BỘ GIÁO DỤC VÀ ĐÀO TẠO

ĐẠI HỌC CÔNG NGHỆ TP.HCM

THỰC HÀNH NGÔN NGỮ LẬP TRÌNH C

Biên Soạn:

ThS. Nguyễn Thúy Loan

www.hutech.edu.vn

THỰC HÀNH NGÔN NGỮ LẬP TRÌNH C

Các ý kiến đóng góp về tài liệu học tập này, xin gửi về e-mail của ban biên tập: tailieuhoctap@hutech.edu.vn

MỤC LỤC

HƯỚNG DẪN

Môn Thực hành Kỹ thuật Lập trình cung cấp cho sinh viên những kỹ năng thực hành cơ bản về ngôn ngữ lập trình C. Môn học này là nền tảng để tiếp thu hầu hết các môn học khác trong chương trình đào tạo. Mặt khác, nắm vững môn này là cơ sở để phát triển tư duy và kỹ năng lập trình để giải các bài toán và các ứng dụng trong thực tế.

Học xong môn này, sinh viên phải nắm được các vấn đề sau:

- Tổng quan về Ngôn ngữ lập trình C.
- Các kiểu dữ liệu trong C.
- Các lệnh có cấu trúc.
- Cách thiết kế và sử dụng các hàm trong C.
- Xử lý các bài toán trên mảng một chiều.
- Biết xây dựng và xử lý các bài toán trên dữ liệu có cấu trúc do người dùng định nghĩa.
- Cách lưu trữ và xử lý các file trong C.

NỘI DUNG MÔN HỌC

- Bài 1: Làm quen với chương trình C.
- Bài 2: Cấu trúc điều khiển – cấu trúc IF và SWITCH
- Bài 3: Cấu trúc điều khiển – cấu trúc lặp
- Bài 4: Chương trình con
- Bài 5: Mảng
- Bài 6: Kiểu dữ liệu có cấu trúc

YÊU CẦU MÔN HỌC

Có tư duy tốt về logic và kiến thức toán học cơ bản.

CÁCH TIẾP NHẬN NỘI DUNG MÔN HỌC

Thực hành Kỹ thuật lập trình là môn học đầu tiên giúp sinh viên làm quen với phương pháp lập trình trên máy tính, giúp sinh viên có khái niệm cơ bản về cách tiếp cận và giải quyết các bài toán tin học, giúp sinh viên có khả năng tiếp cận với cách tư duy của người lập trình, và là tiền đề để tiếp cận với các học phần quan trọng còn lại của ngành Công nghệ Thông tin. Vì vậy, yêu cầu người học phải dự học đầy đủ các buổi thực hành tại lớp, thực hành lại các bài tập ở nhà, nghiên cứu tài liệu trước khi đến lớp và gạch chân những vấn đề không hiểu khi đọc tài liệu để đến lớp trao đổi.

PHƯƠNG PHÁP ĐÁNH GIÁ MÔN HỌC

Để học tốt môn này, người học cần ôn tập các bài đã học, trả lời các câu hỏi và làm đầy đủ bài tập; đọc trước bài mới và tìm thêm các thông tin liên quan đến bài học.

Đối với mỗi bài học, người học đọc trước phần tóm tắt kiến thức lý thuyết, sau đó thực hành các bài từ cơ bản đến nâng cao. Lưu ý xem kỹ hướng dẫn trong mỗi bài thực hành và làm theo. Kết thúc mỗi bài học, học viên cần làm lại các bài thực hành ở nhà và luyện tập thêm.

Điểm đánh giá : gồm 2 phần

1. Điểm quá trình (chuyên cần + điểm bài thực hành hàng tuần).
2. Điểm cuối kỳ (bài thi thực hành trên máy).

BÀI 1: LÀM QUEN VỚI CHƯƠNG TRÌNH C

Sau khi thực hành xong bài này, sinh viên có thể nắm được:

- *Làm quen với cấu trúc chung của một chương trình C đơn giản.*
- *Các lệnh nhập, xuất dữ liệu (printf, scanf).*
- *Các kiểu dữ liệu chuẩn (int, long, char, float, …).*
- *Các phép toán và các hàm chuẩn của ngôn ngữ lập trình C.*
- *Thực hiện viết các chương trình hoàn chỉnh sử dụng các lệnh đơn giản.*

1.1 TÓM TẮT LÝ THUYẾT

1.1.1 Các ký hiệu

Ký hiệu	Diễn giải	Ví dụ

{}	Bắt đầu và kết thúc hàm hay khối lệnh.	int main()
		{
		}

;	Kết thúc khai báo biến, một lệnh, một lời gọi hàm, hay khai báo nguyên mẫu hàm.	int x; void NhapMang(int a[], int &n);
//	Chú thích (ghi chú) cho một dòng. Chỉ có tác dụng đối với người đọc chương trình.	//Ham nay dung de nhap mang void NhapMang(int a[], int &n);
/* */	Tương tự như ký hiệu //, nhưng cho trường hợp nhiều dòng.	/* Dau tien nhap vao n. Sau do nhap cho tung phan tu */ void NhapMang(int a[], int &n);

1.1.2 Các kiểu dữ liệu cơ bản trong C

Kiểu	Ghi chú	Kích thước	Định dạng
Kiểu liên tục (số thực)			
Float		4 bytes	%f
double		8 bytes	%f
long double		10 bytes	%lf
Kiểu rời rạc (số nguyên)			
Char	Ký tự Số nguyên	1 byte 1 byte	%c %d
unsigned char	Số nguyên dương	1 byte	%d
Int	Số nguyên	2 bytes	%d
unsigned int	Số nguyên dương	2 bytes	%u
Long	Số nguyên	4 bytes	%ld
unsigned long	Số nguyên dương	4 bytes	%lu

1.1.3 Bảng ký hiệu và phép toán

Phép toán	Ý nghĩa	Ghi chú
+,-,*,/ %	Cộng, trừ, nhân, chia Chia lấy phần dư	Ví dụ: 5%2=1
>, >=, <, <=, == !=	Lớn hơn, lớn hơn hoặc bằng, nhỏ hơn, nhỏ hơn hoặc bằng Bằng nhau Khác nhau	
! && ||	NOT AND OR	
++ --	Tăng 1 Giảm 1	Nếu toán tử tăng giảm đặt trước thì tăng giảm trước rồi tính biểu thức hoặc ngược lại.

1.1.4 Lệnh nhập, xuất dữ liệu cơ bản

Cú pháp	Diễn giải	Ví dụ
scanf("mã định dạng", địa chỉ của các biến);	Nhập dữ liệu: lấy dữ liệu nhập từ bàn phím lưu vào biến	int x; float y; scanf("%d%f", &x, &y);
printf("Chuỗi định dạng", Các biểu thức);	Xuất 1 thông báo ra màn hình. Xuất giá trị của các biểu thức lên màn hình.	printf("Day la chuong trinh vi du minh hoa\n"); printf("Gia tri cua so vua nhap la %d", x);

1.1.5 Cấu trúc rẽ nhánh if

```
if (biểu thức điều kiện)
        công việc 1;
else
        công việc 2;
```

❖ *Lưu ý:*

- *Biểu thức điều kiện phải đặt trong cặp dấu ngoặc ( ).*
- *Công việc cần thực hiện có thể gồm 1 hay nhiều lệnh. Nếu gồm nhiều lệnh con thì các lệnh con phải được gom vào trong cặp dấu {} ➔ gọi là khối lệnh.*

1.2 THỰC HÀNH CƠ BẢN

Câu 1: Làm quen với cấu trúc chung của một chương trình C đơn giản:

Mở C-free/Dev-C, vào File/New/Source file.

Gõ đoạn code sau và lưu file với phần mở rộng là .cpp (Ví dụ: Cau1.cpp)

```
//Khai báo thư viện stdio.h vì nó chứa hàm printf.
#include<stdio.h>
int main()
{
        printf("Hello, World!");
```

```
        printf("Hello, World!\n");
        printf("Hello, \nWorld!\n");
        printf("Hello, \tWorld!\n");
    return 0;
    }
```

Hãy biên dịch và chạy chương trình, xem kết quả hiện ra màn hình và rút ra nhận xét.

Phím tắt để biên dịch và chạy chương trình (Compile and Run):

- C-free: F5
- Dev-C: F11

Câu 2: Viết chương trình in lên màn hình một thiệp mời dự sinh nhật có dạng:

```
****************************************
    THIEP MOI
    Than moi ban: "Le Loi"
    Toi du le sinh nhat cua minh
    Vao luc 19h ngay 20/10/2016
    Tai: 05/42 Vinh Vien – TP. HCM
    Rat mong duoc don tiep!
    Ho Le Thu
****************************************
```

- *Hướng dẫn:* Áp dụng \t (tab), \n (xuống dòng), \" (in ra dấu ")

Câu 3: Viết chương trình thực hiện:

a. Nhập một số nguyên. Xuất ra màn hình số nguyên vừa nhập.

b. Nhập một số thực. Xuất ra màn hình số thực vừa nhập.

c. Nhập một kí tự. Xuất ra màn hình kí tự vừa nhập.

d. Nhập hai số nguyên. Hãy tính tổng, hiệu, tích, thương của hai số đó và xuất kết quả ra màn hình.

```
#include<stdio.h>
int main()
{
        //câu a
        int a; printf("Moi nhap 1 so nguyen: "); scanf("%d", &a);
        printf("So nguyen da nhap la %d", a);
        //câu b tương tự
        float b ; printf("Moi nhap 1 so thuc: "); scanf("%f ", &b);
        printf("So thuc da nhap la %f", b);
        //câu c
        char z;
        printf("Moi nhap 1 kí tự: ");
        fflush(stdin); //xoá bộ nhớ đệm
        scanf("%c", &z);
        printf("Ki tu da nhap la %c", z);
        //câu d
        int x, y, z;
        printf("Moi nhap hai so nguyen de cong: ");
        scanf("%d%d", &x, &y);
        z = x + y;
        printf("Tong hai so la %d\n", z);
        //hoặc printf("Tong hai so la %d\n", x+y);
        /*tương tự với phép hiệu, tích, thương. Lưu ý kiểu dữ liệu kết quả của phép chia */
        return 0;
}
```

Câu 4: Viết chương trình nhập vào bán kính *r* của một hình tròn. Tính chu vi và diện tích của hình tròn. In các kết quả lên màn hình.

Hướng dẫn:

Khai báo biến r để lưu trữ bán kính của hình tròn

Khai báo biến: **Kiểu_ dữ_ liệu Tên_biến;** VD: float r;

Khai báo hằng số PI = 3.14

Cách 1: **#define Tên_hằng Giá_trị** VD: #define MAX 100

Cách 2: **const Kiểu_dữ_liệu Tên_hằng=Giá trị;**

VD: const int MAX=100;

Diện tích hình tròn: PI*r*r

Chu vi hình tròn: 2*PI*r.

Câu 5: Viết chương trình thực hiện:

a. Nhập vào hai số nguyên. Xuất ra màn hình giá trị lớn nhất.

b. Nhập vào ba số nguyên. Xuất ra màn hình giá trị lớn nhất.

Hướng dẫn:

❖ Nhập vào hai số nguyên. Xuất ra màn hình giá trị lớn nhất.

- Khai báo hai biến *a, b* để lưu hai số nguyên.
- Thông báo và cho người dùng nhập 2 số nguyên *a* và b (dùng printf, scanf).
- Sử dụng một biến *max* để lưu giá trị lớn nhất trong hai số, khai báo *max*.
- So sánh hai số *a* và *b*, nếu số nào lớn hơn thì gán max = số đó.
- Xuất ra màn hình giá trị lớn nhất tìm được: xuất giá trị biến *max*

❖ Nhập vào ba số nguyên. Xuất ra màn hình giá trị lớn nhất.

- Tương tự, khai báo 3 biến *a, b, c* và *max* lưu giá trị lớn nhất.
- Giả sử *a* là số lớn nhất, tức *max* = *a*.
- Lần lượt so sánh hai số còn lại *b, c* với *max*, nếu số nào lớn hơn *max* thì cập nhật *max* = số đó như sau: Nếu b>max thì max=b. Nếu c>max thì max=c.
- Xuất ra màn hình giá trị lớn nhất tìm được: xuất giá trị biến *max.*

1.3 THỰC HÀNH NÂNG CAO

Câu 6: Nhập vào 3 số nguyên dương *a, b, c*. Kiểm tra xem 3 số đó có lập thành tam giác không? Nếu có hãy tính chu vi và diện tích của tam giác theo công thức:

- Chu vi CV = a+b+c.
- Diện tích S = sqrt(p*(p-a)*(p-b)*(p-c)), trong đó: p =CV/2.
- Xuất các kết quả ra màn hình.

Hướng dẫn:

- *a, b, c* là số nguyên dương => khai báo *a, b, c* kiểu *unsigned int*.
- Điều kiện để 3 số lập thành tam giác: tổng 2 cạnh phải lớn hơn cạnh còn lại. Vậy: *a, b, c* lập thành 3 cạnh của tam giác khi và chỉ khi *a+b>c và a+c>b và b+c>a.*
- Tính diện tích và chu vi theo công thức đã cho.
- Hàm căn bậc hai *sqrt(x)* trong thư viện *<math.h>*.

Câu 7: Viết chương trình đảo ngược một số nguyên dương có đúng 3 chữ số.

VD: Nhập vào n=234 ➔ In ra: 432

Hướng dẫn:

- Lần lượt lấy các chữ số (sử dụng phép chia / và phép chia lấy phần dư %) và in ra màn hình theo thứ tự:
  - Chữ số hàng đơn vị
  - Chữ số hàng chục
  - Chữ số hàng trăm.

Ví dụ: 234 %10 =4

234 /10 =23

23 %10 =3

23/10=2

2%10=2

2/10=0 -> dừng

BÀI 2: CẤU TRÚC ĐIỀU KHIỂN – CẤU TRÚC IF VÀ SWITCH

Sau khi thực hành xong bài này, sinh viên có thể nắm được:

- *Nắm vững cú pháp, cách thức hoạt động của cấu trúc rẽ nhánh if và if … else.*
- *Nắm vững cú pháp, cách thức hoạt động của cấu trúc switch*

2.1 TÓM TẮT LÝ THUYẾT

2.1.1 Cấu trúc if

Cú pháp:

```
if (biểu thức điều kiện)
        công việc;
```

Ý nghĩa: Nếu biểu thức điều kiện cho kết quả đúng (khác không) thì thực hiện công việc.

2.1.2 Cấu trúc if …else

Cú pháp:

```
if (biểu thức điều kiện)
        công việc 1;
else
        công việc 2;
```

Ý nghĩa: Nếu biểu thức điều kiện cho kết quả đúng (khác không) thì thực hiện công việc 1, ngược lại thì thực hiện công việc thứ 2.

❖ *Lưu ý:*

- Biểu thức điều kiện phải đặt trong cặp dấu ngoặc ( ).
- Công việc cần thực hiện có thể gồm 1 hay nhiều lệnh. Nếu gồm nhiều lệnh con thì các lệnh con phải được gom vào trong cặp dấu {} ➔ gọi là khối lệnh.

2.1.3 Cấu trúc switch

Cú pháp:

```
switch (bieu_thuc)
{
    case gia_tri_1:
            các câu lệnh;
    break;      /*lệnh break giúp chương trình thoát khỏi lệnh switch sau khi thực
                hiện xong một trường hợp*/
    ...
    case gia_tri_n:
            các câu lệnh;
    break;
    [default: cv thực hiện mặc định nếu biểu thức không có giá trị nào trong các
    case] //có thể có hoặc không tuỳ bài toán
}
```

Gia_tri_i: là các hằng số nguyên hoặc ký tự.

Phụ thuộc vào giá trị của biểu thức viết sau switch, nếu:

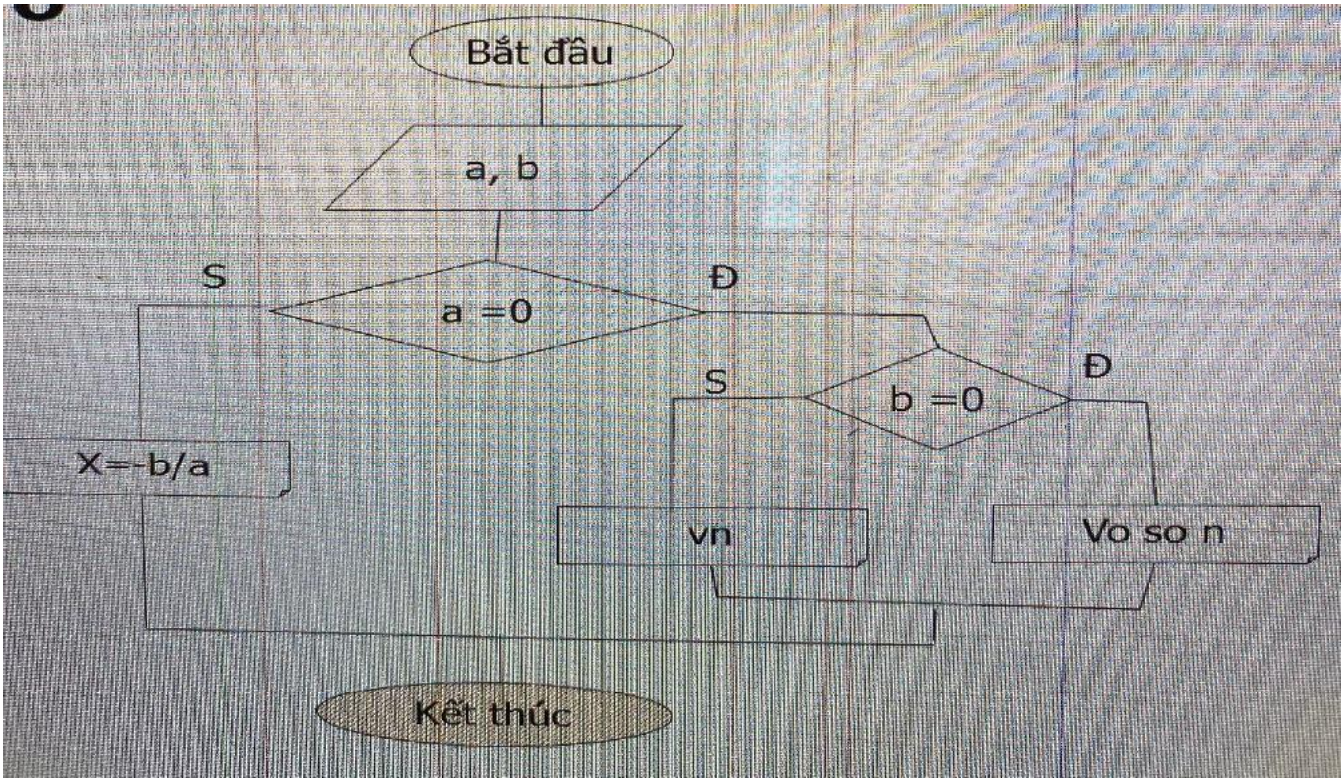
- Giá trị này = *gia_tri_i* thì thực hiện câu lệnh sau case *gia_tri_i*.
- Khi giá trị biểu thức không thỏa tất cả các *gia_tri_i* thì thực hiện câu lệnh sau default nếu có, hoặc thoát khỏi câu lệnh switch.
- Khi chương trình đã thực hiện xong câu lệnh của case *gia_tri_i* nào đó thì nó sẽ thực hiện luôn các lệnh thuộc case bên dưới nó mà không xét lại điều kiện (do các *gia_tri_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**.

2.2 THỰC HÀNH CƠ BẢN

Câu 1: Viết chương trình giải phương trình bậc nhất *ax + b = 0* với *a, b* nhập từ bàn phím.

Hướng dẫn:

- Khai báo hai biến *a, b* kiểu số thực để lưu hệ số của phương trình do người dùng nhập tùy ý từ bàn phím. Nhập *a, b.*
- Giải thuật: Xét đủ 2 trường hợp *a = 0* và *a ≠ 0*.
- Hãy vận dụng lệnh if và if.. else để cài đặt giải thuật trên sao cho chương trình rõ ràng nhất.




```
#include <stdio.h>
#include< conio.h>
int main()
{
  float a,b;
```

```
    nhap a;
    nhap b;
    if ( a ==0 )
        if ( b==0 )
            printf ( "   ");
        else
            printf ( " " );
    else
        printf ( " Phuong trinh có nghiem x= %f ", -b/a );
     return 0;
}
```

Câu 2: Viết chương trình giải phương trình bậc hai $ax^2 + bx + c = 0$. *Với a, b, c* nhập từ bàn phím.

Hướng dẫn:

- Khai báo 3 biến *a, b, c* kiểu số thực. Nhập các hệ số *a, b, c.*
- Xét $a = 0$. Phương trình trở thành bậc nhất, giải và biện luận theo *b, c* (tương tự bài 1).
- Xét $a \neq 0$. Tính delta, *d = b*b-4*a*c*. Xét các trường hợp $d = 0$, $d < 0$ và $d > 0$.
- Nếu d<0 thì phương trình vô nghiệm.
- Nếu d=0 thì phương trình có nghiệm kép x1 = x2 = -b/(2*a)
- Nếu d>0 thì phương trình có 2 nghiệm phân biệt
- Hàm tính căn bậc hai *sqrt(số)* nằm trong thư viện *<math.h>.*

Câu 3: 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, …).

Hướng dẫn:

- *a, b, c* là số nguyên dương => khai báo *a, b, c* kiểu *unsigned int* . Chuỗi định dạng của kiểu này là *%u.*

- Điều kiện để 3 số lập thành tam giác: tổng 2 cạnh phải lớn hơn cạnh còn lại. Vậy: *a, b, c* lập thành 3 cạnh của tam giác $\Leftrightarrow a+b> c$ và $a+c>b$ và $b+c>a$.
- Xét loại tam giác:
  - Tam giác cân $\Leftrightarrow a = b$ hoặc $b = c$ hoặc $c = a$.
  - Tam giác đều $\Leftrightarrow a=b=c$.
- Tam giác vuông $\Leftrightarrow a^2=b^2+c^2$ hoặc $b^2=a^2+c^2$ hoặc $c^2=a^2+b^2$.

Câu 4: Viết chương trình nhập số nguyên có hai chữ số, hiển thị cách đọc số đó.

Hướng dẫn:

- Khai báo và nhập số nguyên *n*.
- Nếu $9 < n < 100$ thì
  - Thực hiện đọc số *n*.

  Ngược lại thì xuất ra thông báo "số nhập vào không phải là số có hai chữ số".

  Cách đọc số *n*:
  - Lấy chữ số hàng chục *n/10*. Sử dụng câu lệnh *switch* để đọc chữ số hàng chục
  - Lấy chữ số hàng đơn vị *n%10*. Sử dụng câu lệnh *switch* để đọc chữ số hàng đơn vị.

Câu 5: Viết chương trình nhập vào tháng của một năm, cho biết số ngày của tháng đó. Nếu tháng nhập vào <1 hoặc >12 thì thông báo "Không tồn tại tháng này".

Hướng dẫn:

- Khai báo biến và nhập vào tháng.
- Các tháng 1, 3, 5, 7, 8, 10, 12 có 31 ngày
- Các tháng 4, 6, 9, 11 có 30 ngày
- Nếu là tháng 2 thì yêu cầu nhập thêm năm, nếu là năm nhuận thì tháng 2 có 29 ngày, còn lại là 28 ngày.
- Năm nhuận là năm chia hết cho 4 (hoặc những năm có 2 số cuối là số 0 thì các bạn lấy số năm chia cho 400, nếu chia hết thì năm đó là năm có nhuận)

- Nếu tháng nhập vào không thuộc các tháng trên thì thông báo "không tồn tại tháng này".

2.3 THỰC HÀNH NÂNG CAO

Câu 6: Nhập vào ngày, tháng, năm. Kiểm tra xem ngày, tháng, năm đó có hợp lệ hay không? In kết quả ra màn hình.

Câu 7: Viết chương trình tính tiền cước TAXI. Biết rằng:

- KM đầu tiên là 5000đ.
- 200m tiếp theo là 1000đ.
- Nếu lớn hơn 30km thì mỗi km thêm sẽ là 3000đ.
- Hãy nhập số km sau đó in ra số tiền phải trả.

BÀI 3: CẤU TRÚC ĐIỀU KHIỂN - CẤU TRÚC LẶP

Sau khi thực hành xong bài này, sinh viên có thể nắm được:

- *Nắm vững cú pháp, cách thức hoạt động của cấu trúc lặp for.*
- *Nắm vững cú pháp, cách thức hoạt động của cấu trúc lặp while.*
- *Nắm vững cú pháp, cách thức hoạt động của cấu trúc lặp do ... while.*

3.1 TÓM TẮT LÝ THUYẾT

3.1.1 Cấu trúc for

Cú pháp:

```
for (biểu thức khởi gán; biểu thức điều kiện; biểu thức tăng/giảm)
{
    khối lệnh thực hiện công việc;
}
```

Ý nghĩa:

Bước 1: Khởi gán cho biểu thức 1

Bước 2: Kiểm tra điều kiện của biểu thức 2.

- Nếu biểu thức 2 ≠ 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.

3.1.2 Cấu trúc while

Cú pháp:

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

Ý nghĩa: Cách hoạt động của while giống for.

3.1.3 Cấu trúc do ... while

Cú pháp:

```
do
{
    Khối lệnh thực hiện công việc;
} while (biểu thức điều kiện) ;
```

Ý nghĩa: Thực hiện khối lệnh cho đến khi biểu thức điều kiện có giá trị sai (có giá trị bằng 0) thì dừng.

❖ **Lưu ý:**

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.

3.2 THỰC HÀNH CƠ BẢN

Câu 1: Viết chương trình thực hiện:

a. Xuất ra màn hình 10 dòng: "XIN CHAO CAC BAN".

b. Xuất ra màn hình *n* dòng: ""XIN CHAO CAC BAN", với *n* nhập từ bàn phím.

Hướng dẫn:

a. Xác định số lần lặp: 10 lần. Xác định công việc lặp: xuất câu thông báo "XIN CHAO CAC BAN" ra màn hình. Đoạn code sau minh hoạ cách dùng các vòng lặp for, while, do … while.

```
//Cách 1: Sử dụng vòng lặp for
int i;
for(i=1; i<=10; i++)
      printf("XIN CHAO CAC BAN\n");
//Cách 2: Sử dụng vòng lặp while--------------------------------------
i=1;
while (i<=10)
{
      printf("XIN CHAO CAC BAN\n");
      i=i+1; //hoặc i++; hoặc i+=1;
}
//Cách 3: Sử dụng vòng lặp do … while---------------------------------
i=1;
do{
      printf("XIN CHAO CAC BAN\n");
      i=i+1;
}while (i<=10);
```

Câu 2: Viết chương trình nhập vào một số nguyên *n > 0*, hãy:

a. Xuất ra màn hình các số trong phạm vi từ 1 đến *n*.

b. Xuất ra màn hình các số chẵn trong phạm vi từ 1 đến *n*.

c. Xuất ra màn hình các số lẻ không chia hết cho 3 trong phạm vi từ 1 đến *n.*

d. Tính các biểu thức sau:

- S1 = 1 + 2 + ... + *n*
- S2 = - 1 +2 - 3 + 4 - ... + $(-1)^n$ *n*.
- S3 = 1/2 + 2/3 + 3/4 ... + *n/(n+1)*
- S4 = x^n (*x* là số thực nhập từ bàn phím).

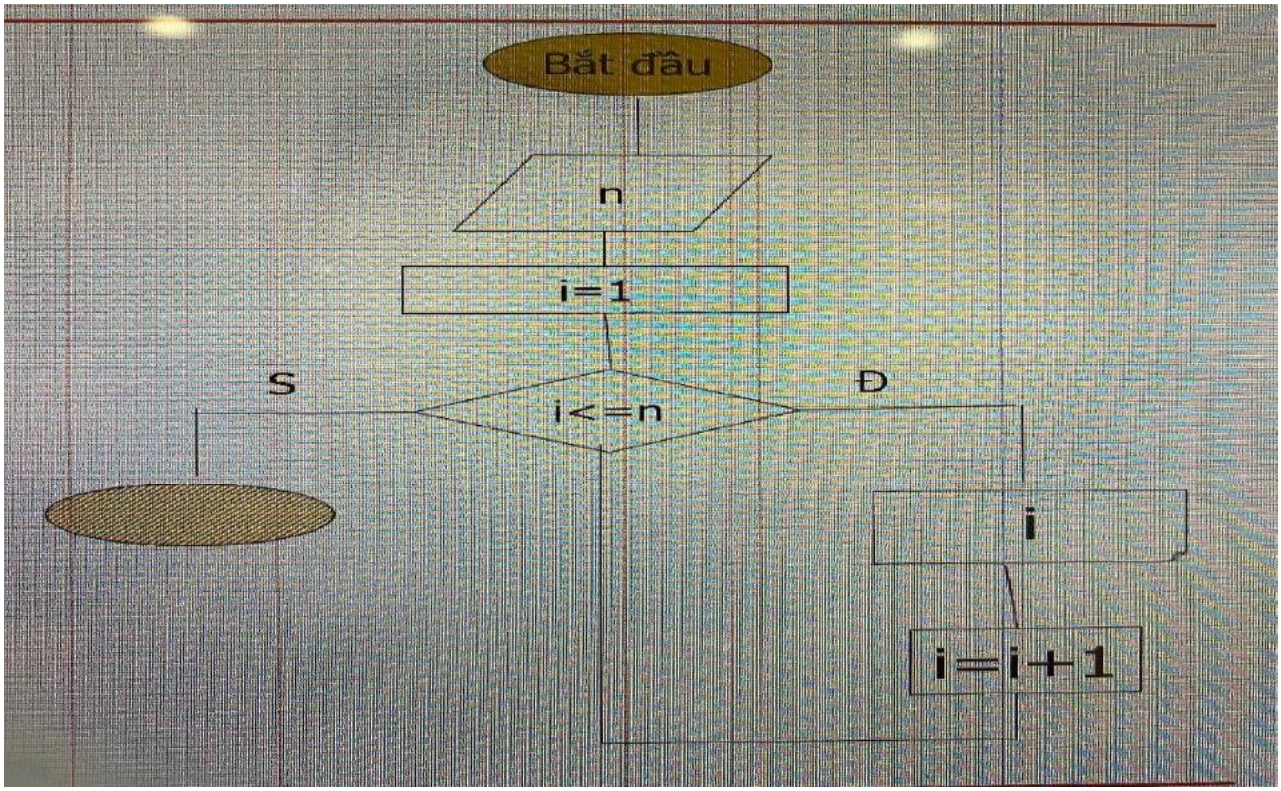
e. Tính tổng các chữ số của n. Ví dụ: n = 125, tổng các chữ số là 8.

Hướng dẫn:

Nhập số nguyên *n > 0*, nếu nhập sai thì bắt nhập lại.

```
do {
        printf (" Nhập vào số phần tử " );
        scanf( " %d" , &n);
        if(n<=0)
            printf( " nhap sai, nhap lai ");
   }while ( n<=0 );
```

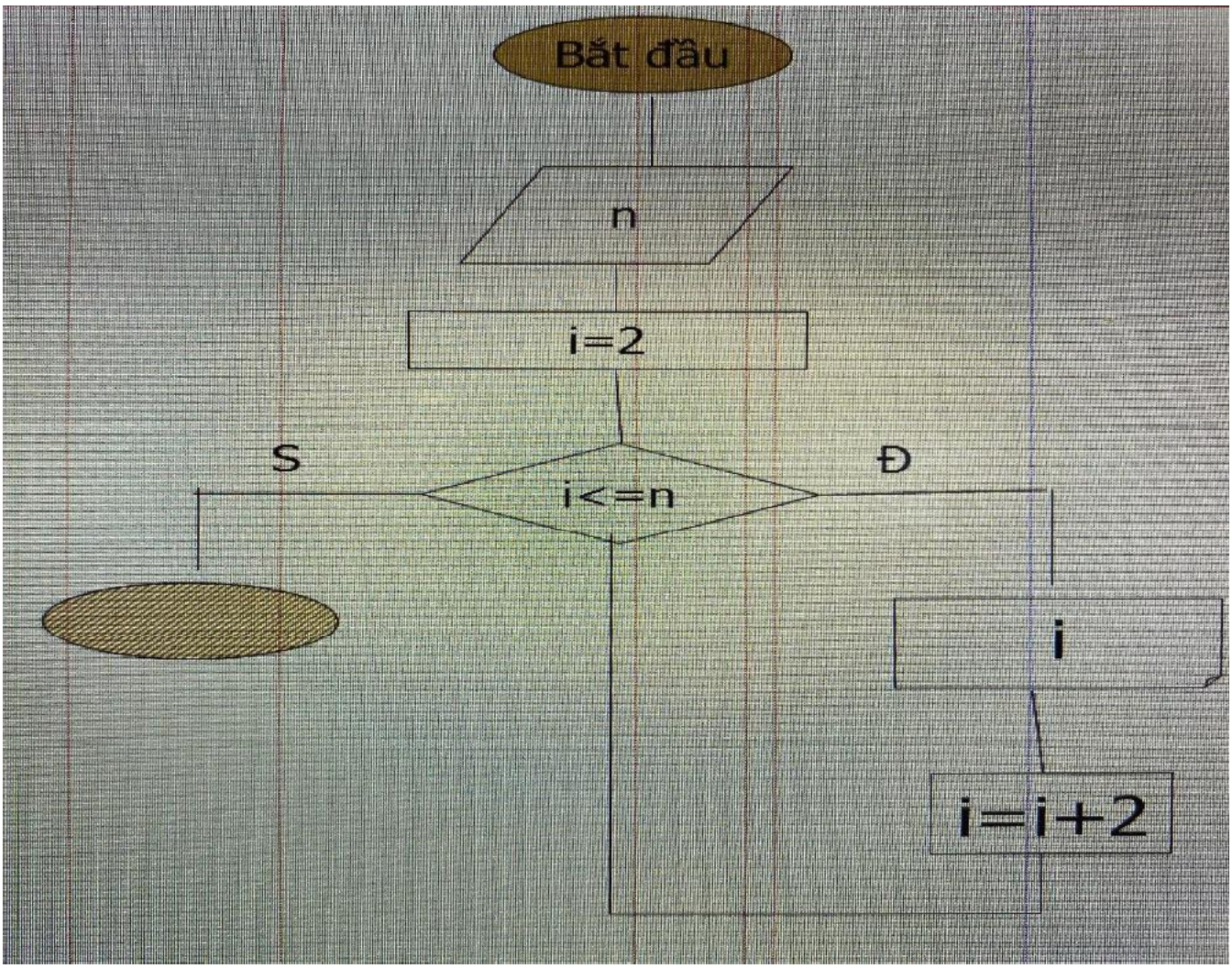
a. Xuất ra màn hình các số trong phạm vi từ 1 đến n.



- Câu a, *b, c, d* có thể áp dụng cấu trúc lặp for/while.

  Hướng dẫn câu b

b. Xuất ra màn hình các số chẵn trong phạm vi từ 1 đến n.

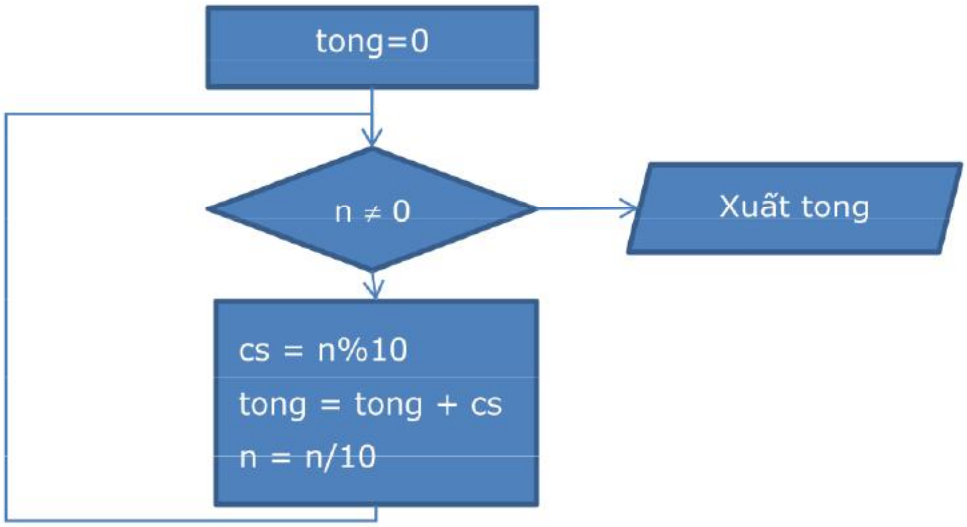


e. Tính tổng các chữ số của n

❖ Lần lượt lấy từng chữ số của số *n* và cộng dồn vào biến tổng.

- Giả sử, khai báo biến lưu tổng các chữ số là ***tong***, biến lưu từng chữ số của *n* là ***cs***.

- Thuật toán như sau:



Câu 3: Viết chương trình thực hiện:

a. Nhập một số nguyên n sao cho *0 < n ≤ 100*. Nếu nhập sai thì yêu cầu nhập lại.

b. Đếm số ước của *n*. Nếu đếm bằng 2 thì xuất ra màn hình *"n là số nguyên tố"*, ngược lại xuất ra *"n không phải là số nguyên tố"*.

c. Tính tổng các ước của n (không tính chính nó). Nếu tổng các ước đúng bằng *n* thì xuất ra màn hình *"n là số hoàn thiện"*, ngược lại xuất ra *"n không là số hoàn thiện"*.

Hướng dẫn:

- ***Điều kiện nhập lại n: n ≤ 0 hoặc n > 100.***

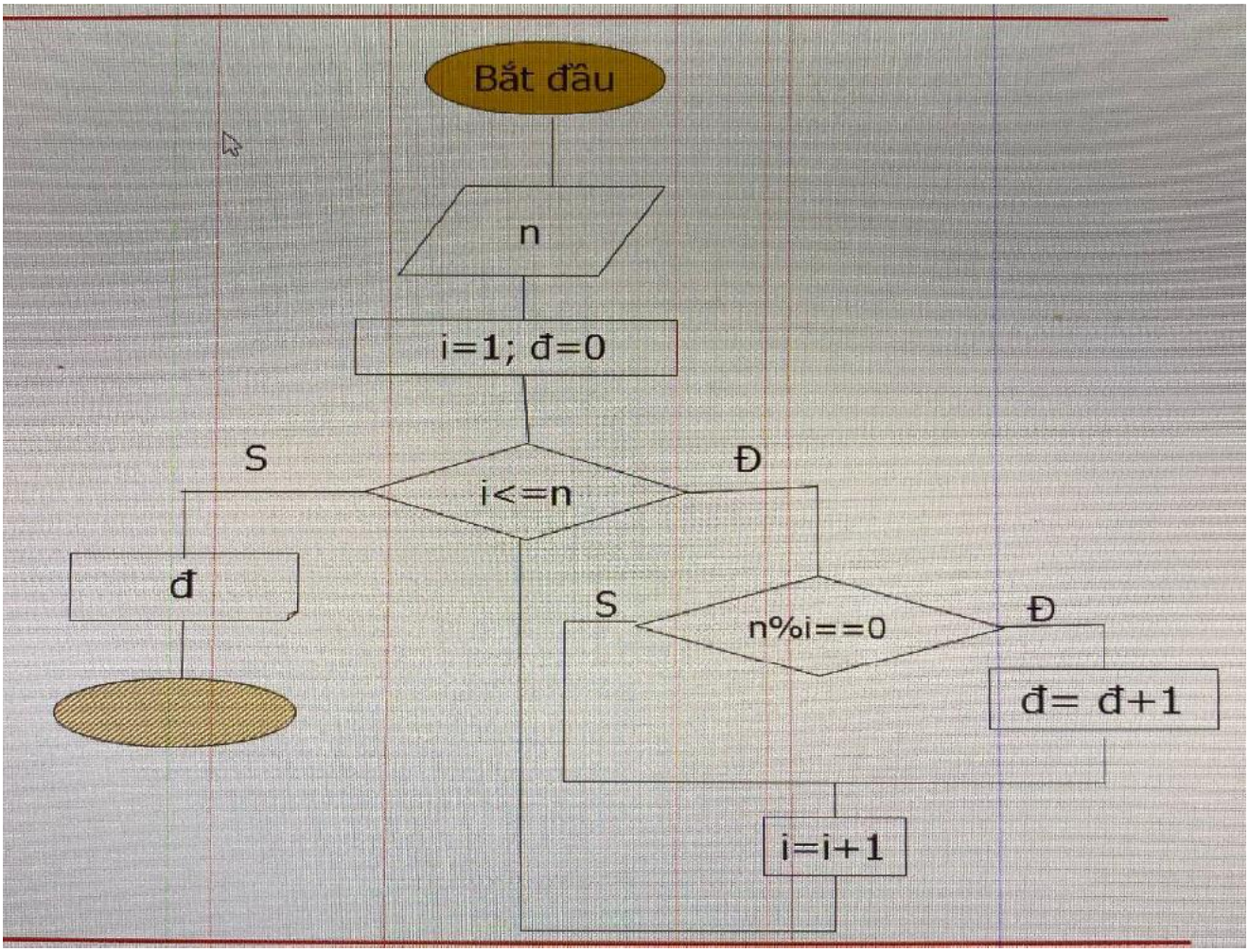
```
do {
        printf (" Nhập vào số phần tử " );
        scanf( " %d" , &n);
        if ( ….. )
            printf ( " nhap sai, nhap lai ");
}while ( n<=0 || n > = 100 );
```

- Sử dụng phép chia lấy dư (%) để xác định số nào là ước của *n.*

a. Đếm số ước của n

Khai báo biến đếm là d, đầu tiên gán d=0;

Lần lượt lấy n chia lấy phần dư (%) cho các số từ 1 đến n, nếu số nào có số dư là 0 thì ta tăng biến đếm lên 1 đơn vị -> dùng vòng lặp



Câu 4: Viết chương trình hiển thị ra màn hình *n* dòng (*0 < n ≤ 10*):

a.

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

b.

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

Cụ thể:

a. Hiển thị trên *n* dòng, mỗi dòng có *k* dấu *, *k* tùy ý người dùng nhập.

b. Hiển thị trên *n* dòng, dòng thứ *i* có *i* dấu *.

Hướng dẫn:

- Câu a: Sử dụng 2 vòng for lồng nhau, vòng for thứ nhất để lặp xét từng dòng (lặp *n* lần), vòng for thứ hai để lặp xuất ra dấu * theo số lượng yêu cầu (lặp *k* lần).
- Câu b: Sử dụng 2 vòng for lồng nhau, vòng for thứ nhất để lặp xét từng dòng (lặp *n* lần), vòng for thứ hai để lặp xuất ra dấu * theo số lượng yêu cầu (lặp *i* lần).

3.3 THỰC HÀNH NÂNG CAO

Câu 5: Viết chương trình hiển thị bảng cửu chương ra màn hình.

Câu 6: Viết chương trình thực hiện:

a. Nhập *n* > 0.

b. Liệt kê các số nguyên tố trong phạm vi từ 1 đến *n*.

c. Đếm số lượng số nguyên tố trong phạm vi từ 1 đến *n.*

d. Tính tổng các số nguyên tố trong phạm vi từ 1 đến *n*.

Hướng dẫn

b. Liệt kê các số nguyên tố trong phạm vi từ 1 đến *n*.

```
for ( int i=2 ; i<=n ; i++ )
  kiểm tra nto ( i ) nếu đúng thì in ra số i ;
```

c. Đếm số lượng các số nguyên tố trong phạm vi từ 1 đến n.

```
int d=0;
for ( int i=2 ; i<=n ; i++ )
  kiểm tra nto ( i ) nếu đúng là nto  thì tăng biến đếm lên 1 đơn vị ;
```

Câu 7: Viết chương trình thực hiện:

a. Nhập *n* > 0.

b. Tính tổng các chữ số của *n*. Ví dụ: với *n* = 12537 thì tổng = 1+2+5+3+7=18.

BÀI 4: CHƯƠNG TRÌNH CON

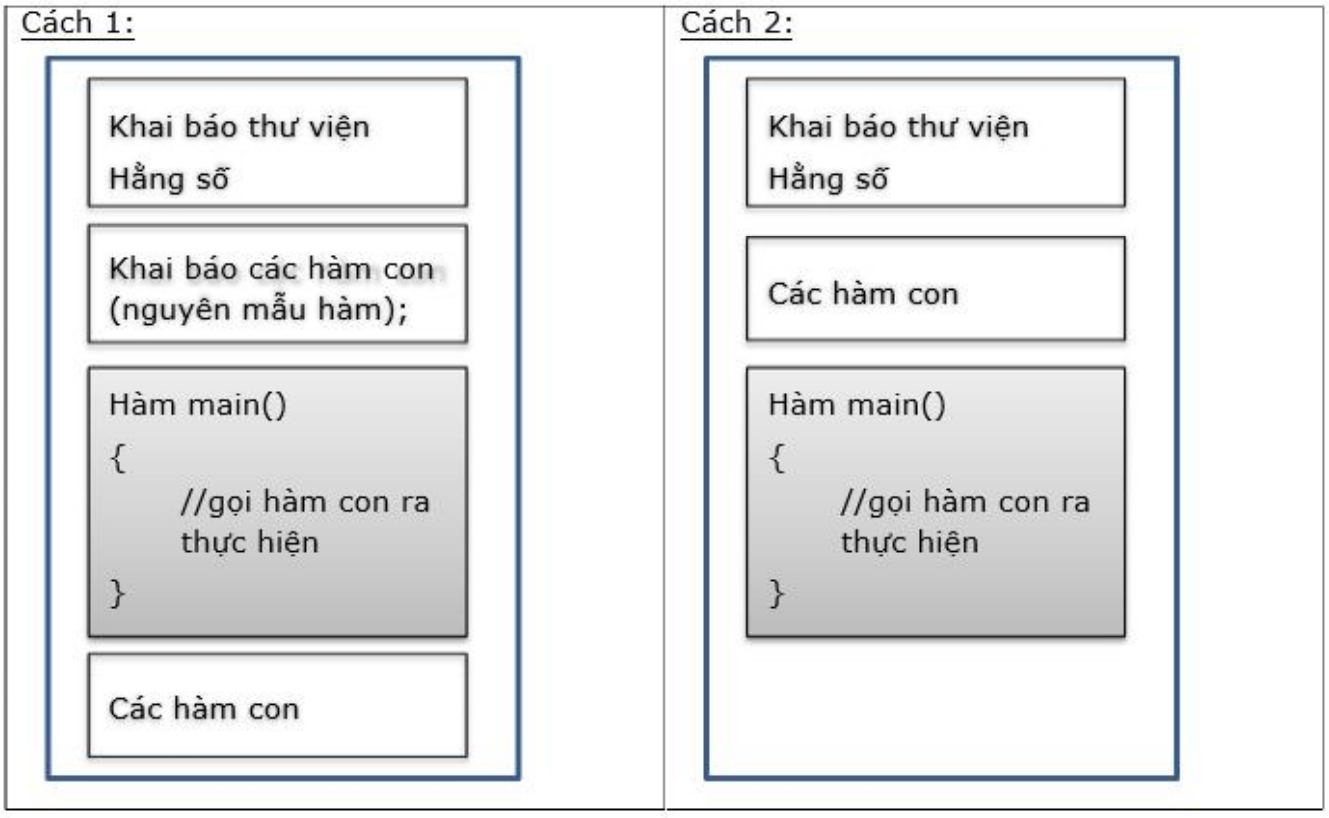
Sau khi thực hành xong bài này, sinh viên có thể nắm được:

- *Khái niệm về hàm (function) trong C.*
- *Cách xây dựng và cách sử dụng hàm trong C.*

4.1 TÓM TẮT LÝ THUYẾT

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.

4.1.1 Cấu trúc một chương trình C theo hàm



- Đầu chương trình là các khai báo về sử dụng thư viện, khai báo hằng số, khai báo các biến toàn cục và khai báo các kiểu dữ liệu tự định nghĩa, khai báo hàm con (các nguyên mẫu hàm).
- 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.
- Hàm chính (main()): Chứa các biến, các lệnh và các lời gọi hàm cần thiết trong chương trình.
- 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.

4.1.2 Cách xây dựng một hàm con

- Định nghĩa một hàm bao gồm:
  - Khai báo kiểu hàm
  - Đặt tên hàm
  - Khai báo các tham số
  - Các câu lệnh cần thiết để thực hiện chức năng của hàm.
- Có 2 loại hàm:
  - **void**: Hàm không trả về trị. Những hàm loại này thường rơi vào những nhóm chức năng: **Nhập / xuất dữ liệu , thống kê, sắp xếp, liệt kê.**
  - **Kiểu dữ liệu cơ bản (rời rạc/ liên tục) hay kiểu dữ liệu có cấu trúc**: Hàm trả về trị. Sau khi thực hiện xong, kết quả được lưu lại. Những hàm loại này thường được sử dụng trong các trường hợp: **Đếm, kiểm tra, tìm kiếm, tính trung bình, tổng, tích, ...**

Hàm không trả về trị:

```
void Ten_Ham(ds tham so neu co)
{
        Khai báo các biến cục bộ;
        Các câu lệnh / khối lệnh hay lời gọi
        đến hàm khác;
}
```

Hàm trả về trị:

```
Kieu_dl Ten_Ham(ds tham so neu co)
{
        Kiểu_dl kq;
        Khai báo các biến cục bộ;
        Các câu lệnh / khối lệnh hay lời gọi đến hàm khác;
        return kq;
}
//kiểu dữ liệu của hàm là kiểu của kết quả trả về.
```

4.1.3 Tham số của hàm

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

- Tham trị: Giá trị của biến tham số đầu vào không thay đổi sau khi hàm thực hiện xong. Tham số dạng này chỉ mang ý nghĩa là dữ liệu đầu vào.

  Ví dụ: int KiemTraNguyenTo(int n);

- 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.

  Ví dụ:

  Hàm nhập số nguyên: void NhapSoNguyen(int &n);

  Hàm hoán vị giá trị hai số: void HoanVi(int &a, int &b);

4.1.4 Cách gọi hàm con ra thực hiện

Một hàm chỉ được thực thi khi ta có một lời gọi đến hàm đó.

Cú pháp gọi hàm:

<Tên hàm>([Danh sách các tham số])

4.2 THỰC HÀNH CƠ BẢN

Câu 1: Viết chương trình thực hiện các chức năng sau (dùng hàm):

a. Nhập vào một số nguyên *n* (*0 < n < 100*).

b. Kiểm tra *n* có phải là số nguyên tố không?

c. Liệt kê các số nguyên tố trong phạm vi từ 1 đến *n*.

d. Đếm số lượng số nguyên tố trong phạm vi từ 1 đến *n*.

e. Tính tổng các số nguyên tố trong phạm vi từ 1 đến *n*.

f. Tính trung bình cộng các số nguyên tố trong phạm vi từ 1 đến *n*.

Hướng dẫn:

a. Nhập vào một số nguyên n (0 < n < 100).

```
void NhapSoNguyen(int &n)
{
    Lặp công việc{
        Nhập n
        Nếu n<=0 hoặc n>=100 thì thông báo nhập sai, hãy nhập lại.
    } Chừng nào (n<=0 hoặc n>=100);
}
```

b. Kiểm tra *n* có phải là số nguyên tố không? Hàm trả về 1: nếu *n* là số nguyên tố

Hàm trả về 0: nếu *n* không phải là số nguyên tố

int KTNT(int n) {…}

c. Liệt kê các số nguyên tố trong phạm vi từ 1 đến *n*.

void LietKeNT(int n) {…}

d. Đếm số lượng số nguyên tố trong phạm vi từ 1 đến *n*.

Hàm trả về số lượng số nguyên tố đã đếm được trong phạm vi từ 1 đến n.

int DemNT(int n) {…}

e. Tính tổng các số nguyên tố trong phạm vi từ 1 đến n.

int TongNT(int n) {…}

f. Tính trung bình cộng các số nguyên tố trong phạm vi từ 1 đến n.

float tbcNT(int n) {…}

❖ **Lưu ý:**

- Với mỗi hàm, ta test thử trong hàm *main()*, sau khi chạy ra kết quả đúng ta mới làm tiếp những hàm khác.
- Cách gọi thực hiện hàm con trong main(): chỉ gọi **tên_hàm (các tham số thực tương ứng)**

Câu 2: Viết chương trình theo hàm cho phép thực hiện chọn lựa công việc:

a. Giải phương trình bậc 1 ax+b=0.

b. Kiểm tra một số nguyên có là số hoàn thiện không?

c. Liệt kê các số hoàn thiện trong phạm vi từ 1..n (n do người dùng nhập)

d. Tìm ước chung lớn nhất của hai số nguyên *a*, *b* nhập từ bàn phím.

e. Thoát khỏi chương trình.

Hướng dẫn:

- Hãy xác định bạn cần viết bao nhiêu hàm, đó là những hàm nào?
- Trong hàm main, gọi hàm như bình thường. Sau khi chương trình đã chạy được và không còn lỗi, hãy sửa đổi để có chương trình thực hiện theo chức năng, tham khảo đoạn code sau:

```
//Xuất ra menu công việc
void XuatMenu()
{
    printf("1: Giai phuong trinh bac 1\n");
    printf("2: Kiem tra so hoan thien\n");
    printf("3: Liet ke so hoan thien trong pham vi tu 1 den n\n");
```

```
        printf("4: Tim uoc chung lon nhat cua hai so nguyen\n");
        printf("0: Thoat\n");
}
//-----------------------------------------------
//đoạn code này viết trong hàm main()
//dùng một biến nguyên để lưu công việc mà người dùng chọn int chon;
  do{
    XuatMenu(); //hiển thị menu công việc cho người dùng
 chọn
    printf("Hay chon cong viec:"); scanf("%d", &chon); switch (chon){
        case 1:
            float a, b;
            //nhập hệ số a, b
            //Gọi hàm giải pt bậc 1
            break;
            case 2:
                  //khai báo và nhập số nguyên
                  //gọi hàm Kiểm tra x có phải số hoàn thiện
               không
            break;
            case 3:
                 //khai báo và nhập số nguyên n
                 //gọi hàm xuất các số hoàn thiện trong phạm vi
             1… n
            break;
            case 4:
                //khai báo và nhập số 2 số nguyên a, b
                //gọi hàm tìm ước chung lớn nhất
            break;
            default: chon=0;
        }
  }while (chon!=0);
```

4.3 THỰC HÀNH NÂNG CAO

Câu 3: Viết chương trình nhập từ bàn phím 2 số a, b và một ký tự ch.

Nếu: ch là **"+"** thì thực hiện phép tính a + b và in kết quả lên màn hình.

ch là **"–"** thì thực hiện phép tính a - b và in kết quả lên màn hình.

ch là **"*"** thì thực hiện phép tính a * b và in kết quả lên màn hình.

ch là **"/"** thì thực hiện phép tính a / b và in kết quả lên màn hình.

Câu 4: 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).*

BÀI 5: MẢNG

Sau khi thực hành xong bài này, sinh viên có thể nắm được:

- *Cách khai báo dữ liệu kiểu mảng*
- *Các thao tác nhập xuất, các kỹ thuật thao tác trên mảng.*
- *Luyện tập cài đặt chương trình con (hàm).*

5.1 TÓM TẮT LÝ THUYẾT

Mảng một chiều 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.

Khai báo:

```
Kiểu_dữ_ liệu Tên_mảng[Kích thước];
```

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

Sử dụng:

```
Tên_biến_mảng[chỉ số];
```

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

Chỉ số (số thứ tự)	0	1	2	3	4
	A[0]	A[1]	A[2]	A[3]	A[4]

5.2 THỰC HÀNH CƠ BẢN

Câu 1: Viết chương trình thực hiện:

a. Nhập mảng số nguyên gồm *n* phần tử *(0 < n ≤ 10).*

b. Xuất mảng vừa nhập

Hướng dẫn:

Chương trình minh hoạ

```
#include <conio.h>
#include <stdio.h>
#define MAX 100
/*---------------------------------
*Hàm nhập sô lượng phần tử cho mảng
*/
void NhapSL(int &n)
{
	do{
		printf ("Nhap so phan tu 0<sl<100: ");
		scanf (" %d ", &n);
	}while (n<=0 || n>100);
}
/*-----------------------------------
*Hàm nhập giá trị cho từng phần tử trong mảng
*/
void NhapMang (int a[], int n)
{
	for (int i = 0; i < n; i ++)
	{
		printf (" a[%d] = ", i);
		scanf (" %d ", &a[i]);
	}
}
/*-----------------------------------
*Hàm xuất các phần tử của mảng ra màn hình
*/
void XuatMang (int a[], int n)
{
	printf ("\nMang gom cac phan tu:\n ");
	for (int i = 0; i < n; i ++)
		printf (" %5d", a[i]);
}
```

```
//================================
int main()
{
	int a[MAX], n;
	NhapMang (a,n);
	XuatMang (a,n);
	return 0;
}
```

Câu 2: Làm tiếp theo trong chương trình của câu 1 với các yêu cầu sau:

a. Xuất các phần tử chia hết cho 3 có trong mảng.

b. Đếm số lượng số dương có trong mảng.

c. Tính tổng các số trong mảng.

d. Tính trung bình cộng của mảng.

e. Tính trung bình cộng các phần tử dương có trong mảng.

f. Xuất các số nguyên tố có trong mảng

g. Đếm số lượng số nguyên tố có trong mảng.

h. Tính tổng các số nguyên tố có trong mảng.

i. Tính trung bình cộng các số nguyên tố có trong mảng.

j. Tìm phần tử dương đầu tiên.

k. Tìm phần tử dương cuối cùng.

l. Tìm giá trị phần tử lớn nhất (nhỏ nhất).

Hướng dẫn

a. Xuất các phần tử chia hết cho 3 có trong mảng.

Giải thuật:

- Đi từ đầu mảng đến cuối mảng
  - for (int i=0; i<n ; i++)
- Kiểm tra từng phần tử của mảng xem có chia hết cho 3 không

- if (a[i] %3==0)

  Nếu có thì in phần tử đó ra

  printf (" %d", a[i]);

c. Tính tổng các số trong mảng.

Giải thuật:

- Khai báo một biến s để lưu trữ tổng s = 0;
- Đi từ đầu mảng đến cuối mảng
  - for (int i=0; i<n ; i++)
- cộng dồn các phần tử a[i] và lưu vào biến s
  - s= s+ a[i] ;

j. Tìm phần tử dương đầu tiên

Giải thuật :

- Đi từ đầu mảng đến cuối mảng
  - for (int i=0; i<n ; i++)
- Kiểm tra từng phần tử của mảng xem có dương không
  - if (a[i] > 0)

    Nếu có thì xuất phần tử đó ra và dừng chương trình

    return a[i] ;

Câu 3: Viết chương trình thực hiện:

a. Nhập mảng số thực gồm *n* phần tử *(0 < n ≤ 100).*

b. Xuất mảng vừa nhập

Hướng dẫn:

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

```
#define MAX 100
---------------------------------
// Hàm nhập sô lượng phần tử cho mảng
void NhapSL(int &n)
{
    do{
        printf ("Nhap so phan tu 0<sl<100: ");
        scanf (" %d ", &n);
    }while (n<=0 || n>100);
}
//  Hàm nhập giá trị cho từng phần tử trong mảng
void NhapMang (float a[], int n)
{
    for (int i = 0; i < n; i ++)
    {
        printf (" a[%d] = ", i);
        scanf (" %f ", &a[i]);
    }
}
// Hàm xuất các phần tử của mảng ra màn hình
void XuatMang (float a[], int n)
{
    printf ("\nMang gom cac phan tu:\n ");
    for (int i = 0; i < n; i ++)
        printf (" %8.2 f ", a[i]);
```

```
}
int main()
{
  float a[MAX];
  int n;
  NhapMang (a,n);
     XuatMang (a,n);
  return 0;
}
```

5.3 THỰC HÀNH NÂNG CAO

Câu 4: Viết chương trình thực hiện:

a. 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.

b. Xuất mảng.

c. Xuất ra màn hình các phần tử là số chính phương nằm tại những vị trí lẻ trong mảng.

d. Xuất ra vị trí của các phần tử có giá trị lớn nhất.

e. Viết hàm tính tổng các phần tử nằm ở vị trí chẵn trong mảng.

f. Viết hàm sắp xếp mảng theo thứ tự tăng dần.

Yêu cầu chương trình thực hiện theo menu chức năng.

BÀI 6: KIỂU DỮ LIỆU CÓ CẤU TRÚC

Sau khi học xong bài này, học viên có thể:

- *Nắm được các khái niệm cơ bản về kiểu dữ liệu có cấu trúc;*
- *Biết nhập, xuất dữ liệu có cấu trúc cho một phần tử..*

6.1 TÓM TẮT LÝ THUYẾT

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.

Cú pháp định nghĩa kiểu cấu trúc:

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

//dùng từ khoá typedef để định nghĩa một tên mới cho kiểu dữ liệu đã có.

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

Khai báo biến kiểu cấu trúc:

```
<tên mới> <tên biến>;
```

Truy xuất:

```
<Tên biến cấu trúc> <tên thành phần>;
```

6.2 THỰC HÀNH CƠ BẢN

Câu 1: Định nghĩa kiểu dữ liệu phân số gồm tử số và mẫu số. Viết chương trình thực hiện:

a. Nhập 1 phân số

b. Xuất 1 phân số

c. Rút gọn phân số

d. Tính tổng 2 phân số

e. So sánh 2 phân số

Hướng dẫn:

Sinh viên tham khảo chương trình minh hoạ.

```
//Khối khai báo và định nghĩa kiểu dữ liệu phân số
#include<stdio.h>
#define MAX 100
typedef struct PHANSO
{
    int tu, mau;
}PS;
//Khai báo các nguyên mẫu hàm----------------------------------------
void Nhap1ps(PS &x); void Xuat1ps(PS x);
int UCLN(int a, int b); //trả về giá trị ucln của hai số a, b
void RutGon(PS &x); PS Cong(PS x, PS y);
//hàm trả về phân số kết quả sau khi cộng
int SoSanh(PS x, PS y);
/* trả về 0 nếu phân số x = phân số y trả về số >0 nếu phân số x > phân số y trả về <0 nếu phân số x < phân số y */
```

```
//Hàm chính----------------------------------------------------------------
int main()
{
    PS x, y;
    //nhập và xuất hai phân số
    printf("Nhap phan so thu nhat:\n"); Nhap1ps (x); Xuat1ps(x);
    printf("Nhap phan so thu hai:\n"); Nhap1ps (y); Xuat1ps(y); //tính tổng hai ps
    và xuất kết quả ra màn hình
    PS tong=Cong(x,y); printf("Tong cua hai phan so la: "); Xuat1ps (tong);
    //so sánh hai ps
    int kq= SoSanh(x,y);
    if (kq==0) printf("ps1 bang ps2");
    else
    {
        if (kq>0) printf("ps1 lon hon ps2"); else printf("ps1 nho hon ps2");
    } return 0;
}
//Các hàm con=========================================
//Hàm nhập 1 phân số
void Nhap1ps (PS &x)
{
    printf("Nhap tu so: "); scanf("%d", &x.tu);
    //nhập mẫu số phải khác 0, nhập sai bắt nhập lại
    do
    {
```

```
    //thông báo và nhập cho tử của phân số x
    } while (x.tu==0);
    //nếu mẫu số âm thì chuẩn hoá, ví dụ 1/-2 chuyển thành -1/2
    if (x.mau<0)
    {
        x.tu=-x.tu; x.mau=-x.mau;
    }
}
//Hàm xuất 1 phân số----------------------------------------------------------
void Xuat1ps(PS x)
{
    //xuất ra theo dạng tu/mau, ví dụ: 3/5
}
//Hàm tìm ước chung lớn nhất--------------------------------------------------
int UCLN(int a, int b)
{
    a=abs(a); b=abs(b); //lấy trị tuyệt đối
    Lặp chừng nào a ≠ b
        Nếu a>b thì a=a-b;
        Nếu b>a thì b=b-a;
    return a;
}
//Hàm rút gọn phân số---------------------------------------------------------
void RutGon(PS &x)
{
```

```
    if (x.tu==0) return; //nếu phân số có tử = 0 thì không cần rút gọn
    //lấy giá trị ucln của tử và mẫu của phân số x bằng cách gọi hàm UCN(?, ?)
    //tính lại x.tu và x.mau
}
//Hàm tính tổng hai phân số----------------------------------------------------------
PS Cong(PS x, PS y)
{
    PS tong;
    //phân số lưu kết quả sau khi cộng
    tong.tu= ….?; tong.mau=..?; RutGon(tong);
    return tong;
}
//Hàm so sánh hai phân số----------------------------------------------------------
int SoSanh(PS x, PS y)
{
    return x.tu*y.mau – y.tu*x.mau;
}
```

6.3 THỰC HÀNH NÂNG CAO

Câu 2: Làm tiếp câu 1, cài đặt các hàm sau:

a. Nhập vào dãy phân số

b. Xuất dãy phân số

c. Tính tổng dãy

d. Tìm phân số lớn nhất

e. Sắp xếp dãy phân số tăng dần.

Câu 3:

3.1 Viết hàm nhập dữ liệu cho một sinh viên, thông tin về một sinh viên gồm có:

a. Mã số sinh viên (chuỗi 10 ký tự).

b. Tên (là chuỗi tối đa 10 ký tự).

c. Ngày tháng năm sinh (theo kiểu int. Mở rộng dd/mm/yy).

d. Giới tính (Nam hoặc Nữ).

e. Lớp (chuỗi 7 ký tự trong đó 2 ký tự đầu là năm vào học, 1 ký tự tiếp là bậc học (D: Đại học, C: Cao đẳng), 2 ký tự tiếp là ngành học (TH: Tin Học, KT: Kế Toán, QT: Điện tử, ĐT: Điện tử...).

f. Điểm toán, lý, tin (Kiểu số thực).

Hướng dẫn

3.1 Khai báo cấu trúc sinh viên

```
struct sinhvien
{   char ten[11];
    char mssv [11];
    char lop [8];
    char gioitinh;
    int ntns;
    float  toan,ly,tin, DTB ;
};
typedef struct sinhvien   sv;
```

// viết hàm nhập 1 sinh viên

Cách 1

```
void  nhap1sv( sv &a)
{
        flushall();
```

```
        printf("nhap ten sinh vien :");      gets (a. ten);
        printf("nhap ma so sinh vien :");      gets(a. mssv);
        printf("nhap  lop :");      scanf( "%s", &a. lop  );
        printf (" nhap nam sinh :") ; scanf( "%d", &a. namsinh  );
        flushall();
        printf(" gioi tinh ( Nam y/nu x ) :");  scanf( "%c", &a. gioitinh  );
        printf("nhap diem toan:");    scanf("%f ", &a.toan) ;
        printf("nhap diem ly:");       scanf("%f ", &a.ly ) ;
        printf("nhap diem tin:");      scanf("%f ", &a.tin );
        a.DTB= ( a.toan +a.ly +a.tin ) /3 ;
}
```

Cách 2

```
void  nhap1sv( sv &a)
{   flushall();
        printf("nhap ten sinh vien :");      gets (a. ten);
        printf("nhap ma so sinh vien :");      gets(a. mssv);
        printf("nhap  lop :");      scanf( "%s", &a. lop  );
        printf (" nhap năm sinh :"); scanf( "%d", &a. namsinh  );
        printf(" gioi tinh ( Nam y/nu x ) :");   a. gioitinh= getche();
        float t;
        printf("nhap diem toan:");    scanf("%f ", &t) ; a.toan= t ;
        printf("nhap diem ly:");       scanf("%f ", &t);   a.ly = t ;
        printf("nhap diem tin:");      scanf("%f ", &t);   a.tin = t;
        a.DTB= ( a.toan +a.ly +a.tin ) /3 ;
}
```

3.2 Viết hàm xuất dữ liệu một sinh viên với thông tin vừa nhập ở trên.

Hướng dẫn

```
void  xuat1sv( sv a)
{
  printf("Ten sinh vien :");      puts(a. ten);
  printf(" Ma so sinh vien :");  puts(a. mssv);
    printf(" Lop  :");  puts(a. lop);
  if(a.gioitinh=='x' )
        printf(" gioi tinh : Nu ");
  else
        printf(" gioi tinh : Nam ");
  printf((" ntns : %d, a. ntns);
  printf("diem toan: %f " , a.toan ) ;
  printf("diem ly: %f " , a.ly ) ;
  printf("diem tin: %f " , a.tin ) ;
  printf("diem TB : %f" , a.DTB ) ;
}
```

3.3 Viết hàm nhập danh sách sinh viên, lưu trên mảng một chiều.

```
void nhapdssv(sv a[ ],  int  n)
{
  for(int i=0 ; i<n ; i++)
  {
        printf("nhap hoc sinh thu: %d",i+1);
        nhap1sv(a[i]);
  }
}
```

3.4 Viết hàm xuất danh sách sinh viên.

```
void xuatdssv(sv a[ ],  int  n)
{
  for(int i=0;i<n;i++)
  {
        printf(" sinh vien thu: %d",i+1);
        xuat1sv(a[i]);
  }
}
```

3.5 Tìm sinh viên có tên là " X ". ("X" do người dùng nhập vào)

Hướng dẫn

```
void timtensv (sv a[ ] ,  int  n, char x[])
{ int flag = 0 ;
  for(int i=0;i<n;i++)
      if ( strcmp ( a[i].ten, x )==0 )
      {
        xuat1sv (a[i] ) ;
        flag = 1 ;
       }
  if ( flag == 0 )
        printf (" ko co ten sv can tim ");
}
```

3.6 Đếm số lượng sinh viên có điểm trung bình >5.

Hướng dẫn

```
int  DemsvDTB>5 (sv a[ ],  int  n)
  {     int dem =0;
  for(int i=0;i<n;i++)
  {     if ( a[i].DTB > 5)
              dem++;
  }
  return dem;
  }
```

3.7 Xuất danh sách sinh viên thuộc ngành công nghệ thông tin.

Hướng dẫn

3.8 Xuất danh sách sinh viên Nữ

Hướng dẫn

Đi từ đầu danh sách dến cuối danh sách

Kiểm tra từng phần tử a[i].gioitinh == ”nữ“ thì xuất sinh viên đó ra.

```
void xuatdssvnu(sv a[ ],  int  n)
{ int flag =0;
  for(int i=0;i<n;i++)
  {     if (a[i].gioitinh==‘x’)
        {       xuat1sv(a[i]);
              flag=1;
        }
  }
  if(flag ==0)
        printf(“ k co SV nu trong lop ”);
}
```

3.9 Sắp xếp danh sách sinh viên tăng dần theo cột DTB.

Hướng dẫn

```
void InterchangeSort( sv a[ ], int n )
{
  sv tam ;
        for ( int i = 0 ; i<n-1 ; i++ )
           for ( int j =i+1; j < n ; j++ )
              if( a[ i ].DTB> a[ j ].DTB )
              {
                    tam = a[i] ;
                    a[i]  = a[j] ;
                    a[j] = tam;
              }
}
```

6.4 KIỂM TRA THỰC HÀNH

TÀI LIỆU THAM KHẢO

1. Phạm Văn Ất, Kỹ thuật Lập trình C- cơ bản và nâng cao, NXB KH & KT - 2003.
2. Nguyễn Tấn Trần Minh Khang, Bài giảngKỹ Thuật Lập trình, Khoa Công Nghệ Thông Tin, Đại học Khoa học Tự nhiên
3. Nguyễn Thanh Thủy (chủ biên), Nhập môn lập trình ngôn ngữ C, Nhà xuất bản Khoa học kỹ thuật – 2000.
4. Trần Minh Thái, Bài giảng và bài tập Lập trình căn bản; Khoa Công Nghệ Thông Tin, Đại học Khoa học Tự nhiên
5. Trần Minh Thái, Giáo Trình Bài Tập Kỹ Thuật Lập Trình, Cao đẳng Công Nghệ Thông Tin Tp. Hồ Chí Minh
6. Brain W. Kernighan & Dennis Ritchie, The C Programming Language, Prentice Hall Publisher, 1988.