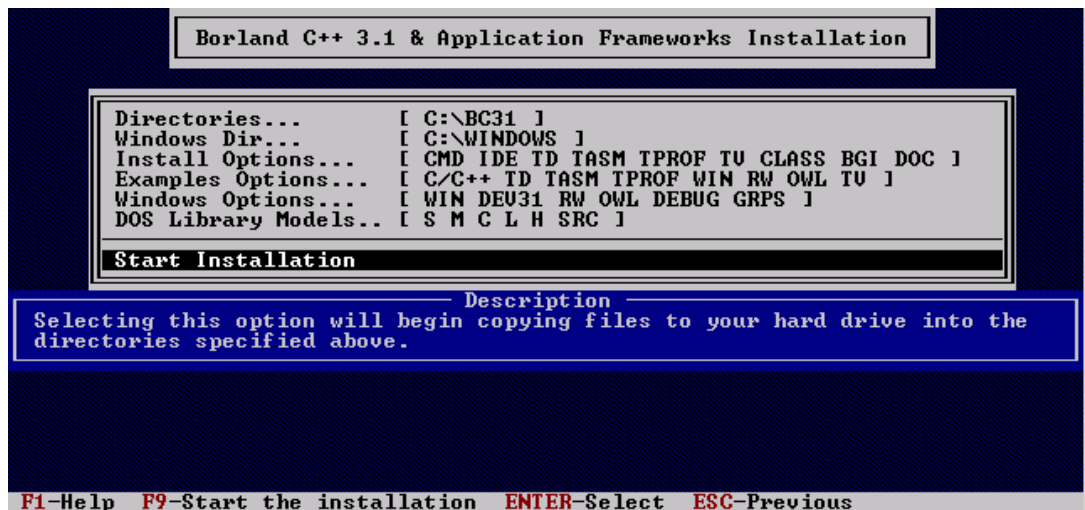
- Gõ tên ổ đĩa và tên thư mục cần cài đặt trong textbox → Enter.
- Sau đó nhấn ESC.



- Tiếp tục nhấn phím ESC.

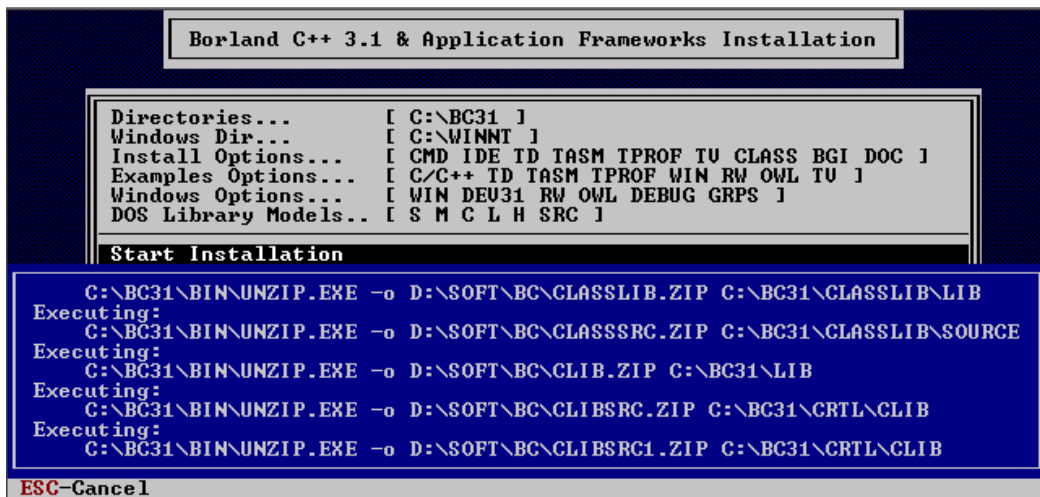


- Tiếp theo kiểm tra xem thư mục cài Windows có đúng đường dẫn như dòng **Windows Dir** hay không (dòng thứ 2). Nếu không đúng thì thay đổi thư mục cho đúng, di chuyển vệt sáng đến đó, thao tác tương tự như thay đổi thư mục BC3.1.
- Thường thì không cần thay đổi vì các máy có cài Windows mặc định là C:\Windows.
- Di chuyển vệt sáng đến dòng **Start Installation** nhấn Enter bắt đầu quá trình cài đặt.

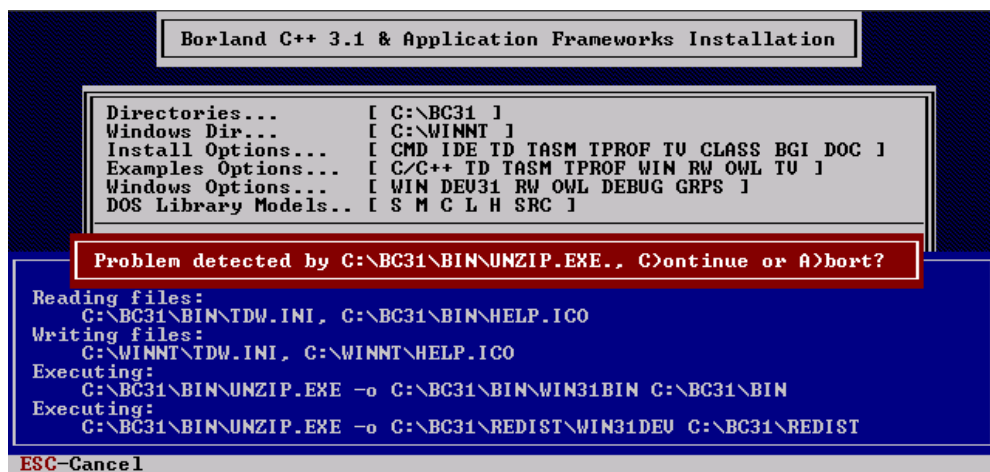


Lưu ý: Ở bước này chỉ thay đổi thư mục cài đặt BC3.1, thư mục Windows (nếu có) còn những mục khác không thay đổi.

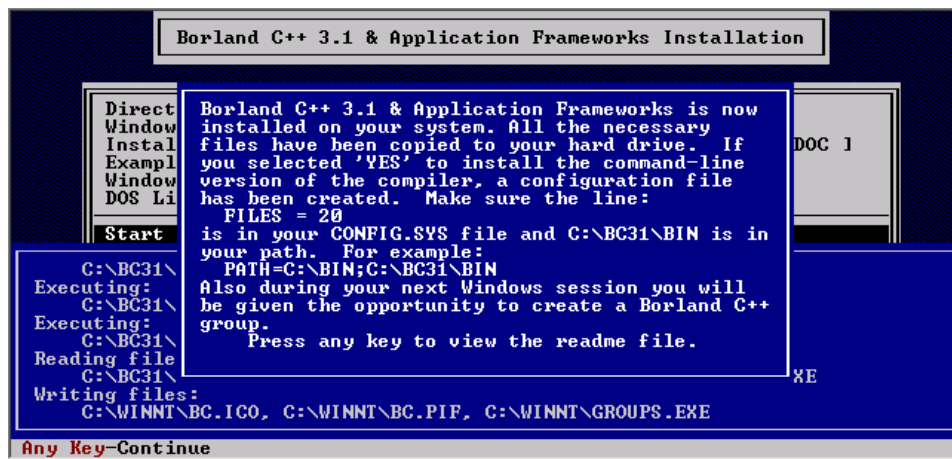
- ❖ Quá trình cài đặt đang thực hiện.



- ❖ Nếu trong quá trình cài đặt gặp thông báo sau:



→ Nhấn phím C để tiếp tục.



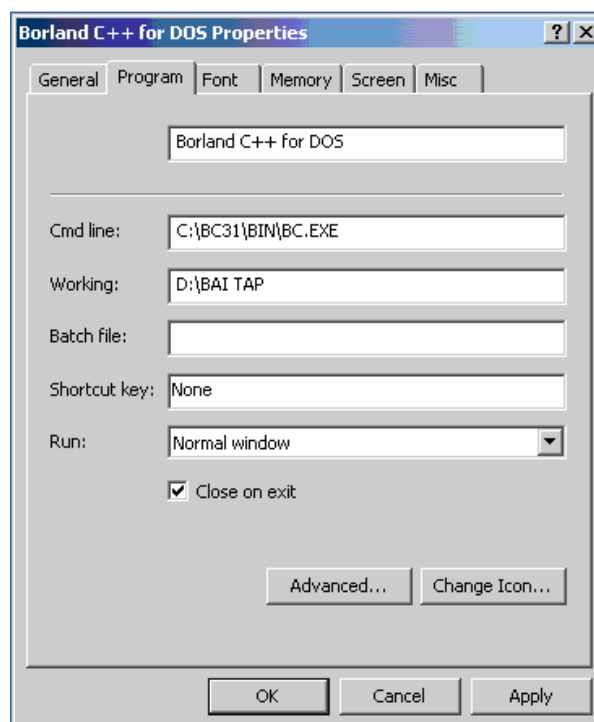
Quá trình cài đặt hoàn tất, nhấn phím **ESC** cho đến khi mất màn hình cài đặt.

❖ Tạo một thư mục để lưu bài tập, chẳng hạn **D:\BaiTap** để làm thư mục làm việc của C, trong quá trình làm bài hay biên dịch chạy chương trình thì tất cả các file đó đều nằm trong thư mục **BaiTap** cho dễ quản lý.

❖ Tạo Shortcut Borland C++3.1 (File bc.exe trong thư mục BIN của thư mục BC31 vừa cài đặt) → Chọn Properties → Chọn Tab Program gõ vào mục Cmd line và Working giống như hình sau nếu cài đặt BC3.1 trên ổ đĩa C:\BC3.1. Nhấn OK.

- Cmd line (đường dẫn đến file chạy BC): C:\BC3.1\BIN\BC.EXE.
- Working (thư mục mới vừa tạo để lưu bài làm): D:\Bai tap

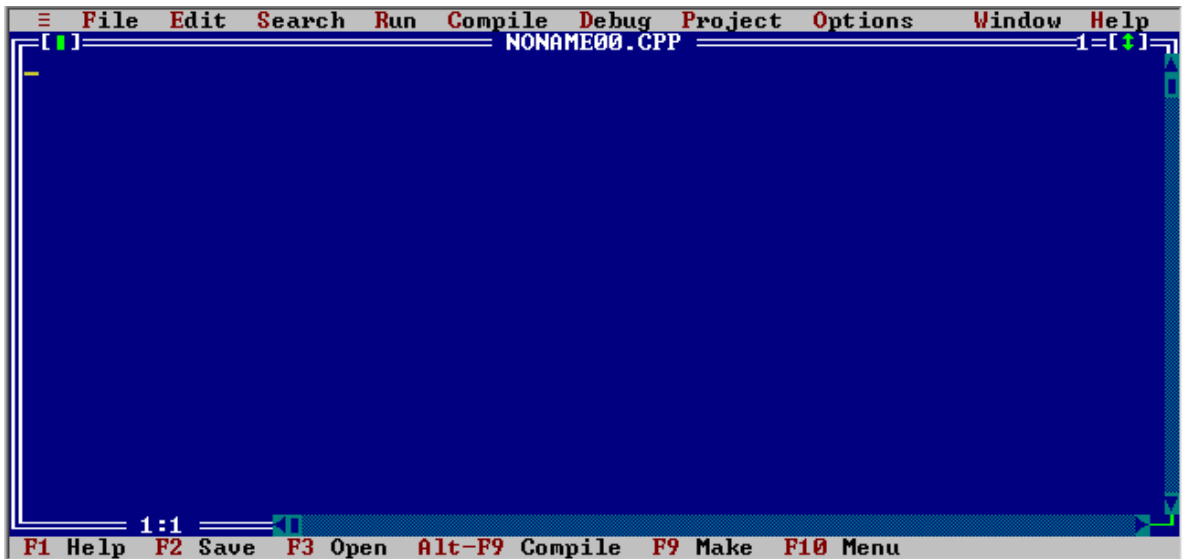
Lưu ý: Đúng đường dẫn thư mục.



II. CÁC BƯỚC VIẾT CHƯƠNG TRÌNH

a. Chuẩn bị viết chương trình

- Khởi động BC3.1



- Vào menu Options\Environment\Editor chỉnh lại Tab size là 4.



b. Các phím chức năng chính

- F3: Mở file chương trình có sẵn.
- F2: Lưu file
- Lưu ý: Chọn đường dẫn và đặt tên file cho đúng. Tên có tối đa 8 ký tự, phần đuôi không cần nhập vào (mặc định là *.cpp).
- F5: Phóng to hoặc trở về kích thước bình thường của cửa sổ soạn thảo.
- F6: Chuyển qua lại các cửa sổ soạn thảo (nếu mở nhiều cửa sổ).
- F9: Biên dịch chương trình. Mục đích là kiểm tra lỗi chương trình.
- Ctr+F9: Thực thi chương trình (Run) khi chương trình không có lỗi.
- Alt+F5: Xem lại màn hình kết quả chương trình đã chạy trước đó.

c. Viết chương trình

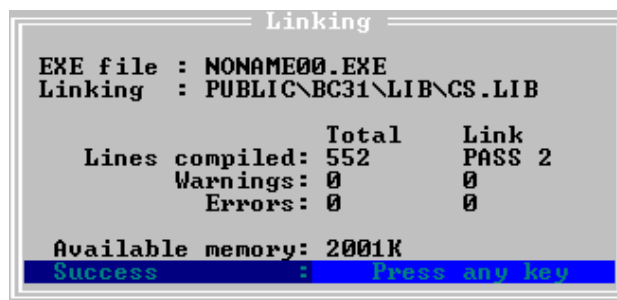
Cấu trúc cơ bản của chương trình gồm phần.

- i. Phần khai báo thư viện hàm.
- ii. Phần khai báo biến toàn cục, khai báo kiểu dữ liệu, khai báo hàm hay khai báo hằng (nếu có).
- iii. Các hàm con (nếu có).
- iv. Hàm main().

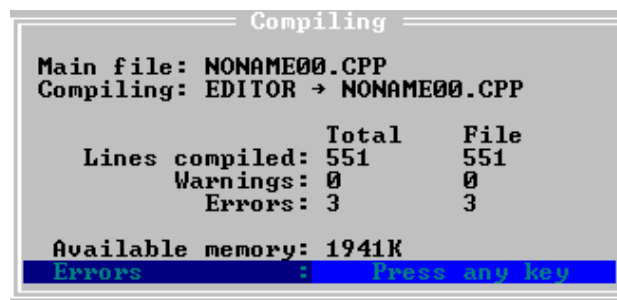
→ Lưu ý trình bày chương trình.

d. Biên dịch và sửa lỗi

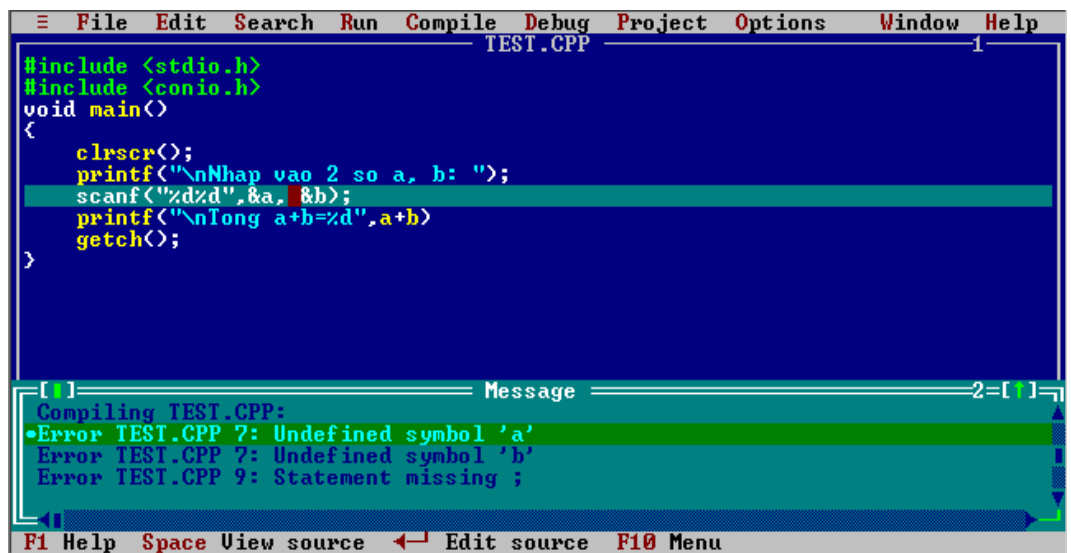
- Sau khi soạn thảo xong chương trình nhấn F2 đặt tên chương trình, để đảm bảo chương trình có thể thực thi được, ta phải nhấn F9 để biên dịch.
- Nếu không có lỗi, ta có thể nhấn Ctrl+F9 để thực thi chương trình.



- Nếu máy bị loop nhấn Ctrl+Break+Enter để trở về màn hình soạn thảo.
- Ngược lại, ta cần phải sửa lỗi cho đến khi hết lỗi.



- Các bước thực hiện khi có lỗi:
 - i. Khi hiển thị màn hình báo lỗi, ta phải nhấn phím Enter để xuất hiện cửa sổ mô tả lỗi (**không nhấn phím ESC**).



- ii. Sử dụng phím mũi tên lên xuống để duyệt lên xuống và xem mô tả lỗi. Khi di chuyển để ý quan sát vệt sáng bên trên khung cửa sổ soạn thảo chương trình. Thông thường vệt sáng sẽ cho biết vị trí lỗi (có thể ngay chính tại dòng có lỗi hoặc trên hoặc dưới một dòng). Có nhiều cách sửa lỗi, nhưng để đơn giản chúng ta nên sửa lỗi từ trên xuống.

e. Một số lỗi thường gặp

| STT | LỖI | MÔ TẢ | KHẮC PHỤC | VÍ DỤ |
|--------------------|------------------------------|---|---|--|
| LỖI CÚ PHÁP | | | | |
| 1 | Statement missing; | Thiếu dấu; khi kết thúc 1 lệnh | Bổ sung thêm dấu ; vào sau khai báo biến hay kết thúc một lệnh. | Sai: int a scanf("%d",&a) Sửa thành: int a; scanf("%d",&a); |
| 2 | Compound statement missing } | Thiếu dấu } khi kết thúc khối lệnh hay làm. | Bổ sung thêm dấu } vào tương ứng | Sai : void main() { int a; scanf("%d",&a); if(a>0) printf("Duong"); Sửa thành: void main() { int a; scanf("%d",&a); if(a>0) printf("Duong"); } |
| 3 | Unexpected } | Thiếu dấu { khi bắt đầu khối lệnh, hàm hay dư dấu } | Kiểm tra xem có dư dấu } hoặc | Sai : void main() |

| | | | | |
|---|-------------------------|---|--|--|
| | | | thiếu dấu { và sửa tương ứng. | <pre>{ int a; scanf("%d",&a); if(a>0) printf("Duong"); }</pre> <p>Sửa thành:</p> <pre>void main() { int a; scanf("%d",&a); if(a>0) printf("Duong"); }</pre> |
| 4 | Misplaced else | Chấm phẩy sau phát biểu if hoặc khối lệnh thực hiện trong phát biểu if chưa đặt trong cặp dấu ngoặc {} | | <p>Sai :</p> <pre>if (a%2); printf("a le"); else if(a>10) printf("a chan"); printf(", > 10"); else printf("a < 10");</pre> <p>Sửa thành:</p> <pre>if (a%2) printf("a le"); else if(a>10) { printf("a chan "); printf(", > 10"); } else printf("a < 10");</pre> |
| 5 | For statement missing ; | Thiếu thành phần trong cú pháp của vòng lặp for hoặc quên dùng dấu chấm phẩy (;) để ngăn cách các thành phần , ... (Phải có đủ 2 dấu chấm phẩy) | Kiểm tra cho đúng cú pháp: for(<Biểu thức khởi>; <biểu thức điều kiện dừng>; <biểu thức tăng giảm>) Trong biểu thức gán hay nhiều thành phần thì mỗi phần cách nhau bởi dấu phẩy (,) | <p>Sai :</p> <pre>for(int i=0, i<n; i++) { printf("a[%d]: ",i); scanf("%d",&a[i]); }</pre> <p>Sửa thành:</p> <pre>for(int i=0; i<n; i++) { printf("a[%d]: ",i); scanf("%d",&a[i]); }</pre> |
| 6 | Function call missing) | Thiếu dấu phẩy phân cách giữa phần định dạng và danh sách biến trong hàm printf và scanf. | Thêm dấu phẩy giữa phần định dạng và danh sách biến. | <p>Sai :</p> <pre>for(int i=0; i<n; i++) { printf("a[%d]: "i); scanf("%d",&a[i]); }</pre> |

| | | | | |
|--------------------------------------|---|--|--|--|
| | | | | <pre> } Sửa thành: for(int i=0; i<n; i++) { printf("a[%d]: ",i); scanf("%d",&a[i]); } </pre> |
| LỖI KHAI BÁO | | | | |
| 1 | Declaration terminated incorrectly | Khai báo tên biến trùng với tên hằng đã định nghĩa trước. | Đổi tên biến. | <pre> Sai : #define MAX 100 void main() { int MAX; } Sửa thành: #define MAX 100 void main() { int x; } </pre> |
| 2 | Multiple declaration for 'i' | Khai báo biến trùng tên, khai báo nhiều lần. | Kiểm tra và bỏ bớt khai báo lại biến hoặc đổi tên biến khác. | <pre> Sai : int i; for(int i=0; i<n; i++) scanf("%d",&a[i][j]); Sửa thành: int x; for(i=0; i<n; i++) scanf("%d",&a[i][j]) </pre> |
| 3 | Undefined symbol 'a' | Sử dụng biến chưa khai báo. | Khai báo biến. | <pre> Sai : printf("Nhap vào n:"); scanf("%d", &n); Sửa thành: int n; printf("Nhap vào n:"); scanf("%d", &n); </pre> |
| 4 | Declaration syntax error | Thiếu dấu ; sau khai báo biến. | Bổ sung dấu ; sau khi kết thúc khai báo biến. | <pre> Sai : int n Sửa thành: Int n; </pre> |
| THƯ VIỆN HÀM HOẶC SAI TÊN HÀM | | | | |
| 1 | Function 'printf' should have a prototype Function 'scanf' should have a prototype Function 'XXX' should have a prototype | Thiếu sai báo thư viện hàm nếu sử dụng hàm thư viện, ngược lại phải kiểm tra xem có khai báo nguyên mẫu hàm, hoặc gọi sai tên hàm. | Bổ sung #include <stdio.h> #include <conio.h> | |

| CÁC CẢNH BÁO | | | | |
|--------------|-------------------------------|--|---------------------------------|---|
| 1 | Possibly incorrect assignment | Dùng ký hiệu trong phép toán quan hệ. | Dùng ký hiệu trong phép so sánh | Sai : <code>if(n%2=0)</code> <code>printf("n chan);</code> Sửa thành: <code>if(n%2==0)</code> <code>printf("n chan");</code> |
| 2 | Code has no effect | Dùng ký hiệu phép toán quan hệ cho phép gán. | Dùng ký hiệu phép toán số học. | |

f. Debug

Mặc dù chương trình không còn lỗi nhưng khi chạy chương trình vẫn ra kết quả sai, những lỗi đó có thể là:

- Dùng chấm phẩy sau: if, else, for, while, ... mà chưa thực hiện lệnh.
- Định dạng nhập xuất sai hay khai báo sai kiểu dữ liệu.
- Chia cho 0.
- Không có điều kiện dừng (điều kiện dừng sai).
- Phân tích thuật toán thiếu (chưa vét hết các trường hợp) hoặc sai.

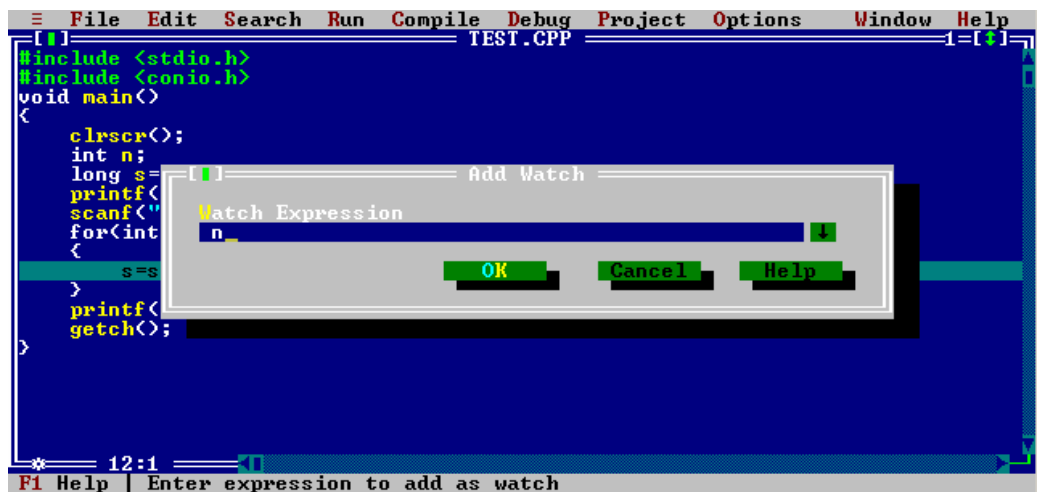
Các thao tác debug:

- Nhấn F7 hoặc F8 để chạy từng bước (nếu không có lỗi khi biên dịch)

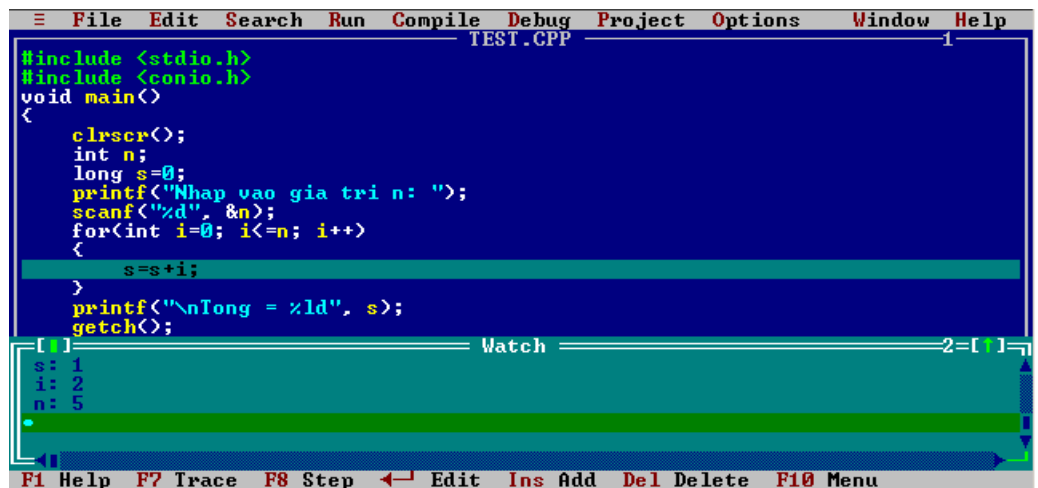
```

#include <stdio.h>
#include <conio.h>
void main()
{
    clrscr();
    int n;
    long s=0;
    printf("Nhap vao gia tri n: ");
    scanf("%d", &n);
    for(int i=0; i<=n; i++)
    {
        s=s+i;
    }
    printf("\nTong = %ld", s);
    getch();
}
    
```

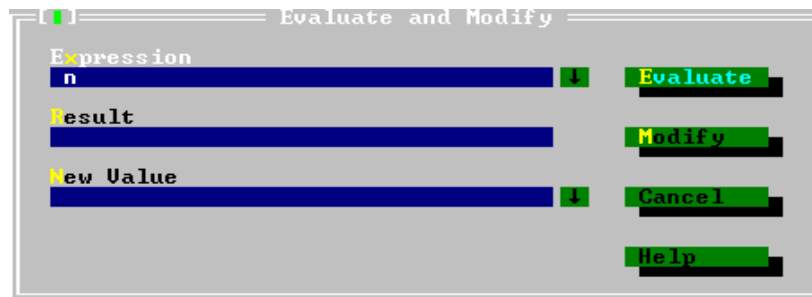
- F7: Đi từng lệnh của hàm con nếu có gọi hàm.
 - F8: không vào chi tiết từng lệnh khi gọi đến hàm con (chỉ đưa ra kết quả của hàm con).
- Quan sát vệt sáng để biết chương trình đang thực hiện đến vị trí lệnh nào.
- Nhấn Ctrl+F7 (hoặc nhấn phím Insert nếu đã có cửa sổ Watch): Nhập vào biến cần theo dõi giá trị các biến khi thực hiện xong lệnh hay hàm nào đó.



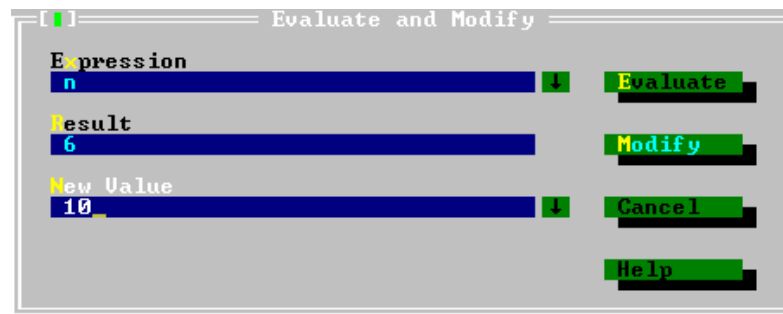
- Có thể xóa biến trên cửa sổ Watch bằng cách chọn biến trên cửa sổ Watch và nhấn phím Delete.
- Nếu không thấy cửa sổ hiển thị giá trị biến (Watch) nhấn Alt+W+W hoặc vào menu Window chọn Watch.



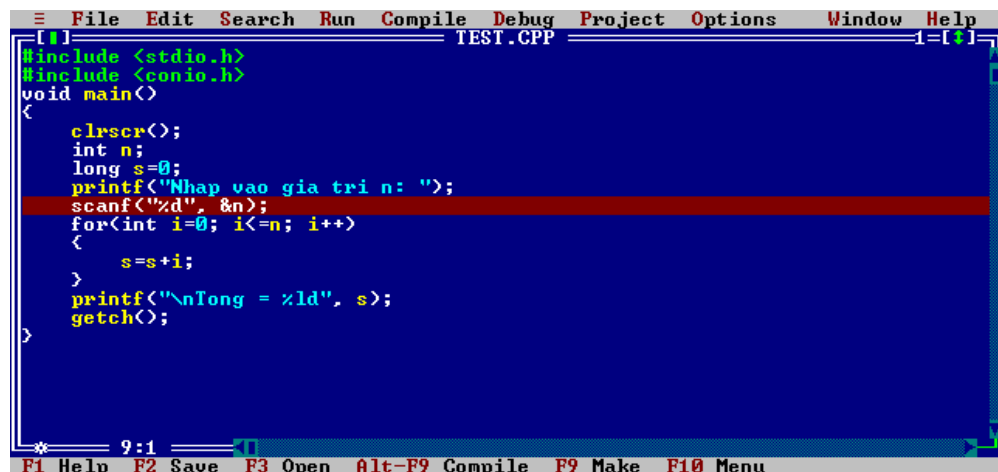
- Nếu muốn bỏ qua một đoạn nào đó (tức không cần kiểm tra đoạn đó) thì nhấn F4 để chương trình thực thi tới vị trí dòng của dấu nhảy rồi dừng lại đó (dấu nhảy phải tại vị trí những dòng phía sau của vệt sáng, nhấn F6 để chuyển qua lại các cửa sổ).
- Muốn thay đổi giá trị của biến ta dùng phím Ctrl+F4 để hiển thị cửa sổ.



- Nhập vào tên biến ở ô Expression, chọn nút Evaluate (hoặc nhấn Enter), ô Result sẽ hiển thị kết quả tại thời điểm đó, sau đó nhập giá trị mới cho biến tại ô New Value → Enter (dùng phím tab để di chuyển vị trí chọn).



- Ngoài ra có thể đánh dấu để chương trình thực thi đến vị trí đánh dấu (khi chưa chạy từng bước) dùng phím F8 để đánh dấu ngay vị trí dấu nhảy. Vị trí đánh dấu sẽ có vệt sáng màu đỏ.



- Có thể đánh dấu nhiều vị trí khác nhau. Nhấn **Ctrl+F9** để chương trình thực thi đến vị trí đánh dấu theo thứ tự từ trên xuống dưới, đồng thời cũng có thể dùng phím **F7** hoặc **F8** giống như trên để chạy từng bước.
- Ngoài ra, có thể dùng phím **ALT+F5** để xem kết quả xuất trong quá trình debug (để kiểm tra nhập xuất).
- Trong quá trình chạy từng bước có thể kết thúc bằng cách nhấn **Ctrl+F2**.

g. Các thao tác liên quan đến cửa sổ Watch

- Di chuyển cửa sổ Watch: Chọn cửa sổ Watch, nhấn **Ctrl+F5**. Sau đó dùng phím mũi tên để di chuyển cửa sổ tới vị trí mới. Nhấn phím Enter.
- Thay đổi kích thước cửa sổ Watch (khi đang chọn bằng **Ctrl+F5** trên cửa sổ Watch) nhấn Shift + phím mũi tên rồi nhấn phím Enter.

TÀI LIỆU THAM KHẢO



1. **PHẠM VĂN ÁT**: “Kỹ thuật lập trình C: cơ sở và nâng cao”. Nhà Xuất Bản Khoa Học Kỹ Thuật – 1996.
2. **LÊ HOÀI BẮC – LÊ HOÀNG THÁI – NGUYỄN TẤN TRẦN MINH KHANG – NGUYỄN PHƯƠNG THẢO**: “Giáo trình ngôn ngữ C”. Nhà Xuất Bản Đại Học Quốc Gia Tp. Hồ Chí Minh – 2003.
3. **NGUYỄN TẤN TRẦN MINH KHANG**: “Bài tập Kỹ thuật lập trình – Tập 1”. Nhà Xuất Bản Đại Học Quốc Gia Tp. Hồ Chí Minh – 2004.
4. **NGUYỄN ĐÌNH TÊ – HOÀNG ĐỨC HẢI**: “Giáo trình lý thuyết & Bài tập ngôn ngữ C”. Nhà Xuất Bản Mũi Cà Mau.
5. **HUỲNH TẤN DŨNG – HOÀNG ĐỨC HẢI**: “Bài tập ngôn ngữ C từ A đến Z”. Nhà Xuất Bản Lao Động – Xã Hội.
6. **NGUYỄN THANH SƠN**: “Tập bài giảng Kỹ thuật lập trình” – 2004.
7. **TRẦN MINH THÁI**: “Tập bài giảng Kỹ thuật lập trình” – 2005.
8. **SANFORD LEESTMA LARRY NYHOFF**: “Pascal Programming and Solving”. Macmillan Publishing Company – 1990.
9. **JOHN R. HUBBARD**: “455 Bài tập cấu trúc dữ liệu cài đặt bằng C++”. Bản dịch của Minh Trung, Gia Việt – Nhà Xuất Bản Thống Kê.

MỤC LỤC

| | |
|---|-----------|
| LỜI MỞ ĐẦU | 1 |
| LỊCH TRÌNH THỰC HÀNH..... | 2 |
| CHƯƠNG 1 LƯU ĐỒ THUẬT TOÁN (FLOWCHART) | 3 |
| I. TÓM TẮT LÝ THUYẾT | 3 |
| <i>I.1. Khái niệm.....</i> | 3 |
| <i>I.2. Phương pháp duyệt</i> | 3 |
| <i>I.3. Các ký hiệu</i> | 3 |
| <i>I.4. Các cấu trúc điều khiển cơ bản</i> | 4 |
| <i>a. Cấu trúc tuần tự</i> | <i>4</i> |
| <i>b. Cấu trúc lựa chọn.....</i> | <i>5</i> |
| <i>c. Cấu trúc lặp.....</i> | <i>6</i> |
| <i>d. Các ví dụ.....</i> | <i>8</i> |
| II. BÀI TẬP | 11 |
| <i>II.1. Bài tập cơ bản</i> | 11 |
| <i>II.2. Bài tập luyện tập và nâng cao.....</i> | 12 |
| III. KẾT LUẬN..... | 12 |
| CHƯƠNG 2 CẤU TRÚC ĐIỀU KHIỂN..... | 13 |
| I. TÓM TẮT LÝ THUYẾT | 13 |
| <i>I.1. Các ký hiệu</i> | 13 |
| <i>I.2. Các kiểu dữ liệu cơ bản trong C.....</i> | 13 |
| <i>I.3. Bảng ký hiệu các phép toán.....</i> | 14 |
| <i>I.4. Các hàm cơ bản.....</i> | 15 |
| <i>I.5. Cấu trúc rẽ nhánh</i> | 15 |
| <i>a. Cấu trúc if.....</i> | <i>15</i> |
| <i>b. Cấu trúc if ... else.....</i> | <i>16</i> |
| <i>I.6. Cấu trúc lựa chọn switch</i> | 16 |
| <i>I.7. Cấu trúc lặp</i> | 18 |
| <i>a. for</i> | <i>18</i> |
| <i>b. while</i> | <i>19</i> |

| | | |
|-----------------|--|-----------|
| c. | <i>do ... while</i> | 20 |
| I.8. | <i>break và continue</i> | 20 |
| a. | <i>break</i> | 20 |
| b. | <i>continue</i> | 21 |
| II. | BÀI TẬP | 21 |
| II.1. | <i>Phương pháp chạy tay từng bước để tìm kết quả chương trình</i> | 21 |
| II.2. | <i>Bài tập cơ bản</i> | 23 |
| a. | <i>Cấu trúc if/ if..else và switch</i> | 23 |
| b. | <i>Cấu trúc lặp</i> | 25 |
| II.3. | <i>Bài tập luyện tập và nâng cao</i> | 29 |
| III. | KẾT LUẬN | 30 |
| CHƯƠNG 3 | HÀM CON | 31 |
| I. | TÓM TẮT LÝ THUYẾT | 31 |
| I.1. | <i>Khái niệm</i> | 31 |
| I.2. | <i>Ví dụ</i> | 31 |
| I.3. | <i>Cấu trúc một chương trình C</i> | 33 |
| a. | <i>Khởi khai báo</i> | 33 |
| b. | <i>Hàm chính (main())</i> | 33 |
| c. | <i>Các hàm con</i> | 33 |
| d. | <i>Nguyên mẫu hàm</i> | 33 |
| I.4. | <i>Cách xây dựng một hàm con</i> | 34 |
| a. | <i>Kiểu dữ liệu của hàm</i> | 34 |
| b. | <i>Tham số</i> | 34 |
| c. | <i>Tên hàm</i> | 35 |
| d. | <i>Ví dụ</i> | 35 |
| II. | BÀI TẬP | 37 |
| II.1. | <i>Bài tập cơ bản</i> | 37 |
| II.2. | <i>Bài tập luyện tập và nâng cao</i> | 39 |
| III. | KẾT LUẬN | 39 |
| CHƯƠNG 4 | MẢNG MỘT CHIỀU | 41 |
| I. | TÓM TẮT LÝ THUYẾT | 41 |
| I.1. | <i>Khái niệm</i> | 41 |
| I.2. | <i>Khai báo mảng</i> | 41 |

| | | |
|-----------------|---|-----------|
| I.3. | <i>Truy xuất phần tử của mảng</i> | 42 |
| II. | BÀI TẬP | 43 |
| II.1. | <i>Một số kỹ thuật cơ bản</i> | 43 |
| a. | <i>Kỹ thuật đặt cờ hiệu</i> | 43 |
| b. | <i>Kỹ thuật đặt lính canh</i> | 44 |
| II.2. | <i>Bài tập cơ bản</i> | 45 |
| a. | <i>Nhập xuất mảng một chiều</i> | 45 |
| b. | <i>Tìm kiếm trên mảng một chiều</i> | 46 |
| c. | <i>Đếm – Tần suất</i> | 47 |
| d. | <i>Tính tổng – Trung bình có điều kiện</i> | 48 |
| e. | <i>Sắp xếp</i> | 49 |
| f. | <i>Xoá</i> | 50 |
| g. | <i>Chèn</i> | 50 |
| h. | <i>Tách / ghép mảng</i> | 51 |
| II.3. | <i>Bài tập luyện tập và nâng cao</i> | 53 |
| III. | KẾT LUẬN | 56 |
| CHƯƠNG 5 | CHUỖI KÝ TỰ | 57 |
| I. | TÓM TẮT LÝ THUYẾT | 57 |
| I.1. | <i>Khái niệm</i> | 57 |
| I.2. | <i>Khai báo chuỗi</i> | 57 |
| I.3. | <i>Các thao tác trên chuỗi</i> | 57 |
| a. | <i>Nhập chuỗi</i> | 57 |
| b. | <i>Xuất chuỗi</i> | 58 |
| c. | <i>Các hàm thư viện (string.h)</i> | 58 |
| d. | <i>Ví dụ</i> | 60 |
| II. | BÀI TẬP | 60 |
| II.1. | <i>Bài tập cơ bản</i> | 60 |
| II.2. | <i>Bài tập luyện tập và nâng cao</i> | 62 |
| III. | KẾT LUẬN | 63 |
| CHƯƠNG 6 | MẢNG HAI CHIỀU | 64 |
| I. | TÓM TẮT LÝ THUYẾT | 64 |
| I.1. | <i>Khái niệm</i> | 64 |
| I.2. | <i>Khai báo mảng</i> | 64 |

| | | |
|-----------------|---|-----------|
| I.3. | <i>Truy xuất phần tử của mảng</i> | 64 |
| I.4. | <i>Ma trận vuông và các khái niệm liên quan</i> | 65 |
| a. | <i>Khái niệm</i> | 65 |
| b. | <i>Tính chất của ma trận vuông</i> | 65 |
| II. | BÀI TẬP | 66 |
| II.1. | <i>Một số kỹ thuật cơ bản</i> | 67 |
| II.2. | <i>Bài tập cơ bản</i> | 69 |
| a. | <i>Bài tập nhập xuất</i> | 69 |
| b. | <i>Bài tập tính tổng</i> | 69 |
| c. | <i>Bài tập tìm kiếm</i> | 70 |
| d. | <i>Bài tập đếm</i> | 70 |
| e. | <i>Bài tập sắp xếp</i> | 71 |
| f. | <i>Bài tập Thêm – Xoá – Thay thế</i> | 72 |
| II.3. | <i>Bài tập luyện tập và nâng cao</i> | 73 |
| III. | KẾT LUẬN | 77 |
| CHƯƠNG 7 | Kiểu dữ liệu có cấu trúc | 78 |
| I. | TÓM TẮT LÝ THUYẾT | 78 |
| I.1. | <i>Khái niệm</i> | 78 |
| I.2. | <i>Định nghĩa kiểu dữ liệu</i> | 78 |
| I.3. | <i>Khai báo</i> | 79 |
| I.4. | <i>Truy xuất</i> | 80 |
| I.5. | <i>Ví dụ minh họa</i> | 81 |
| I.6. | <i>Mảng cấu trúc</i> | 82 |
| I.7. | <i>Nguyên tắc viết chương trình có mảng cấu trúc</i> | 82 |
| II. | BÀI TẬP | 91 |
| II.1. | <i>Bài tập cơ bản</i> | 91 |
| II.2. | <i>Bài Tập Luyện Tập</i> | 92 |
| III. | KẾT LUẬN | 96 |
| CHƯƠNG 8 | TẬP TIN | 97 |
| I. | TÓM TẮT LÝ THUYẾT | 97 |
| I.1. | <i>Khái niệm</i> | 97 |
| I.2. | <i>Thao tác với tập tin</i> | 97 |
| a. | <i>Khai báo</i> | 97 |

| | | |
|------------------|--|------------|
| <i>b.</i> | <i>Mở tập tin</i> | 97 |
| <i>c.</i> | <i>Các hàm đọc ghi nội dung tập tin</i> | 98 |
| <i>d.</i> | <i>Đóng tập tin</i> | 99 |
| <i>e.</i> | <i>Các thao tác khác trên tập tin</i> | 99 |
| <i>f.</i> | <i>Ví dụ minh hoạ</i> | 99 |
| I.3. | <i>Các ví dụ minh hoạ</i> | 100 |
| <i>a.</i> | <i>Tập tin văn bản</i> | 100 |
| <i>b.</i> | <i>Tập tin nhị phân</i> | 102 |
| II. | BÀI TẬP | 103 |
| II.1. | <i>Bài tập cơ bản</i> | 103 |
| II.2. | <i>Bài tập luyện tập và nâng cao</i> | 105 |
| III. | KẾT LUẬN | 108 |
| CHƯƠNG 9 | ĐỆ QUI | 109 |
| I. | TÓM TẮT LÝ THUYẾT | 109 |
| I.1. | <i>Khái niệm</i> | 109 |
| I.2. | <i>Phân loại đệ qui</i> | 109 |
| <i>a.</i> | <i>Đệ qui tuyến tính</i> | 109 |
| <i>b.</i> | <i>Đệ qui nhị phân</i> | 110 |
| <i>c.</i> | <i>Đệ qui phi tuyến</i> | 112 |
| <i>d.</i> | <i>Đệ qui hồi tương</i> | 113 |
| I.3. | <i>Tìm hiểu cách hoạt động của hàm đệ qui</i> | 114 |
| I.4. | <i>Ví dụ</i> | 115 |
| II. | BÀI TẬP | 116 |
| II.1. | <i>Bài tập cơ bản</i> | 116 |
| II.2. | <i>Bài tập luyện tập và nâng cao</i> | 117 |
| III. | KẾT LUẬN | 117 |
| CHƯƠNG 10 | LẬP TRÌNH THEO PHƯƠNG PHÁP PROJECT | 118 |
| I. | MỤC TIÊU | 118 |
| II. | PHƯƠNG PHÁP | 118 |
| II.1. | <i>Tạo một project mới</i> | 118 |
| II.2. | <i>Mở project có sẵn</i> | 118 |
| II.3. | <i>Một số lưu ý</i> | 119 |
| II.4. | <i>Ví dụ minh hoạ</i> | 119 |

| | |
|---|------------|
| III. BÀI TẬP | 123 |
| PHỤ LỤC 1 ĐỀ THI MẪU | 124 |
| PHỤ LỤC 2 HƯỚNG DẪN VIẾT CHƯƠNG TRÌNH TRÊN MÔI TRƯỜNG BORLAND C++ 3.1 (BC31) | 133 |
| I. CÀI ĐẶT BC3.1..... | 133 |
| II. CÁC BƯỚC VIẾT CHƯƠNG TRÌNH | 138 |
| <i>a. Chuẩn bị viết chương trình</i> | <i>138</i> |
| <i>b. Các phím chức năng chính.....</i> | <i>138</i> |
| <i>c. Viết chương trình.....</i> | <i>139</i> |
| <i>d. Biên dịch và sửa lỗi.....</i> | <i>139</i> |
| <i>e. Một số lỗi thường gặp</i> | <i>140</i> |
| <i>f. Debug</i> | <i>143</i> |
| <i>g. Các thao tác liên quan đến cửa sổ Watch.....</i> | <i>145</i> |
| TÀI LIỆU THAM KHẢO | 146 |
| MỤC LỤC | i |