Thực tập KỸ THUẬT LẬP TRÌNH

Tuần 1-3: Xây dựng khung chương trình và menu chọn

Bài toán: Xây dựng chương trình quản lý sinh viên (**QLSV**) với các thông tin cần quản lý của một sinh viên gồm: Mã lớp, Mã sinh viên, Họ và tên, Ngày sinh, Điểm trung bình tích lũy (ĐTBTL). Các chức năng chính của chương trình gồm: Thêm, sửa, xóa hồ sơ sinh viên; In danh sách sinh viên theo lớp hoặc toàn bộ; Sắp xếp danh sách sinh viên theo một (hoặc nhiều) các tiêu chí: Họ tên, Ngày sinh, ĐTBTL bằng các thuật toán sắp xếp chọn, chèn, quicksort, mergesort, heapsort; Tìm kiếm sinh viên theo một (hoặc nhiều) các tiêu chí: Họ tên, Ngày sinh, ĐTBTL bằng các thuật toán tìm kiếm tuần tự, tìm kiếm nhị phân; Thực hiện các báo cáo thống kê phần trăm xếp loại học tập theo lớp, tổng số sinh viên theo lớp. Dữ liệu được lưu trữ dạng file nhị phân có cấu trúc. Chương trình được viết trên C/C++.

I. Yêu cầu

Xây dựng khung chương trình và giao diện dạng menu với nội dung như sau:

- Giao diện chính gồm các mục chọn:
    1. Thêm mới hồ sơ (M1)
    2. In danh sách (M2)
    3. Sắp xếp (M3)
    4. Tìm kiếm (M4)
    5. Thống kê (M5)
    6. Thoát (M5)
- Khi chọn M1, chương trình cho phép nhập vào hồ sơ sinh viên gồm các thông tin:
    - Mã lớp
    - Mã sinh viên
    - Họ và tên
    - Ngày sinh
    - Điểm trung bình tích lũy

- Khi chọn M2 chương trình cho phép in ra danh sách sinh viên theo thứ tự đã sắp xếp (khi chọn M3) và tìm kiếm (khi chọn M4)
- Khi chọn M3 chương trình cho phép chọn thuật toán sắp xếp(chọn, chèn, quicksort, mergersort) và khóa để sắp xếp (mã sinh viên, họ và tên, ngày sinh, điểm trung bình tích lũy). Có thể xây dựng các mục chọn này dạng menu (cấp 2).
- Khi chọn M4 chương trình cho phép chọn thuật toán tìm kiếm (tuần tự, nhị phân), khóa cần tìm kiếm (mã lớp, mã sinh viên, Họ và tên, ngày sinh, điểm trung bình tích lũy) và giá trị của khóa cần tìm. Có thể xây dựng các mục chọn này dạng menu (cấp 2).
- Khi chọn M5 chương trình cho phép chọn báo cáo số lượng SV theo lớp hoặc tỷ lệ phân loại kết quả học tập (xuất sắc, giỏi, khá, trung bình, yếu) theo lớp. Có thể xây dựng các mục chọn này dạng menu (cấp 2).
- Khi chọn M6 chương trình kết thúc.

II. Kiến thức liên quan

1. Lệnh if

Lệnh *if* cho phép chương trình có thể thực hiện công việc này hay công việc khác tùy thuộc vào điều kiện nào đó của dữ liệu là đúng hay sai. Nói cách khác câu lệnh *if* cho phép người lập trình lựa chọn một trong hai công việc cần làm tùy thuộc vào điều kiện logic nào đó.

Cú pháp (dạng 1):

```
if (Biểu_thức_logic)
    {Các lệnh cho công việc 1}
else
    {Các lệnh cho công việc 2}
```

Hoặc (dạng 2):

```
if (Biểu_thức_logic)
    {Các lệnh cho công  việc 1}
```

Trong đó:

- *Biểu_thức_logic*: Biểu thức logic, biểu thức này sẽ trả về một trong hai giá trị là đúng (*true*) hoặc (*false*);
- *Các lệnh cho công việc 1*: Các lệnh nhằm thực hiện công việc thứ nhất khi *Biểu_thức_logic* trả về kết quả là *đúng*;
- *Các lệnh cho công việc 2*: Các lệnh nhằm thực hiện công việc thứ hai khi *Biểu_thức_logic* trả về kết quả là *sai* (nếu lệnh *if* được viết theo dạng 1).

Cách thực hiện:

1. Đầu tiên chương trình sẽ tính giá trị của *Biểu_thức_logic,* nếu kết quả là đúng thì *Các lệnh cho công việc 1* sẽ được thực hiện;
2. Nếu *Biểu_thức_logic* kết quả trả về là sai và câu lệnh *if* viết theo dạng 1 thì

Các lệnh cho công việc 2 sẽ được thực hiện.

3. Kết thúc lệnh *if*.

Với cú pháp dạng 2 thì khi *Biểu_thức_logic* trả về kết quả là sai thì chương trình cũng không thực hiện bất kỳ công việc gì.

Ví dụ 1.6: Viết chương trình cho phép giải phương trình bậc nhất *a*x + b = 0*.

```
#include <stdio.h>
#include<conio.h>
int main ()
{
    int a, b; // bieu dien cac he so
    float x; // bieu dien nghiem cua phuong trinh
    printf("Nhap vao cac he so a,b:");
    scanf("%d %d",&a,&b);
    if (a==0)
    {
        printf("Phuong trinh khong co nghiem");
    }
    else
    {
       x=(float)(-b)/a;
       printf("Phuong trinh co nghiem x = %0.5f",x);
    }
    getch();
    return 0;
}
```

Trong ví dụ trên, lệnh *if (a==0)* cho phép kiểm tra xem nếu *a = 0* thì chương trình sẽ in ra dòng thông báo *"Phuong trinh khong co nghiem"* và ngược lại (*else*) thì lệnh *x=(float)(-b)/a* tính nghiệm và lệnh *printf("Phuong trinh co nghiem x = %0.5f",x)* in ra kết quả đó.

Ví dụ 1.7: Viết chương trình cho phép nhập vào một số nguyên dương là tháng trong năm và in ra số ngày của tháng đó, biết rằng tháng 1, 3, 5, 7, 8, 10, 12 có 31 ngày; tháng 4, 6, 9, 10 có 30 ngày; và tháng 2 có 28 hoặc 29 ngày.

```
#include <stdio.h>
#include<conio.h>
int main ()
{
    int thg;
    printf("Nhap vao thang trong nam !");
    scanf("%d",&thg);
    if(thg==1||thg==3||thg==5||thg==7||thg==8||thg==10||thg==12)
    {
       printf("\n Thang %d co 31 ngay",thg);
    }
    else
    {
       if (thg==4||thg==6||thg==9||thg==11)
             printf("\n Thang %d co 30 ngay",thg);
```

```
        else
           if (thg==2)
               printf("\n Thang %d co 28 hoac 29 ngay",thg);
           else
               printf("Khong co thang %d",thg);
     }
     getch();
     return 0;
}
```

Ví dụ trên minh họa việc sử dụng câu lệnh *if … else …* , các lệnh:

```
{
   if (thg==4||thg==6||thg==9||thg==11)
         printf("\n Thang %d co 30 ngay",thg);
   else
      if (thg==2)
         printf("\n Thang %d co 28 hoac 29 ngay",thg);
      else
         printf("Khong co thang %d",thg);
}
```

sau từ khóa *else* thứ nhất chỉ được thực hiện nếu tháng nhập vào không phải là một trong các giá trị 1, 3, 5, 7, 8, 10, 12. Lệnh *printf("\n Thang %d co 30 ngay",thg)* chỉ được thực hiện khi tháng nhập vào không phải là một trong các giá trị 1, 3, 5, 7, 8, 10, 12 mà thuộc vào một trong các giá trị 4, 6, 9, 11. Lệnh *printf("Khong co thang %d",thg)* được thực hiện khi tháng nhập vào không phải là giá trị nằm trong khoảng từ 1 đến 12.

Chú ý:

- Khi biểu diễn *Biểu_thức_logic*, nên nhớ rằng phép so sánh bằng trong C/C++ là dấu *==*, trong khi dấu *=* là phép gán. Thêm nữa, khi người lập trình sử dụng nhầm phép so sánh bằng với phép gán trong *Biểu_thức_logic* thì nói chung trình biên dịch không báo lỗi. Ví dụ trong đoạn lệnh sau:

```
else if (thg==2)
    printf("\n Thang %d co 28 hoac 29 ngay",thg);
else
    printf("Khong co thang %d",thg);
```

  nếu thay *(thg==2)* bởi *(thg=2)* thì chương trình xét về cú pháp là không lỗi, tuy nhiên về ý nghĩa là hoàn toàn sai.

- Khi viết lệnh *if,* một số người do sơ xuất hoặc hiểu nhầm nên đặt dấu chấm phảy (;) ngay sau *Biểu_thức_logic*, trong một số tình huống cách viết này không xảy ra lỗi cú pháp nhưng về ý nghĩa cũng hoàn toàn sai. Ví dụ xét đoạn chương tình sau:

```
…
printf("nhap vao mot so nguyen: ");
scanf("%d", &a);
if (a%7==0);
    printf("So %d chia het cho 7",a);
getch();
…
```

Đoạn chương trình trên khi biên dịch sẽ không báo sai lỗi cú pháp, tuy nhiên dòng lệnh *printf("So %d chia het cho 7",a);* luôn được thực hiện với bất kỳ giá trị nào của *a*.

2. Lệnh switch

Nếu lệnh *if* chỉ cho phép lựa chọn một trong nhiều nhất là hai công việc để thực hiện thì lệnh *switch* cho phép chương trình lựa chọn một trong nhiều công việc để thực hiện.

Cú pháp:

```
switch (Biểu_thức_điều_kiện)
   {
      case  Giá_trị_1:
            Các lệnh cho công việc 1
            [break;]
      case  Giá_trị_2:
            Các lệnh cho công việc 2
            [break;]
      …
      case  Giá_trị_n:
            Các lệnh cho công việc n
            [break;]
      [default:
            Các lệnh cho công việc n+1]
   }
```

Trong đó:

- *Biểu_thức_điều_kiện*: Biểu thức điều kiện để xác định công việc cần làm, biểu thức này phải trả về giá trị nguyên hoặc ký tự;
- *Giá_trị_1, Giá_trị_2, .. , Giá_trị_n*: là các hằng nguyên hoặc ký tự;
- *Các lệnh cho công việc 1*: Các lệnh nhằm thực hiện công việc thứ 1 khi giá trị của biểu thức điều kiện bằng *Giá_trị_1;*

  …
- *Các lệnh cho công việc n*: Các lệnh nhằm thực hiện công việc thứ *n* khi giá trị của biểu thức điều kiện bằng *Giá_trị_n*;

Cách thực hiện:

1. Tính giá trị của biểu thức *Biểu_thức_điều_kiện*;
2. So sánh kết quả của biểu thức điều kiện lần lượt với các giá trị *Giá_trị_1, Giá_trị_2, … , Giá_trị_n*, nếu giá trị của biểu thức điều kiện bằng giá trị của nhánh (*case*) thứ *i* là *Giá_trị_i* thì chương trình sẽ thực hiện bắt đầu từ dãy *Các lệnh cho công việc i* cho đến khi gặp lệnh *break*, hoặc nếu không gặp lệnh *break* nào thì sẽ thực hiện cho đến hết lệnh *switch*.
3. Nếu quá trình so sánh không gặp trường hợp *Giá_trị_i* nào bằng với giá trị của *Biểu_thức_điều_kiện* thì chương trình thực hiện dãy các *Các lệnh cho công việc n+1* trong nhánh *default* nếu có.

Trường hợp câu lệnh *switch* không có nhánh *default* và *Biểu_thức_điều_kiện* không khớp với bất cứ nhánh *case* nào thì lệnh *switch* đó không thực hiện bất kỳ công việc nào.

Ví dụ 1.8: Giả sử thời khóa biểu tuần của một sinh viên như sau: thứ 2 học *Giải*

tích, thứ 3 học *Đại số tuyến tính*, thứ 4 học *Anh văn*, thứ 5 học *Kỹ thuật lập trình*, thứ 6 học *Vật lý đại cương*, thứ 7 học *Hóa học đại cương* và chủ nhật là ngày nghỉ. Hãy viết chương trình cho phép nhập vào một ngày trong tuần và in ra công việc cần làm của sinh viên trong ngày đó.

```
#include<stdio.h>
#include<conio.h>
int main()
{
    int thu;
    printf("Nhap vao thu (2-8, 8 la CN):");
    scanf("%d",&thu);
    switch(thu)
    {
        case 2:printf("Giai tich");
            break;
        case 3:printf("Dai so tuyen tinh");
            break;
        case 4:printf("Anh van");
            break;
        case 5:printf("Ky thuat lap trinh");
            break;
        case 6:printf("Vat ly dai cuong");
            break;
        case 7:printf("Hoa hoc dai cuong");
            break;
        case 8:printf("Nghi hoc");
            break;
        default:printf("Nhap sai ngay!");
    }
    getch();
    return 0;
}
```

Ví dụ này sử dụng biểu thức điều kiện của lệnh ***switch*** và các giá trị hằng trong mỗi nhánh *case* là số nguyên.

Ví dụ 1.9: Nhập vào 2 số nguyên và 1 ký tự biểu diễn phép toán. Nếu phép toán là '+', '-', '*' thì in ra kết qua là tổng, hiệu, tích của 2 số; nếu phép toán là '/' thì kiểm tra xem nếu số thứ 2 khác không thì in ra thương của chúng, ngược lại thì in ra thông báo "*khong chia cho 0*".

```
#include <stdio.h>
#include<conio.h>
int main ()
{
    int so1, so2;
    float thuong;
    char pheptoan;
    printf("\n Nhap vao 2 so nguyen ");
    scanf("%d%d",&so1,&so2);
```

```
    fflush(stdin);//Xoa ky tu enter trong vung dem truoc khi nhap phep toan
    printf("\n Nhap vao phep toan ");
    scanf("%c",&pheptoan);
    switch(pheptoan)
    {
        case '+':
            printf("\n %d + %d =%d",so1, so2, so1+so2);
            break;
        case '-':
            printf("\n %d - %d =%d",so1, so2, so1-so2);
            break;
        case '*':
            printf("\n %d * %d =%d",so1, so2, so1*so2);
            break;
        case '/':
            if (so2!=0)
                {
                        thuong=float(so1)/float(so2);
                        printf("\n %d / %d =%f", so1, so2, thuong);
            }
                    else
                    printf("Khong chia duoc cho 0");
            break;
        default :
            printf("\n Chua ho tro phep toan %c", pheptoan);
              break;
    }
    getch();
    return 0;
}
```

Ví dụ này sử dụng biểu thức điều kiện của lệnh *switch* và các giá trị hằng trong mỗi nhánh *case* là ký tự.

Ví dụ 1.10: Viết chương trình cho phép nhập vào một số nguyên dương là tháng trong năm và in ra số ngày của tháng đó, biết rằng tháng 1, 3, 5, 7, 8, 10, 12 có 31 ngày; tháng 4, 6, 9, 10 có 30 ngày; tháng 2 có 28 hoặc 29 ngày.

```
#include <stdio.h>
#include<conio.h>
int main ()
{
    int thang;
    printf("\n Nhap vao thangs trong nam ");
    scanf("%d",&thang);
    switch(thang)
    {
        case 1:
        case 3:
        case 5:
```

```
            case 7:
            case 8:
            case 10:
            case 12:
                printf("\n Thang %d co 31 ngay ",thang);
                break;
            case 4:
            case 6:
            case 9:
            case 11:
                printf("\n Thang %d co 30 ngay ",thang);
                break;
            case 2:
                printf ("\ Thang 2 co 28 hoac 29 ngay");
                break;
            default :
                printf("\n Khong co thang %d", thang);
                break;
        }
        getch();
        return 0;
    }
```

Ví dụ trên minh họa cách sử dụng lệnh *break* để điều khiển việc kết thúc lệnh *switch*, cách viết:

```
    case 1:
    case 3:
    case 5:
    case 7:
    case 8:
    case 10:
    case 12:
        printf("\n Thang %d co 31 ngay ",thang);
        break;
```

cho phép thực hiện tất cả các nhánh này với cùng một dòng in ra kết quả số tháng là 31 và kết thúc bằng *break*. Tương tự như vậy cho các trường hợp tháng nhập vào là 4, 6, 9 và 11. Các lệnh:

```
    default :
        printf("\n Khong co thang %d", thang);
        break;
```

được thực hiện khi giá trị nhập vào không nằm trong khoảng từ 1 đến 12.

Chú ý:

- Để lệnh *switch* chỉ thực hiện duy nhất các lệnh cho công việc thứ *i* (khi *Biểu_thức_điều_kiện=Giá_trị_i*) thì cuối dãy lệnh thứ *i* thêm vào lệnh *break* để kết thúc lệnh *switch*, xem ví dụ 1.8 và 1.9;

Lệnh *break* trong phần *default* của lệnh *switch* là không cần thiết.

3. Lệnh for

Cho phép thực hiện công việc nào đó lặp đi lặp lại một số lần.

Cú pháp:

```
for ( [Khởi_tạo]; [Kiểm_tra]; [Biến_đổi])
   {Các lệnh}
```

Trong đó:

- *Khởi_tạo*: Là một biểu thức hoặc một số câu lệnh đơn. Phần này thường được dùng để khởi tạo giá trị ban đầu cho một biến đếm dùng để kiểm soát số bước lặp;
- *Kiểm_tra*: Là một biểu thức hoặc một số câu lệnh đơn. Phần này thường được dùng để kiểm tra điều kiện kết thúc của vòng lặp bằng một biểu thức logic;
- *Biến_đổi*: Là một biểu thức hoặc một số câu lệnh đơn. Phần này thường được dùng để thay đổi giá trị biến đếm.

Cách thực hiện:

4. Trước tiên *Khởi_tạo* được thực hiện nhằm khởi tạo giá trị ban đầu cho các biến điều khiển của vòng lặp;
5. Tiếp đến là phần *Kiểm_tra* được tính toán, nếu nó trả về giá trị là đúng (1) thì *{Các_lệnh}* sẽ được thực hiện, ngược lại thì vòng lặp *for* sẽ chuyển đến bước kết thúc (bước 4);
6. Sau khi thực hiện được một vòng lặp thi *Biến_đổi* được thực hiện nhằm làm thay đổi giá trị của biến điều khiển, sau đó điều khiển được chuyển về bước 2; và vòng lặp sẽ tiếp tục mãi cho đến khi *Kiểm_tra* có giá trị bằng sai (0).
7. Kết thúc vòng lặp.

Ví dụ 1.11: Viết chương trình để in các số từ 1 đến 10 ra màn hình.

```
#include <stdio.h>
#include<conio.h>
int main ()
{
    printf("Day so tu 1 den 10 :\n");
    for (int i=1; i<=10; i++)
      {
          printf("%d \n",i);
      }
    getch();
    return 0;
}
```

Trong ví dụ trên *Khởi_tạo* là khai báo biến nguyên i và gán giá trị 0 cho nó; *Kiểm_tra* là biểu thức logic để kiểm tra *i<=10* xem *i* có nhỏ hơn hoặc bằng 10 hay không và *Biến_đổi* là phép tăng biến đếm *i* lên 1 đơn vị *i++*.

Ví dụ 1.12: Viết chương trình cho phép nhập vào số *n*, in ra tổng các số nguyên từ 1 đến *n* và tổng của chúng.

```
#include <stdio.h>
#include<conio.h>
int main ()
{
```

```
    unsigned int n,i,tong;
    printf("\n Nhap vao so nguyen duong n:"); scanf("%d",&n);
    tong=0;
    for (i=1; i<=n; i++)
      {
          printf("\n %d ",i);
          tong+=i;
      }
    printf("\n Tong tu 1 den %d =%d ",n,tong);
    getch();
    return 0;
}
```

Trong chương trình trên, khi chạy cần lưu ý đến độ lớn của *n*, khi *n* lớn thì tổng các số từ 1 đến *n* sẽ rất lớn, khi đó kiểu *unsigned int* của biến *tong* có thể sẽ không phù hợp nữa. Trong ví dụ trên, lệnh *tong=0* được đặt trước vòng lặp *for* là bắt buộc; trong vòng *for* lệnh *tong+=i* có thể được thay bằng *tong = tong + i*.

Ví dụ 1.13: Giả sử tiền gửi tiết kiệm được tính với lãi suất là *m%* mỗi tháng, sau *n* tháng thì tiền lãi được cộng vào gốc. Viết chương trình cho phép tính và in ra màn hình số tiền có được sau *K* tháng gửi tiết kiệm với số tiền gốc ban đầu là *T*.

```
#include<stdio.h>
#include<conio.h>
int main()
{
    float m,T,lai;
    int n, K;
    printf("Lai suat              : ");     scanf("%f",&m);
    m=m/100;
    printf("So thang de lai vao goc: ");     scanf("%d",&n);
    printf("So tien gui           : ");    scanf("%f",&T);
    printf("So thang gui          : ");  scanf("%d",&K);
    lai= 0;
    for (int i=1; i<=K; i++)
      {
          lai = lai+ m*T;
          if (i%n==0)
          {
            T=T+lai;
            lai =0;
          }
      }
    printf("So tien: %0.5f",T+lai);
    getch();
    return 0;
}
```

Bài toán nêu trong ví dụ trên có nhiều cách giải, tuy nhiên cách giải như trong chương trình nêu trên thể hiện một cách tự nhiên quá trình giải quyết vấn đề. Trong ví dụ trên, đoạn chương trình:

```
if (i%n==0)
{
    T=T+lai;
     lai =0;
 }
```

cho phép kiểm tra xem nếu tháng đang tính *i* chia hết cho *n* (*%* là phép lấy phần dư) thì số tiền gốc *T* được công thêm phần lãi *lai*, đồng thời khi đó tiền lãi sẽ bằng 0.

Chú ý:

- *Khởi_tạo* và *Biến_đổi*: có thể chứa nhiều câu lệnh đơn, mỗi lệnh cách nhau bởi một dấu phảy (,). Ví dụ:

```
#include <stdio.h>
int main()
{
    int i, j;
    for ( i = 5, j = 10 ; i + j < 20; i++, j++ )
        printf("\n i + j = %d", (i + j) );
}
```

  Phần khởi tạo thực hiện hai lệnh gán *i = 5* và *j = 10*, và phần biến đổi thực hiện hai lệnh *i++* và *j++*.

- Các thành phần *Khởi_tạo*, *Kiểm_tra* và *Biến_đổi* trong lệnh *for* là tùy chọn, nghĩa là có thể có hoặc không. Khi bị bỏ qua thì phần *Kiểm_tra* luôn được nhận giá trị là đúng. Ví dụ chương trình sau đây sẽ thực hiện lặp vô tận:

```
#include <stdio.h>
int main()
{
    int i, j;
    for (  ;   ;  )
        printf( "\n i + j = %d", (i + j) );
}
```

- Mục dù các thành phần *Khởi_tạo, Kiểm_tra* và *Biến_đổi* trong lệnh *for* được thiết kế với mục đích là: khởi tạo, kiểm tra và biến đổi biến điều khiển của vòng lặp thì người lập trình vẫn có thể sử dụng chúng vào các mục đích khác. Ví dụ có thể sử dụng lệnh *printf* ở phần *Kiểm_tra* như sau:

```
#include <stdio.h>
int main()
{
     int i;
     for( i = 0; i < 5; printf("%d\n", i), i++)    ;
}
```

4. Lệnh while

Cho phép thực hiện công việc nào đó lặp đi lặp lại một số lần.

Cú pháp:

```
while (Biểu_thức_điều_kiện)
    {Các lệnh}
```

Trong đó:

- *Biểu_thức_điều_kiện* là biểu thức điều kiện dùng để điều khiển vòng lặp.

Cách thực hiện:

1. Tính giá trị *Biểu_thức_điều_kiện*, nếu giá trị nhận được khác không thì thực hiện bước 2, nếu bằng không thì chuyển đến bước 3;
2. Thực hiện các lệnh *{Các lệnh}* sau đó trở lại bước 1;
3. Kết thúc lệnh *while*.

Ví dụ 1.14: Viết chương trình để in các số từ 1 đến 10 ra màn hình sử dụng vòng lặp *while*.

```
#include <stdio.h>
#include<conio.h>
int main ()
{
    int i;
    printf("Day so tu 1 den 10 :\n");
    i=1;
    while (i<=10)
    {
        printf("%d \n",i);
        i++;
    }
    getch();
    return 0;
}
```

Ví dụ này thực hiện công việc tương tự như trong ví dụ 1.11 là in ra màn hình 10 số từ 1 đến 10. Vòng lặp *while (i<=10)* kết thúc khi *i>10*, lưu ý rằng trong thân vòng lặp phải làm biến đổi *Biểu_thức_điều_kiện*, trong trường hợp này là biến *i*. Để đảm bảo tính đúng đắn của chương trình thì giá trị khởi tạo của biến *i* trước khi thực hiện vòng lặp *i=1* là rất quan trọng.

Ví dụ 1.15: Cho hai số nguyên dương *a* và *b*, viết chương trình tính và in ra ước số chung lớn nhất (*USCLN*) của hai số đó.

```
#include <stdio.h>
#include<conio.h>
int main ()
{
    int a, b, m, n, tg;
    printf("Nhap vao cac he so a,b:");
    scanf("%d %d",&a,&b);
    m=a, n=b;
    while (n>0)
    {
       tg = m % n;
       m = n;
       n = tg;
    }
    printf("USCLN cua %d va %d la: %d",a,b,m);
    getch();
    return 0;
}
```

Giải thuật trên được xây dựng theo thuật toán *Euler*, phần chính của thuật toán

này là vòng lặp khi số "*nhỏ*" hơn còn lớn hơn không. Trong chương trình này, các giá trị nhập vào được lưu vào hai biến *a, b;* sau đó dùng các biến *m, n* để lưu các giá trị này và dùng để tính toán. Vòng lặp trong ví dụ trên kết thúc khi *n<=0*, với *n* là phần dư của phép chia *m* cho *n*. Trong ví dụ trên, khi người sử dụng nhập giá trị cho *b* là *0* thì chương trình còn đúng không? Bạn đọc tự tìm hiểu và giải thích để hiểu hơn lệnh này.

Ví dụ 1.16: Viết chương trình cho phép nhập một kí tự từ bàn phím và ghi mã *ASCII* của nó ra màn hình. Chương trình kết thúc khi nhấn phím *Enter*.

```
#include <stdio.h>
#include<conio.h>
int main ()
{
    char ch;
    ch = '~' ;
    while (ch!=13)
    {
       ch=getch();
       printf("%c - %d \n",ch, ch);
    }
    return 0;
}
```

Theo bảng mã *ASCII* (Bảng 1-2) *Enter* có mã là 13. Hàm *getch()* cho phép đọc một ký tự từ bàn phím và gán vào biến *ch*. Trong chương trình trên lệnh *ch='~';* có vẻ là không cần thiết, tuy nhiên trước khi thực hiện biểu thức *ch!=13* thì *ch* chưa xác định giá trị nên lệnh gán trước đó là cần thiết. Đây là tình huống không hay khi sử dụng lệnh *while* và chúng ta sẽ viết lại đẹp đẽ hơn khi sử dụng lệnh *do .. while* sau đây.

5. Lệnh do .. while

Cho phép thực hiện công việc nào đó lặp đi lặp lại một số lần.

Cú pháp:

```
do
        {Các lệnh}
while (Biểu_thức_điều_kiện)
```

Trong đó:

- *Biểu_thức_điều_kiện* là biểu thức điều kiện dùng để điều khiển vòng lặp.

Cách thực hiện:

8. Thực hiện các lệnh *{Các lệnh}*;
9. Tính giá trị *Biểu_thức_điều_kiện*, nếu giá trị nhận được khác không thì trở lại bước 1, nếu bằng không thì chuyển đến bước 3;
10. Kết thúc.

Ví dụ 1.17: Viết chương trình để in các số từ 1 đến 10 ra màn hình sử dụng vòng lặp *do .. while*.

```
#include <stdio.h>
#include<conio.h>
int main ()
{
    int i;
```

```
    printf("Day so tu 1 den 10 :\n");
    i=1;
    do
    {
        printf("%d \n",i);
        i++;
    } while (i<=10);
    getch();
    return 0;
}
```

Ví dụ này cho kết quả giống với ví dụ 1.11 và 1.14. Trong ví dụ này vòng lặp còn thực hiện khi *i<=10*, sau khi đã thực hiện lần đầu tiên của vòng lặp. Cũng như ví dụ 1.14 trong thân vòng lặp phải làm biến đổi *Biểu_thức_điều_kiện*, và việc khởi tạo của biến *i* trước khi thực hiện vòng lặp.

Ví dụ 1.18: Viết chương trình tính USCLN của hai số sử dụng lệnh *do .. while*.

```
#include <stdio.h>
#include<conio.h>
int main ()
{
    int a, b, m, n, tg;
    printf("Nhap vao cac he so a,b:");
    scanf("%d %d",&a,&b);
    m=a, n=b;
    do
    {
       tg = m % n;
       m=n;
       n=tg;
    } while (n>0);
    printf("USCLN cua %d va %d la: %d",a,b,m);
    getch();
    return 0;
}
```

Về cơ bản chương trình trên là giống với chương trình trong ví dụ 1.15. Điểm không hay của chương trình này so với ví dụ trước đó là phép chia *m%n* lần đầu không đảm bảo rằng *n<>0*. Ví dụ này cho thấy để giải bài toán tìm *USCLN* của 2 số thì sử dụng lệnh *while* sẽ thuận lợi hơn.

Ví dụ 1.19: Viết lại ví dụ 1.16 sử dụng lệnh *do .. while*.

```
#include <stdio.h>
#include<conio.h>
int main ()
{
    char ch;
    do
    {
          ch=getch();
       printf("%c - %d \n",ch, ch);
```

```
    }
    while (ch!=13)
    return 0;
}
```

Chương trình này rõ ràng là "*đẹp*" hơn chương trình tương tự trong ví dụ 1.16.

6. Lệnh break và continue

Trong phần trình bày về các câu lệnh lặp ở trên ta thấy việc kết thúc lặp chỉ diễn ra khi điều kiện lặp không còn thỏa mãn. Vấn đề là nếu muốn thoát ra khỏi vòng lặp sớm hơn thì có được không? Trong một số ngôn ngữ lập trình điều này là được phép, và trong C/C++ đó là lệnh *break* hoặc *continue*.

Lệnh break: Dùng để kết thúc sự thực hiện của một trong các lệnh *do, for, switch* hoặc *while* chứa nó, sau đó điều khiển được chuyển đến câu lệnh kế tiếp.

Cú pháp:

```
break;
```

Lệnh *break* thường được dùng để kết thúc việc xử lý một nhánh nào đó của lệnh *switch*, và nếu thiếu các lệnh *break* trong lệnh này thường dẫn đến lỗi chương trình.

Trong các lệnh lặp, lệnh *break* chỉ kết thúc duy nhất một câu lệnh *do, for*, hoặc *while* trực tiếp chứa nó.

Lệnh continue: Lệnh này dùng để quay lại đầu vòng lặp mà không thực hiện các lệnh trong khối lệnh lặp (dạng *for*, *do* hay *while*) kể từ sau lệnh *continue*.

Cú pháp:

```
continue;
```

Vòng lặp tiếp theo được xác định như sau:

- Trong vòng lặp *do* và *while*, vòng lặp tiếp theo được bắt đầu bằng cách tính lại biểu thức điều khiển của câu lệnh.
- Với vòng lặp *for* (dạng *for(i; c; e)*) thì lệnh *continue* sẽ cho phép thực hiện công việc của *e*, sau đó tính lại *c* và tùy thuộc vào kết quả đúng hay sai mà vòng lặp được tiếp tục hay không.

7. Cấu trúc chương trình

Một chương trình hoàn chỉnh trong C/C++ có 6 phần chính (nhưng không bắt buộc) theo thứ tự như sau:

- Chỉ thị tiền xử lý;
- Định nghĩa kiểu dữ liệu;
- Khái báo prototype;
- Khai báo biến ngoài;
- Chương trình chính và
- Cài đặt hàm.

Kết quả: Chương trình **QLSV** chạy được theo các yêu cầu đặt ra.