Bài 1: Kiểu dữ liệu. Biểu thức toán học. Hàm nhập – xuất

A. Lý thuyết:

1) Các kiểu dữ liệu trong C:

- char	lưu trữ 1 kí tự bất kì. VD: 'A', '1', ' ', ...	(1 byte)
- int	lưu trữ số nguyên	(2 byte)
- long	lưu trữ số nguyên vượt ngoài khoảng của int	(4 byte)
- float	lưu trữ số thực	(4 byte)
- double	lưu trữ số thực vượt ngoài khoảng của float	(8 byte)

***Lưu ý:** 1) Cách tính khoảng rộng của kiểu dữ liệu

- VD: 1 byte = 8 bit -> $2^8 = 256$

  -> độ rộng [-256/2 ; 256/2 -1] -> [-128 ; 127]

2) Các tiền tố đi kèm kiểu dữ liệu: short, long, unsigned, signed,...

- VD: int -> [-32768 ; 32767]
  unsigned int -> [0 ; 65535]

- Cú pháp khai báo biến:

Kiểu dữ liệu + tên biến(= giá trị khởi tạo) ;

VD: int abc; (Khởi tạo biến abc)

char abc = 'A'; (Khởi tạo biến abc với giá trị ban đầu là 'A');

2) Các phép toán trong C:

- Cộng trừ nhân chia	->	+, -, *, /
- Chia lấy phần nguyên	->	/
- Chia lấy phần dư	->	%
- Phép tăng giá trị của biến lên 1	->	++
- Phép giảm giá trị của biến đi 1	->	--

BK-FCMP

***Lưu ý:** Phân biệt giữa phép chia và chia lấy phần nguyên

- VD: thực hiện phép '/' trên 2 số nguyên (kiểu int)

$4/3 = 1$; $3/3 = 1$

- VD: thực hiện phép '/' trên 2 số thực (kiểu float)

$4/3 = 1.333$; $3/3 = 1$

3) Hàm nhập – xuất (vào – ra) cơ bản:

- Cú pháp hàm in:

```
printf("chuoi_can_in", tham_so);
```

VD: printf("Hello world"); -> Hello world

->(TH ko có tham số truyền vào)

VD: printf("1+1=%d",2); -> 1+1=2

->(TH có tham số truyền vào)

- Cú pháp hàm nhập:

```
scanf("%d",&ten_bien);
```

VD: int a; (Phải khởi tạo biến a trước khi cho biến a nhận giá trị)

scanf("%d",&a); (Cho biến a nhận giá trị bằng cách nhập từ màn hình)

***Lưu ý:**

Kiểu dữ liệu	Hàm printf	Hàm scanf
char	%c	%c
int	%d	%d
long	%d	%ld
float	%f	%f
double	%f	%lf
char[ ]	%s	%s

BK-DCMP

B. Ví dụ tổng quan:

1)Chương trình tính tổng

```c
#include<stdio.h>                          //Khai bao thu vien (bat buoc)
main(){                                    //Khai bao ham main (bat buoc)
    int a = 3, tong = 0;                   //Khoi tao 2 bien a, tong chua gia tri ban dau
    int b;                                 //Khoi tao bien b ko chua gia tri
    printf("Moi ban nhap gia tri cho b: "); //Cau lenh in ko chua tham so
    scanf("%d",&b);                        //Nhap gia tri cho b
    tong = a+b;                            //Bieu thuc
    printf("Tong cua %d voi 3 la %d",b,tong); //Cau lenh in chua 2 tham so
}                                          //Dong ham main
```

```
Moi ban nhap gia tri cho b: 6
Tong cua 6 voi 3 la 9
--------------------------------
Process exited after 2.616 seconds with return value 0
Press any key to continue . . .
```

2)Chương trình tìm giao điểm của đường thẳng có dạng ax+b với trục hoành

```c
#include<stdio.h>
main(){
    float a,b;
    printf("Nhap gia tri cho a va b: ");
    scanf("%f %f",&a,&b);
    printf("Duong thang cat truc hoanh tai diem (%f;%d)",-b/a,0);
}
```

```
Nhap gia tri cho a va b: 3 4
Duong thang cat truc hoanh tai diem (-1.333333;0)
--------------------------------
Process exited after 1.985 seconds with return value 49
Press any key to continue . . .
```

- Tại ví dụ này ta có thể thấy hàm nhập scanf cũng có thể chứa nhiều tham số tương tự hàm printf

- Ta chọn kiểu dữ liệu float cho 2 biến a, b vì thực hiện phép '/'

C. Bài tập:

1) Viết chương trình cho người dùng nhập chiều dài, chiều rộng của 1 hình chữ nhật rồi in ra diện tích của hình chữ nhật đó

2) Viết chương trình tính số bình phương của một số được nhập.

3) Viết chương trình tính trung bình cộng của 5 số được nhập từ bàn phím

Bài 2: Cấu trúc rẽ nhánh của chương trình

A. Lý thuyết:

1) Các phép toán quan hệ và logic:

- Các phép toán quan hệ: >, >=, <, <=, ==, !=

Toán tử	Ý nghĩa	VD
>	lớn hơn	2 > 3 (mệnh đề sai -> có giá trị 0) 6 > 4 (mệnh đề đúng -> có giá trị 1)
>=	lớn hơn hoặc bằng	6 >= 4 (có giá trị 1)
<	nhỏ hơn	3 < 2 (có giá trị 0)
<=	nhỏ hơn hoặc bằng	4 <= 4 (có giá trị 1)
==	so sánh bằng	3 == 4 (có giá trị 0)
!=	so sánh khác	3 != 4 (có giá trị 1)

- Các phép toán logic: &&, ||, !

+) Phép VÀ (&&): dieu_kien_1 && dieu_kien_2

-> mệnh đề đúng khi cả 2 điều kiện đúng

VD: (4 < 5) && (6 >= 3)-> mệnh đề đúng -> có giá trị là 1

+) Phép HOẶC (||): dieu_kien_1 || dieu_kien_2

BK-DSVP

-> mệnh đề đúng khi 1 trong 2 điều kiện đúng

VD: (1 != 2) || (1 >=2) -> mệnh đề đúng -> có giá trị là 1

+) Phép PHỦ ĐỊNH (!): !(dieu_kien_1)

-> mệnh đề đúng khi điều kiện 1 sai

VD: !(1 == 2) -> mệnh đề đúng -> có giá trị là 1

Mệnh đề trên tương đương với (1 != 2)

***Lưu ý**: Phân biệt phép gán "=" với phép so sánh bằng "=="

VD: a = 3; -> Gán cho biến a giá trị là 3

a == 3; -> So sánh xem giá trị biến a có bằng 3 hay ko

2) Câu lệnh if...else:

- Cú pháp:

```
if(dieu_kien){
    //Khối lệnh 1;
} else {
    //Khối lệnh 2;
}
```

Nếu điều kiện bên trong if đúng sẽ thực hiện khối lệnh 1, còn sai thì thực hiện khối lệnh 2

VD:

```
if (4 % 2 == 0) printf("4 la so chan");
else printf("4 la so le");
```

***Lưu ý:** 1 câu lệnh if chỉ có 1 lệnh else, muốn có nhiều else thì có thể sử dụng else if. Và câu lệnh if cũng có thể không có câu else

VD:

```
if (x < 0) printf("X la so am");
else if (x == 0) printf("X la so 0");
else printf("X la so duong");
```

BK-DCMP

3) Câu lệnh switch-case:

- Cú pháp : switch(bieu_thuc){

```
        case gia_tri_1:     lenh_1;  (break;)
        case gia_tri_2:     lenh_2;  (break;)
        .....
        [default:     lenh_n;   (break;)]
    }
```

- Câu lệnh sẽ thực hiện lệnh/khối lệnh tương ứng với giá trị của biểu thức trong switch

VD:
```
switch(1){                          //Giá trị trong switch = 1
    case 1:     lenh_1;  break;     //Lệnh 1 được thực hiện
    case 2:     lenh_2;  break;     //Vì case 1 có break nên lệnh 2 ko đc thực hiện
    default:    lenh_n;  break;     //Lệnh default đc thực hiện khi
}
```

***Lưu ý:** - câu lệnh switch-case có thể ko có default

- Lệnh case thiếu break; sẽ chạy tiếp khối lệnh của lệnh case tiếp theo

B. Ví dụ tổng quan:

BK-DCMP

```c
#include<stdio.h>
main(){
    int a;
    printf("Moi ban nhap gia tri cho a: ");
    scanf("%d",&a);
    switch(a){
        case 1:
            printf("Ban nhap so 1");
            break;
        case 2:
            printf("Ban nhap so 2");
            break;
        default:
            printf("Moi ban nhap so khac");
            break;
    }
}
```

```
Moi ban nhap gia tri cho a: 17
So ban nhap la so le nguyen duong
```

***Lưu ý:** - Trong trường hợp bên trong ngoặc nhọn {...} là 1 câu lệnh ta có thể lược bỏ không viết {...}, nhưng trường hợp là khối lệnh thì bắt buộc phải có {...}

(thường gặp ở cấu trúc if...else, switch-case, vòng lặp,...)

```c
#include<stdio.h>
main(){
    int a;
    printf("Moi ban nhap gia tri cho a: ");
    scanf("%d",&a);
    if((a > 0) && (a%2==0)){
        printf("So ban nhap la so chan nguyen duong");
    } else if((a > 0) && (a%2!=0)){
        printf("So ban nhap la so le nguyen duong");
    } else if(a == 0){
        printf("So ban nhap la so 0");
    } else{
        printf("So ban nhap la so am");
    }
}
```

```
Moi ban nhap gia tri cho a: 0
Moi ban nhap so khac
```

C. Bài tập:

1) Viết chương trình cho người dùng nhập 1 số nguyên. Nếu số đó thuộc [14;30] hoặc bình phương số đó bằng 25 thì in ra màn hình là “Chien thang”, và “Thua” nếu số đó không thuộc đoạn cho trước

2) Viết chương trình switch-case theo phép toán người dùng nhập (+, -, *, /). Rồi thực hiện phép toán đó lên 2 số a = 3 và b = 4. In kết quả thực hiện được lên màn hình. VD: phép ‘-’ -> a-b -> -1

3) Tương tự bài 2, thay vì 2 số a = 3 và b = 4 thì để người dùng nhập vào 2 số đó.

Bài 3: Vòng lặp

A. Lý thuyết:

1) Vòng lặp for:

- Cú pháp:
```
for([bieu_thuc_1]; [bieu_thuc_2]; [bieu_thuc_3]){
    //khối lệnh;
}
```

- Thường dùng để thực hiện lặp đi lặp lại khối lệnh bên trong với số lần lặp xác định.

- Biểu thức 1: Chứa biến chạy (thông thường được kí hiệu là i)

- Biểu thức 2: Chứa điều kiện để xem vòng lặp có tiếp tục lặp hay ko. Nếu điều kiện đúng, vòng lặp sẽ tiếp tục lặp

- Biểu thức 3: Là biểu thức thay đổi giá trị của biến chạy

- VD: In ra các số từ 1->10 sử dụng vòng lặp for

```
int i;
for(i=1; i<=10; i++){
    printf("%d ",i);
}
```

***Lưu ý:** - Vòng lặp for có thể ko cần đầy đủ cả 3 biểu thức, hoặc viết các biểu thức ở vị trí khác

VD:
```
int i =1;
for( ;  i<=10 ; ){
    printf("%d ",i);
    i++;
}
```

BK-DCVP

- Câu lệnh i++; tương đương i+= 1; tương đương i = i+1; và đều mang ý nghĩa tăng giá trị biến i lên 1 đơn vị

2) Vòng lặp while:

- Cú pháp :
```
while(dieu_kien){
        //Khối lệnh;
}
```
- Thường dùng để thực hiện lặp với số lần lặp không xác định
- VD: In ra các số từ 1->10 sử dụng vòng lặp while

```
int i =1;           //câu lệnh tương đương với biểu thức 1 ở vòng for
while(i<=10){   //điều kiện giống với biểu thức 2 ở vòng for
    printf("%d ",i);
    i++;                //câu lệnh giống biểu thức 3 ở vòng for
}
```

3) Vòng lặp do{...} while(); :

- Cú pháp:
```
do{
        //Khối lệnh;
}while(dieu_kien);
```
- Chức năng tương tự vòng lặp while và thường được dùng với số lần lặp không xác định.
- VD: In các số từ 1->10 bằng vòng lặp do{...}while

```
int i=1;
do{
    printf("%d ",i);
    i++;
}while(i<=10);
```

BK-DCMP

***Lưu ý:**

- Điểm khác của vòng lặp do{...}while với vòng lặp while:

  - Vòng lặp do{...}while sẽ thực hiện khối lệnh bên trong trước rồi mới kiểm tra điều kiện để tiếp tục lặp hoặc dừng -> số lần thực hiện khối lệnh tối thiểu là 1 lần
  - Vòng lặp while sẽ kiểm tra điều kiện trước rồi mới thực hiện khối lệnh

    -> có thể khối lệnh sẽ không được thực hiện

- Các vòng lặp đều có thể dùng thay thế được cho nhau

B. Ví dụ tổng quan:

1) Chương trình tính tổng các số chẵn từ 1->100

```
#include<stdio.h>
main(){
    int tong =0, i;
    for(i=2; i<=100; i=i+2){
        tong += i;
    }
    printf("Tong cac so chan tu 1->100 la: %d",tong);
}
```

```
Tong cac so chan tu 1->100 la: 2550
--------------------------------
```

- Biến chạy i chạy từ 2 đến 100 và sau mỗi lần lặp i được tăng 2 đơn vị.
- Biến tong += i có nghĩa là tong mới sẽ bằng tong trước cộng thêm i

  VD: Ban đầu tong =0, i =2

  -> tong += i; có nghĩa là tong = tong + i = 0 + 2 =2

  Tiếp theo, khi i chạy lên giá trị bằng 4 -> tong (cũ) =2 và i=4

  -> tong += i; có nghĩa là tong = 2 + 4 = 6

  Và cứ tiếp tục cho đến i = 100, tong (cũ) = 2450 -> 2550 là kết quả

2) Chương trình đếm xem số được nhập có bao nhiêu chữ số

```c
#include<stdio.h>
main(){
    int N, dem = 0;
    scanf("%d",&N);
    while(N!=0){
        N = N/10;
        dem++;
    }
    printf("So ban nhap co %d chu so",dem);
}
```

```
15315
So ban nhap co 5 chu so
--------------------------------
```

- N là số được người dùng nhập vào, biến đếm là biến thể hiện đã đếm được bao nhiêu chữ số (ban đầu =0)

- Ý tưởng thuật toán: ta chia lấy phần nguyên của số N cho 10. Như vậy sau mỗi lần chia là ta đếm được 1 chữ số -> tăng biến đếm lên 1 đơn vị. Và kết thúc vòng lặp khi N = 0

VD: Ban đầu số được nhập là 15315 -> N = 15315, dem = 0

Do N != 0 nên khối lệnh của while được thực hiện

N = N/10 -> Chia lấy nguyên ta được N mới bằng 1531, dem =1

Tiếp theo N = 1531 và khác 0 nên -> N mới bằng 153, dem = 2

Và cứ như vậy đến N = 1, dem =4 -> N = N/10 = 1/10 =0, dem =5

N =0 -> kết thúc vòng lặp và in ra kết quả của biến dem

3) Chương trình kiểm tra số người dùng nhập có phải thuộc [0;100]. Nếu sai người dùng sẽ phải nhập lại. Nếu đúng in số vừa nhập ra màn hình

```c
#include<stdio.h>
main(){
    int N;
    do{
        scanf("%d",&N);
    }while((N<0) || (N>100));
    printf("So vua nhap la %d",N);
}
```

```
101
1413
14
So vua nhap la 14
--------------------
```

***Lưu ý:** -Khi làm bài về vòng lặp chúng ta cần để ý dấu ngoặc { } và ( ) của vòng lặp. Tránh trường hợp viết thiếu hoặc viết sai vị trí của dấu ngoặc (xác định rõ những câu lệnh nào cần để trong vòng lặp).

C. Bài tập:

1) Tính tổng các số chia hết cho 3 từ 1 đến 100

2) Viết chương trình cho nhập n số dương, chương trình kết thúc khi số được nhập là số âm

3) Tính giai thừa của một số N được nhập, rồi in kết quả lên màn hình (Nếu N không phải là số nguyên dương thì cho người dùng nhập lại)

4*) In ra hình vuông bằng dấu * có kích thước NxN với N là số nguyên được người dùng nhập. VD:

INPUT		OUTPUT
3	->	*** *** ***

BK-DCMP

Bài 4: Vòng lặp(tiếp)

Điều khiển vòng lặp. Ép kiểu dữ liệu

A. Lý thuyết:

1) Câu lệnh continue:

- Cú pháp : continue;

- Câu lệnh continue; dùng để bỏ qua những câu lệnh nằm sau nó trong thân vòng lặp

- VD: In ra các số chẵn từ 1->10

```
int i =1;
for(i; i<=10; i++){
    if(i%2 != 0) continue;   //Nếu i không chia hết cho 2
    printf("%d ",i);         //-> i là số lẻ -> bỏ qua câu lệnh printf
}
```

2) Câu lệnh break:

- Cú pháp: break;

- Dùng để lập tức thoát khỏi vòng lặp, hoặc thoát khỏi cấu trúc switch

- VD: Cho nhập vô hạn số N, nếu số N được nhập < 0 thì kết thúc quá trình nhập

```
int N;
do{
    scanf("%d",&N);
    if(N<0) break;          //Khi N < 0 thoát khỏi vòng lặp do-while
}while(1);                  //while(1) tức là điều kiện luôn đúng
```

3) Ép kiểu dữ liệu:

- Ta được biết trong C, để thực hiện các phép tính giữa 2 biến thì 2 biến đó phải có cùng kiểu dữ liệu. VD: int + int -> int. Tổng 2 số kiểu int sẽ ra 1 số kiểu

BK-DCMP

int. Nhưng trong một số trường hợp khác kiểu dữ liệu thì ta phải tiến hành ”ép kiểu” (cũng gần giống như việc đổi cho cùng đơn vị)

- Trường hợp được chuyển kiểu tự động: (ép lên)

char -> int -> long int -> float -> double -> long double

- Đây có thể gọi là “ép lên” tức là VD: kiểu float (4 byte) là tập con của kiểu double (8 byte). Như vậy mọi phần tử thuộc kiểu float đều thuộc kiểu double -> Được chuyển kiểu tự động
- VD: int N=1;
  float f;
  f = N; //f lúc này có giá trị =1

- Trường hợp không được chuyển kiểu tự động: (ép xuống)

- Cú pháp: (Kiểu dữ liệu mới) bieu_thuc;
- VD: int N;
  float f = 1.6;
  N = (int) f; //N sẽ chỉ lấy phần nguyên của f
  printf("%d",N); //Kết quả in ra là 1

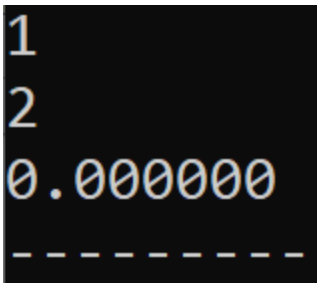
B. Ví dụ tổng quan:

1) Tính phân số a/b với a,b là kiểu int do người dùng nhập

```
#include<stdio.h>
main(){
    int a,b;
    float phanso;
    scanf("%d %d",&a,&b);
    phanso = (float) a/b;
    printf("%f",phanso);
}
```

```
1
2
0.500000
----------
```

Ta có thể xóa (float) và chạy lại chương trình, lúc này phanso sẽ nhận kết quả khác do không được ép kiểu sẽ hiểu phép “/” là phép chia lấy nguyên



C. Bài tập:

1) Kiểm tra xem một số được nhập có là số nguyên tố hay không ?

2) Tính tổng:

$$S = 1 + x + \frac{x^2}{2} + \frac{x^3}{3} + \cdots + \frac{x^n}{n}$$

với x và n được nhập từ bàn phím, x và n nguyên dương. (Gợi ý: sử dụng thư viện math.h bằng câu lệnh #include<math.h> , rồi dùng hàm pow(x,i) – hàm này có nghĩa là x^i và chú ý phần ép kiểu)

3) Viết một chương tình tự do có sử dụng câu lệnh break; hoặc continue; hoặc cả dùng cả 2

BK-DCAP

Bài 5: Mảng

A. Lý thuyết:

1) Khái niệm và cách sử dụng mảng:

- Mảng là tập hợp các phần tử có cùng kiểu dữ liệu

-> Công dụng dùng để lưu trữ nhiều giá trị có cùng kiểu dữ liệu

- Cú pháp khai báo: kieu_du_lieu ten_mang[so_phan_tu];

VD: int arr[10]; -> mảng arr có 10 phần tử, mỗi phần tử coi như là 1 biến kiểu int.

- Mảng giúp tối ưu hóa việc khai báo. Thay vì khai báo 10 lần int arr1,arr2,...; ta chỉ việc khai báo mảng như ở ví dụ trên

- Cách sử dụng mảng:

- Các phần tử trong mảng đều trùng tên là tên của mảng nhưng phân biệt với nhau bằng vị trí của phần tử trong mảng được đánh số 0,1,2,....hay còn là chỉ số. VD: arr[0] là phần tử thứ nhất

*Lưu ý: - Ngoài mảng một chiều ta còn có mảng 2 chiều, nhiều chiều.

VD: Khai báo mảng 2 chiều

int arr[5][5]; -> mảng 2 chiều có 25 phần tử, phần tử đầu là arr[0][0], phần tử thứ 2 là arr[0][1],.....

Tương tự với mảng nhiều chiều

- Có thể coi mảng 1 chiều như dãy dữ liệu còn mảng 2 chiều như một bảng dữ liệu

arr[0]	arr[1]	....	arr[n]

Mảng 1 chiều

arr[0][0]	arr[0][1]	.....	arr[0][n]
arr[1][0]	arr[1][1]	.....	arr[1][n]
.....	.....	.....	.....
arr[n][0]	arr[n][1]	.....	arr[n][n]

Mảng 2 chiều

BK-DCMP

2) Các thao tác cơ bản với mảng:

a) Nhập dữ liệu cho mảng:

- Thông thường sau bước khai báo mảng, ta sẽ nhập dữ liệu cho mảng. Dữ liệu có thể do ta qui định hoặc do người dùng nhập từ bàn phím.

- VD1: `int arr[3] = {4,17,-123};` (Dữ liệu tự qui định, chú ý dùng dấu ngoặc nhọn và dấu phẩy ngăn cách)

- VD2:
```
int i=0, arr[10];
for(i; i <10; i++)
scanf("%d",&arr[i]);
```
(Dữ liệu mảng được nhập từ bàn phím qua câu lệnh scanf)

- Ở ví dụ 2, ta cần lưu ý những điều sau:
  - Phần tử đầu tiên của mảng có chỉ số là 0 nên biến i sẽ chạy bắt đầu từ 0 đến 9 tức là <10 (mảng arr có 10 phần tử)
  - arr[i] được coi là 1 biến kiểu int
  - Vòng lặp for sẽ duyệt hết cả mảng từ phần tử đầu -> cuối

b) Xuất dữ liệu chứa trong mảng:

- Muốn in các giá trị có trong mảng ta dùng vòng lặp duyệt mảng, rồi dùng hàm printf in ra màn hình.

- VD:
```
int i = 0, arr[3] = {4,17,-123};
for(i; i <3; i++)
printf("%d ",arr[i]);
```

***Lưu ý:** - Đối với mảng chứa nhiều dữ liệu, thì câu lệnh printf nên có thêm các kí tự đặc biệt như "\n" (new line - In xuống dòng), "\t" (tab – In dấu tab). Mục đích để hiển thị dữ liệu dễ nhìn hơn

BK-DCMP

B. Ví dụ tổng quan:

1) Nhập, xuất mảng 1 chiều có 5 phần tử

```c
#include<stdio.h>
main(){
    int arr[5];
    for(int i = 0; i<5; i++){
        printf("Nhap arr[%d]: \n",i);
        scanf("%d",&arr[i]);
    }
    printf("Mang ban vua nhap la: \n");
    for(int i = 0; i<5; i++)    printf("%d \t",arr[i]);
}
```

```
Nhap arr[0]:
14
Nhap arr[1]:
3
Nhap arr[2]:
2000
Nhap arr[3]:
15
Nhap arr[4]:
5
Mang ban vua nhap la:
14      3       2000    15      5
--------------------------------
```

2) Tìm giá trị lớn nhất của mảng:

```c
#include<stdio.h>
main(){
    int arr[5];
    for(int i = 0; i<5; i++){
        printf("Nhap arr[%d]: \n",i);
        scanf("%d",&arr[i]);
    }                                   //Ket thuc phan nhap mang
    int max = arr[0];                   //Gia su max la phan tu dau tien
    for(int i = 0; i<5; i++){           //Duyet mang
        if(max < arr[i])    max = arr[i];   //Neu co phan tu > max thi gán max = phan tu do
    }
    printf("GTLN cua mang la: %d",max);     //In ket qua max ra man hinh
}
```

```
Nhap arr[0]:
14
Nhap arr[1]:
3
Nhap arr[2]:
2000
Nhap arr[3]:
15
Nhap arr[4]:
5
GTLN cua mang la: 2000
--------------------------------
```

3) Sắp xếp mảng theo thứ tự tăng dần:

- Ý tưởng thuật toán sắp xếp:

- Ở lượt xếp đầu (i=0), ta so sánh arr[0] với toàn bộ phần tử đứng sau nó trong mảng -> j = i+1 (j chạy duyệt toàn bộ phần tử đứng sau i). Nếu gặp phần tử nào nhỏ hơn arr[0] ta sẽ đổi chỗ. Sau lượt xếp này phần tử nhỏ nhất sẽ đứng ở đầu
- Ở lượt xếp thứ 2 (i=1), tương tự lượt xếp đầu, ta lại so sánh arr[1] với tất cả phần tử đứng sau nó và được phần tử bé thứ 2 = arr[1]
- Tương tự cho đến lượt xếp cuối cùng (i là phần tử gần cuối), ở ví dụ dưới là arr[3] sẽ đi so sánh với phần tử đứng sau là phần tử cuối arr[4]. Nếu arr[3] > arr[4] thì lại tiếp tục đổi chỗ.

- Ý tưởng thuật toán đổi chỗ giữa 2 biến (arr[i] và arr[j]):

- Ta không thể gán trực tiếp arr[i] = arr[j], vì làm như vậy arr[i] sẽ nhận giá trị của arr[j] và giá trị arr[i] sẽ bị mất.
  -> Ta dùng biến phụ temp để lưu tạm giá trị của arr[i], rồi sau đó mới gán arr[i] = arr[j]
- Lúc này temp đã lưu giá trị của arr[i] nên bước cuối chỉ cần gán arr[j]=temp

```
#include<stdio.h>
main(){
    int arr[5];
    for(int i = 0; i<5; i++){
        printf("Nhap arr[%d]: \n",i);
        scanf("%d",&arr[i]);
    }                                       //Ket thuc phan nhap mang
    for(int i = 0; i<4; i++){               //Thuat toan sap xep
        for(int j=i+1; j<5; j++){
            if(arr[i] > arr[j]){
                int temp = arr[i];          //Su dung bien phu temp
                arr[i] = arr[j];
                arr[j] = temp;
            }
        }
    }
    printf("Mang sau khi sap xep: \n");     //In ket qua
    for(int i = 0; i<5; i++)    printf("%d \t",arr[i]);
}
```

```
Nhap arr[0]:
14
Nhap arr[1]:
3
Nhap arr[2]:
2000
Nhap arr[3]:
15
Nhap arr[4]:
5
Mang sau khi sap xep:
3       5       14      15      2000
--------------------------------
```

4) Nhập, xuất mảng 2 chiều:

```c
#include<stdio.h>
main(){
    int arr[3][3];
    for(int i=0; i<3; i++){                 //Bat dau phan nhap
        for(int j=0; j<3; j++){
            printf("Nhap arr[%d][%d]: \n",i,j);
            scanf("%d",&arr[i][j]);
        }
    }
    printf("Mang ban vua nhap la: \n");     //Bat dau phan xuat
    for(int i=0; i<3; i++){
        for(int j=0; j<3; j++){
            printf("%d \t",arr[i][j]);
        }
        printf("\n");                       //In dau xuong dong de tao du lieu dang bang
    }
}
```

- Mảng 2 chiều có 2 chỉ số qui định nên ta dùng 2 vòng for lồng nhau để duyệt

- Mảng 2 chiều có thể coi là 1 bảng dữ liệu -> chỉ số thứ nhất (i) có thể coi là số hàng của bảng; chỉ số thứ 2 (j) có thể coi là số cột. Như ở ví dụ trên, hết vòng for của j (tức là hết số cột) ta in dấu xuống dòng để in hàng mới.

```
Mang ban vua nhap la:
14      3       2000
15      5       2000
27      7       2020
```

Kết quả in ra sau khi nhập

C. Bài tập:

1) Nhập, xuất mảng 1 chiều. Tính tổng các phần tử có trong mảng

2) Cho người dùng nhập số nguyên N, tạo mảng có N phần tử

3) Nhập, xuất mảng 1 chiều có N phần tử. Với N là do người dùng qui định, N là số nguyên dương (0<N<100). Quá trình nhập kết thúc khi nhập vào mảng là số âm

a) Tìm GTNN của mảng, rồi in kết quả ra màn hình

b) Tìm kiếm xem trong mảng có bao nhiêu phần tử có giá trị bằng 14

4) Nhập mảng có 10 phần tử. Thêm 1 phần tử vào vị trí thứ 6 của mảng

Bài 6: Hàm

A. Lý thuyết:

1) Khái niệm:

- Trong toán học, ta có khái niệm hàm số f(x), cứ "truyền vào" hàm số 1 giá trị x_0 thì hàm sẽ trả ra kết quả $y = f(x_0)$. Thì ở trong tin học, kết quả trả về của hàm có thể là số, kí tự, hay bất kì kiểu dữ liệu nào hoặc cũng có thể không trả về kết quả.

- Công dụng:

  - Tối ưu việc sử dụng cùng 1 khối lệnh nhiều lần, tránh việc lặp code.
  - Đơn giản hóa hàm main
  - Thể hiện rõ từng chức năng (có thể qua cách đặt tên)
  - Cách sử dụng, gọi chạy hàm đơn giản

- Hàm được viết bên ngoài hàm main

- Cú pháp: [kiểu dữ liệu trả về] tên_hàm (<danh sách tham số>){

  //khối lệnh

  [return biểu_thức;] (Lệnh return chỉ có khi hàm có trả về giá trị)

  }

  (Đối với hàm không có giá trị trả về thì kiểu dữ liệu là void)

BK-DCMP

- VD1: Hàm có kiểu trả về và tham số truyền vào

```
int binhPhuong(int x){
    int y;
    y = x*x;
    return y;
}
```

- VD2: Hàm không có kiểu trả về lẫn tham số truyền vào

```
void xinChao(){
    printf("Hello World");
}
```

2) Cách sử dụng, gọi chạy hàm tại hàm main:

- Giả sử ta có 2 hàm như ở VD1, VD2 phía trên, thì ở hàm main muốn dùng 2 hàm đó ta sẽ gọi như sau:

TH1: Gọi hàm có dữ liệu trả về

```
main(){
    int a;
    a = binhPhuong(3);
    printf("%d",a);                    //Kết quả in ra là 9
}
```

TH2: Gọi hàm không có dữ liệu trả về

```
main(){
    xinChao();                         //Kết quả in ra "Hello World"
}
```

BK-DCMP

*Lưu ý:

- TH1: Coi dữ liệu trả về của hàm binhPhuong là 1 số kiểu int. Ta dùng 1 biến kiểu int để nhận giá trị mà hàm trả về rồi in ra. Hoặc ta có thể in ra trực tiếp printf("%d", binhPhuong(3));

- TH2: Đối với hàm kiểu void không có giá trị trả về thì ta gọi trực tiếp luôn

- Đối với hàm có 2 hay nhiều tham số, ta phải để ý thứ tự tham số truyền vào

B. Ví dụ tổng quan:

1) Chương trình tính giai thừa của 1 số

```c
#include<stdio.h>
int giaiThua (int x){
    int ketqua = 1;
    for(int i=1; i<=x; i++){
        ketqua *= i;
    }
    return ketqua;
}
void inKetQua(int kqua){
    printf("Ket qua la: %d",kqua);
}
main(){
    int a;
    printf("Moi ban nhap 1 so nguyen: ");
    scanf("%d",&a);
    inKetQua(giaiThua(a));
}
```

```
Moi ban nhap 1 so nguyen: 5
Ket qua la: 120
--------------------------------
```

- Ta viết tách riêng thuật toán tính giai thừa ra thành 1 hàm khiến hàm main trở nên đơn giản hơn.

2) Chương trình tính $f(x_0)$ với x_0 là số được nhập và hàm f(x) được định nghĩa như sau:

$$f(x) = \begin{cases} 7x+3 & \text{khi } x < 0 \\ x^2 - 16x + 9 & \text{khi } x \geq 0 \end{cases}$$

```c
#include<stdio.h>
float hamCuaToi(float x){
    if(x < 0) return 7*x+3;
    if(x >= 0) return x*x-16*x+9;
}
main(){
    float x;
    printf("Moi ban nhap 1 gia tri: \n");
    scanf("%f",&x);
    printf("Ket qua la: %f",hamCuaToi(x));
}
```

```
Moi ban nhap 1 gia tri:
2.1
Ket qua la: -20.189999
```

***Lưu ý:** - Phạm vi của biến: hay là biến có thể sử dụng ở đâu (mọi chỗ hay chỉ 1 hàm nhất định) phụ thuộc vào nơi khai báo biến

VD1: int abc(){

int x;

}

->Biến x chỉ sử dụng được trong hàm abc (Biến địa phương)

VD2: int x;

int abc(){....}

main(){.....}

-> Biến x có thể sử dụng được ở hàm cả 2 hàm abc lẫn hàm main (Biến cục bộ)

BK-DCMR

C. Bài tập:

1) Nhập vào bàn phím 1 số thực. Hiển thị 20 giá trị của hàm số f(x) = 3x+1 với 20 giá trị đối số giảm dần bắt đầu từ số vừa nhập với bước giảm $\Delta x = 0.01$ (giống chức năng TABLE của máy tính bỏ túi)

2) Viết hàm tính delta của phương trình bậc 2 rồi in kết quả $x = \frac{-b \pm \sqrt{b^2-4ac}}{2a}$, với a,b,c được nhập từ bàn phím. (sử dụng thư viện math.h có hàm sqrt(x) = $\sqrt{x}$; hàm pow(x,y) = x^y)

3) Viết hàm kiểm tra 1 số có là số nguyên tố hay không. Đếm xem 1 mảng có N phần tử được nhập có bao nhiêu số nguyên tố.

Bài 7: Xử lý kiểu dữ liệu kí tự và xâu kí tự

A. Lý thuyết:

1) Kiểu dữ liệu kí tự (char):

a) Nhập, xuất đối với kiểu dữ liệu char:

- Kiểu char bản chất là 1 giá trị số nguyên (int) được chuyển đổi sang kí tự dựa theo bảng mã ASCII

```
VD:   char kitu = 65;          //65 tương ứng với kí tự 'A'
      printf("%c",kitu);       //Kết quả in ra kí tự 'A'
```

- Bảng mã ASCII được viết ở trong sách giáo trình tin học đại cương tr32. Hoặc ta có thể viết vòng lặp for cho i chạy từ 0 -> 255 rồi in ra %c của i tương ứng với 256 kí tự của bảng

- Cú pháp hàm nhập scanf và hàm xuất printf đối với kiểu dữ liệu char có thể xem lại bài 1

- VD: Dùng hàm nhập getchar(); và hàm xuất putchar(); để nhập-xuất dữ liệu

```
char a;
a = getchar();      //a được gán bằng 1 kí tự được nhập
putchar(a);         //In ra màn hình giá trị của a
```

BK-DCMP

***Lưu ý:** - Khi nhập 2 hay nhiều biến kiểu char, sẽ xảy ra hiện tượng bị tràn dữ liệu. VD:

```
char a,b;
a = getchar();
b = getchar();
putchar(a);
putchar(b);
```

⇨ chỉ nhập được a, còn b sẽ không được nhập và có giá trị là NULL

- Để trống tràn dữ liệu ta thêm câu lệnh fflush(stdin); vào sau câu lệnh getchar();

```
#include<stdio.h>
main(){
    char a,b;
    a = getchar();
    fflush(stdin);
    b = getchar();
    putchar(a);
    putchar(b);
}
```

```
h
i
hi
-------
```

➔ 2 biến a và b được nhập bình thường

- Giá trị NULL hay '\0' là giá trị mặc định của kiểu dữ liệu char.
  VD: char a; -> a = '\0';

b) Các hàm xử lý kí tự trong thư viện ctype.h:

- Trước tiên muốn sử dụng được hàm bất kì, ta phải xác định kiểu trả về và tham số truyền vào (Xem lại bài 6 về hàm)

VD: hàm int toupper (int ch) //Kiểu trả về và tham số đều là int nhưng chúng ta ngầm hiểu là char, vì bản chất char là số

-> char ch = 'a'; char c;

c = toupper(ch);

putchar(c); //Kết quả in ra là 'A'

BK-TCMP

- Hàm int toupper(int ch): Dùng để chuyển một kí tự chữ cái thường thành hoa

- Hàm int tolower(int ch): Dùng để chuyển một kí tự chữ hoa thành thường

- Hàm int isalpha(int ch): Kiểm tra kí tự truyền vào có phải là chữ hay ko, nếu đúng trả về giá trị khác 0, nếu sai trả về 0

- Hàm int isdigit(int ch): Kiểm tra kí tự truyền vào có phải là số hay ko

- Hàm int islower(int ch): Kiểm tra kí tự truyền vào có phải là chữ thường ko

- Hàm int isupper(int ch): Kiểm tra kí tự truyền vào có phải là chữ hoa ko

- Hàm int isspace(int ch): Kiểm tra kí tự truyền vào có phải là dấu cách ko

***Lưu ý:** - Muốn sử dụng các hàm trên ta nhớ khai báo thư viện ctype.h bằng câu lệnh #include<ctype.h>

2) Kiểu dữ liệu xâu (chuỗi):

a) Khái niệm:

- Kiểu xâu kí tự chính là mảng kí tự (char xau[ ]), dùng để lưu trữ nhiều kí tự viết liên tiếp nhau

VD: char xau[7] = "Ha Noi"; //Cách khai báo xâu có giá trị ban đầu

➔ Dữ liệu được lưu như sau:

'H'	'a'	' '	'N'	'o'	'i'	'\0'

***Lưu ý:** - Dấu nháy đơn thể hiện kí tự, dấu nháy kép thể hiện xâu. VD:

char c = 'a'; char xau[10] = "Bac Giang" ;

- Độ dài xâu "Bac Giang" là 9 kí tự nhưng kí tự NULL ('\0') được thêm vào cuối nên khi khai báo mảng char phải có lớn hơn hoặc bằng 10 phần tử. Kí tự NULL được tự động thêm vào cuối ở bất kì xâu nào.

BK-DCMP

b) Nhập, xuất đối với kiểu dữ liệu xâu:

- Để nhập dữ liệu cho xâu ta có 2 hàm scanf() và gets()

- Cú pháp hàm scanf: scanf("%s", ten_xau);
  ➔ Hàm scanf của xâu kí tự không cần thêm dấu '&'.
  Hàm scanf không đọc được kí tự dấu cách. (nhập "Ha Noi" -> "Ha")
- Cú pháp hàm gets: gets(ten_xau);
  ➔ Hàm gets khác hàm scanf là đọc được dấu cách nên được dùng nhiều hơn đối với kiểu xâu

*Lưu ý: - Khi nhập 2 hay nhiều xâu nếu có hiện tượng tràn dữ liệu ta lại dùng câu lệnh fflush(stdin);

- Để xuất dữ liệu ta có 2 hàm printf() và puts()

- Cú pháp hàm printf: printf("%s", ten_xau);
- Cú pháp hàm puts: puts(ten_xau);
- Điểm khác nhau giữa 2 hàm này là hàm puts sau khi in xâu sẽ in thêm dấu xuống dòng. Còn printf nếu muốn xuống dòng phải viết thêm '\n'

c) Các hàm xử lý xâu trong thư viện string.h:

- Hàm strlen(ten_xau): Khi truyền 1 xâu vào, hàm sẽ trả về độ dài của xâu kí tự đó. VD: printf("%d",strlen("Ha Noi")); ->kết quả là 6 ('\0' ở cuối ko đc đếm)

- Hàm strcpy(xau_dich, xau_nguon): Hàm sẽ sao chép nội dung ở xâu nguồn rồi ghi lên xâu đích. Vì xâu là mảng char và ta ko thể gán xau1 = xau2; nên hàm này khá thông dụng đối với kiểu xâu

- Hàm strcmp(xau1, xau2): Dùng để so sánh 2 xâu. Hàm trả về

- giá trị < 0 nếu xau1 $<$ xau 2
- giá trị 0 nếu xau1 = xau 2
- giá trị > 0 nếu xau1 $>$ xau2

cách so sánh: xâu nào có độ dài lớn hơn thì lớn hơn, trường hợp bằng nhau thì lần lượt so sánh từng kí tự của xâu dựa theo so sánh số của bảng mã ASCII tương

BK-DCVP

- Hàm strcat(xau_dich, xau_nguon): Ghép xâu nguồn vào ngay sau xâu đích. Hàm trả về xâu mới là xâu được ghép nối

- Hàm strchr(xau, kitu): Dùng để tìm kiếm vị trí của kí tự trong xâu. Nếu có kí tự hàm trả về con trỏ tới kí tự đầu tiên tìm được. Nếu không trả về con trỏ NULL

- Hàm strstr(xau1, xau2): Dùng để tìm kiếm vị trí xau2 trong xau1. Trả về tương tự hàm strchr

- Hàm int atoi(xau1): Dùng để chuyển xau1 thành số nguyên kiểu int. Nếu đổi thành công hàm trả về giá trị số chuyển đổi được. Nếu không trả về 0

- Hàm long int atol(xau1): Chuyển xau1 thành số nguyên kiểu long int. Trả về tương tự hàm atoi

- Hàm double atof(xau1): Chuyển xau1 thành số thực. Trả về tương tự hàm atoi, atol

***Lưu ý:** - Để sử dụng được các hàm atoi, atol, atof, ta khai báo thư viện stdlib.h

- Để sử dụng các hàm còn lại, ta khai báo thư viện string.h

B. Ví dụ tổng quan:

1) Chương trình chuyển đổi 1 xâu kí tự thường thành kí tự hoa:

```c
#include<stdio.h>
#include<ctype.h>
#include<string.h>
main(){
    char str[100];
    printf("Moi ban nhap 1 xau ki tu: ");
    gets(str);
    for(int i = 0; i<strlen(str); i++){
        str[i] = toupper(str[i]);
    }
    puts(str);
}
```

```
Moi ban nhap 1 xau ki tu: abc def
ABC DEF

--------------------------------
```

RV-DCMP

- Thư viện ctype.h để dùng hàm toupper, thư viện string.h để dùng hàm strlen. Coi xâu str như 1 mảng, ta dùng vòng for duyệt mảng và gán từng phần tử tức là từng kí tự thành kí tự hoa bằng hàm toupper.

2) Chương trình kiểm tra trong 1 xâu kí tự có kí tự số hay không

```c
#include<stdio.h>
#include<ctype.h>
#include<string.h>
main(){
    char str[100];
    int check =0 ;
    printf("Moi ban nhap 1 xau ki tu: \n");
    gets(str);
    for(int i = 0; i<strlen(str); i++){
        if(isdigit(str[i])){
            check = 1;
            break;
        }
    }
    if(check == 1) printf("Xau co ki tu so");
    else printf("Xau khong co ki tu so");
}
```

```
Moi ban nhap 1 xau ki tu:
abc 123 def
Xau co ki tu so
--------------------------------
```

- Ban đầu gán biến check = 0, nếu duyệt mảng thấy phần tử thứ i của xâu str (str[i]) là số thì gán biến check = 1

C. Bài tập:

1) Viết chương trình đảo ngược kí tự của 1 xâu được nhập:
VD: hust.edu.vn -> n v . u d e . t s u h

2) Viết chương trình chuyển đổi xâu kí tự số thành số. (Hàm atoi, atof)
VD: “12345” -> số nguyên int 12345

3) Cho người dùng nhập tên rồi in ra “Xin chao + ten”. Nếu tên chứa số, kí tự đặc biệt không phải là chữ thì cho nhập lại.

Bài 8: Kiểu dữ liệu cấu trúc (struct)

A. Lý thuyết:

1) Khái niệm:

- Ta được biết mảng là tập hợp gồm các phần tử cùng kiểu dữ liệu, vậy nếu như ta muốn nhóm một tập hợp gồm các phần tử khác kiểu dữ liệu, ta sẽ dùng đến struct
- Kiểu dữ liệu struct là kiểu dữ liệu do người viết tự qui định và đặt tên
- Các thành phần trong cấu trúc được gọi là trường (thuộc tính)

VD: Kiểu dữ liệu SinhVien gồm các trường như MSSV, họ tên,....

2) Khai báo và cách sử dụng struct:

- Cú pháp:

```
struct ten_cau_truc{
    <Khai báo các trường dữ liệu>
};
```

VD:

```
struct SinhVien{
    int MSSV;
    char hoVaTen[32];
    float gpa;
};
```

- Cách sử dụng: Sau khi khai báo (định nghĩa) kiểu dữ liệu struct như ở ví dụ trên, muốn sử dụng ta chỉ việc coi SinhVien như 1 kiểu dữ liệu

```c
#include<stdio.h>
struct SinhVien{
    int MSSV;
    char hoVaTen[32];
    float gpa;
};
main(){
    SinhVien a = {20200730, "Nguyen Van A", 3.69};
    printf("%f",a.gpa);
}
```

- Biến a có kiểu dữ liệu SinhVien được khởi tạo với giá trị ban đầu

DK-DCMP

- Chú ý ta dùng dấu chấm để truy cập giá trị của từng trường. Ở VD trên muốn in ra gpa của sinh viên a, ta dùng a.gpa. Tương tự với a.MSSV, a.hoVaTen

tên_biến_cấu_trúc.tên_trường

B.Ví dụ tổng quan:

1) Cách nhập dữ liệu cho 1 sinh viên:

```
#include<stdio.h>
struct SinhVien{
    int MSSV;
    char hoVaTen[32];
    float gpa;
}a;
void nhapSinhVien(){
    printf("Nhap MSSV: \n");
    scanf("%d",&a.MSSV);
    printf("Nhap ten SV: \n");
    fflush(stdin); gets(a.hoVaTen);
    printf("Nhap gpa: \n");
    scanf("%f",&a.gpa);
}
void hienThiSinhVien(){
    printf("MSSV \t\t Ten \t\t GPA\n");
    printf("%d \t %s \t %f",a.MSSV,a.hoVaTen,a.gpa);
}
main(){
    nhapSinhVien();
    hienThiSinhVien();
}
```

- Đầu tiên ta khởi tạo 1 struct SinhVien và a là biến có kiểu dữ liệu SinhVien. Có thể viết gộp struct SinhVien{...}a; (Biến a là biến toàn cục)

- Tiếp theo viết tách riêng 2 hàm nhapSinhVien và hàm hienThiSinhVien để hàm main được gọn hơn. Thực hiện 2 chức năng riêng biệt

- Trong hàm nhapSinhVien(), ta chú ý dùng đúng cú pháp nhập kiểu dữ liệu đối với từng trường
- Hàm hiển thị sẽ thực hiện chức năng hiển thị ra toàn bộ thông tin của sinh viên a

- Ta được kết quả sau khi nhập:

```
Nhap MSSV:
20200730
Nhap ten SV:
Nguyen Van A
Nhap gpa:
3.69
MSSV            Ten             GPA
20200730        Nguyen Van A    3.690000
--------------------------------
```

2) Nâng cấp ví dụ 1 lên thành 1 mảng sinh viên:

```c
#include<stdio.h>
#include<string.h>
struct SinhVien{
    int MSSV;
    char hoVaTen[32];
    float gpa;
};
SinhVien arr[10];
int soLuongSV = 0;
void nhapSinhVien(){
    //Tu nhap sinh vien thu nhat
    arr[0].MSSV = 20200730;
    strcpy(arr[0].hoVaTen, "Nguyen Van A");
    arr[0].gpa = 3.69;
    soLuongSV++;

    //Tu nhap sinh vien thu hai
    arr[1].MSSV = 20200731;
    strcpy(arr[1].hoVaTen, "Nguyen Van B");
    arr[1].gpa = 3.96;
    soLuongSV++;

    //Nhap sinh vien thu ba
    printf("Nhap MSSV: \n");
    scanf("%d",&arr[soLuongSV].MSSV);
    printf("Nhap ten SV: \n");
    fflush(stdin); gets(arr[soLuongSV].hoVaTen);
    printf("Nhap GPA: \n");
    scanf("%f",&arr[soLuongSV].gpa);
    soLuongSV++;
}
void hienThiSinhVien(){
    printf("MSSV \t\t Ten \t\t GPA\n");
    for(int i=0; i<soLuongSV; i++){
        printf("%d \t %s \t %f\n",arr[i].MSSV,arr[i].hoVaTen,arr[i].gpa);
    }
}
main(){
    nhapSinhVien();
    hienThiSinhVien();
}
```

- Khai báo một mảng sinh viên có 10 phần tử, và biến đếm số lượng sinh viên có trong mảng (ban đầu bằng 0).

- Hàm nhapSinhVien ta tự nhập 2 sinh viên A, B. Còn sinh viên thứ 3 nhập theo cách lấy dữ liệu từ bàn phím

- Chú ý trong phần nhập cứ mỗi 1 sinh viên được thêm vào mảng thì số lượng sinh viên được tăng thêm 1
- Chỉ số của sinh viên trong mảng chính là số lượng sinh viên -> nhập dữ liệu vào arr[soLuongSV]

- Hàm hiển thị sinh viên sẽ dùng vòng lặp duyệt mảng sinh viên và in toàn bộ thông tin SV có trong mảng

- Kết quả ta in ra được danh sách sinh sinh viên trong mảng

```
Nhap MSSV:
20200801
Nhap ten SV:
Nguyen Thi C
Nhap GPA:
3.66
MSSV            Ten             GPA
20200730        Nguyen Van A    3.690000
20200731        Nguyen Van B    3.960000
20200801        Nguyen Thi C    3.660000

--------------------------------
```

***Lưu ý:** - Sử dụng đúng cú pháp theo từng kiểu dữ liệu của trường

C. Bài tập:

1) Tạo 1 cấu trúc phanSo gồm 2 trường {int tuSo; int mauSo}

a) Viết 1 hàm nhận tuSo với mauSo là tham số truyền vào. Hàm này trả ra kết quả của phân số dưới dạng số thực = tuSo/mauSo

b) Tạo 1 mảng phanSo gồm 5 phần tử. Nhập dữ liệu cho mảng (có thể tự nhập hoặc nhập từ bàn phím). Tính tổng của các phân số rồi in ra kết quả

c) Nhập tử số và mẫu số từ bàn phím. Tìm trong mảng có bao nhiêu phân số có giá trị bằng với phân số vừa nhập

2) Thông tin về 1 cuốn sách được lưu trong cấu trúc gồm: Mã sách (số nguyên), Tên sách (xâu không quá 32 kí tự), Tác giả (xâu không quá 16 kí tự), Giá sách (số thực).

a) Tạo mảng sách và lưu thông tin của 3 cuốn sách (tự nhập hoặc nhập từ bàn phím)

b) Hiển thị danh sách của mảng sách vừa tạo

c) In ra những cuốn sách của tác giả "Felik". Nếu không có in ra "Khong co cuon sach nao"

d) Sắp xếp danh sách theo thứ tự giá tiền tăng dần rồi in ra màn hình danh sách đó

Bài 9: Con trỏ

A. Lý thuyết:

1) Khái niệm và cách khai báo con trỏ:

- Khi khai báo 1 biến để lưu trữ dữ liệu, ta luôn có hai đặc tính:

- Giá trị của biến. VD: printf("%d",a); -> In ra giá trị của biến a
- Địa chỉ của biến. VD: scanf("%d",&a); ->&a chính là địa chỉ của biến a

- Như vậy, con trỏ được tạo ra để tương tác giữa các địa chỉ của biến từ đó cũng có thể lấy được giá trị của biến.

- Cú pháp: *Kiểu_dữ_liệu *tên_con_trỏ;*

VD: int *p, a =3; (Kí tự * thể hiện là p là một biến con trỏ)

p = &a; (Gán giá trị cho p bằng địa chỉ của biến a)

***Lưu ý:** - Con trỏ kiểu void có thể nhận địa chỉ của 1 biến thuộc bất kì kiểu dữ liệu nào

- Phân biệt toán tử * và &:

- Toán tử * (*p): là toán tử trả về giá trị của 1 biến được trỏ tới bởi con trỏ p
- Toán tử & (&a): là toán tử trả về địa chỉ của biến a

VD: int a = 14;

int *p = &a;

BK-DCMP

```
printf("Gia tri cua a la: %d",*p);
```

➔ Toán tử dấu & gán cho con trỏ p chứa địa chỉ của biến a. Khi cần lấy giữ liệu của biến a, ta có thế lấy qua con trỏ p (*p)

2) Mối quan hệ giữa con trỏ và mảng một chiều:

- Giả sử ta có con trỏ p trỏ tới phần tử đầu tiên của mảng arr (&arr[0]). Như vậy các phần tử tiếp theo lần lượt được trỏ tới bởi con trỏ (p+1), (p+2),...

VD: int a[10], *p;

p = a; (Không cần dấu &, câu lệnh tương đương với p = &a[0])

Như vậy, (p+1) sẽ trỏ tới a[1] và có giá trị là *(p+1)

➔ Tổng quát, (p+i) sẽ trỏ tới a[i] và có giá trị là *(p+i)

- VD: Dùng con trỏ để duyệt mảng một chiều

```
int i, *p, arr[5] = {1,2,3,4,5};
p = arr;
for(i = 0; i<5; i++){
        printf("%d \t",*(p+i));
}
```

➔ Kết quả in ra "1 2 3 4 5"

3) Công dụng của con trỏ:

- Đối với hàm: Nếu ta khai báo biến ở trong một hàm thì phạm vi của biến chỉ được sử dụng trong hàm đó. Hàm khác muốn sử dụng có thể truyền con trỏ làm tham số

- Đối với mảng: Ta chỉ cần có địa chỉ của phần tử đầu tiên trong mảng là đã có thể duyệt tất cả phần tử trong mảng. Tương tự đối với xâu kí tự

BK-DCMP

B. Ví dụ tổng quan:

1) Đổi chỗ 2 số dùng hàm kết hợp với con trỏ:

```c
#include<stdio.h>
void doiCho(int *p1, int *p2){
    int temp;
    temp = *p1;
    *p1 = *p2;
    *p2 = temp;
}
main(){
    int a = 3, b = 4;
    int *p1, *p2;
    p1 = &a;
    p2 = &b;
    printf("Truoc khi doi cho: %d \t %d \n",a,b);
    doiCho(p1, p2);
    printf("Sau khi doi cho: %d \t %d \n",a,b);
}
```

```
Truoc khi doi cho: 3     4
Sau khi doi cho: 4       3
```

- Biến a, b ko truyền trực tiếp vào hàm doiCho() nhưng hàm đổi chỗ vẫn thay đổi được giá trị của 2 biến này

- Trong trường hợp truyền thẳng giá trị 2 biến a, b vào hàm doiCho():

```c
#include<stdio.h>
void doiCho(int x, int y){
    int temp;
    temp = x;
    x = y;
    y = temp;
}
main(){
    int a = 3, b = 4;
    printf("Truoc khi doi cho: %d \t %d \n",a,b);
    doiCho(a, b);
    printf("Sau khi doi cho: %d \t %d \n",a,b);
}
```

```
Truoc khi doi cho: 3     4
Sau khi doi cho: 3       4
```

➔ Trường hợp này ko đổi chỗ được 2 giá trị a,b do phạm vi biến của a,b hàm doiCho() không nhận

2) Chương trình xóa kí tự dấu cách của một xâu được nhập:

```
#include<stdio.h>
#include<string.h>
#include<ctype.h>
void delSpace(char *xau){
    for(int i = 0; i<strlen(xau); i++){
        if(isspace(xau[i])){
            for(int j = i; j<strlen(xau); j++){
                xau[j] = xau[j+1];
            }
            i--;
        }
    }
}
main(){
    char str[32], *p;
    gets(str);
    p = str;
    delSpace(p);
    puts(str);
}
```

```
      Nguyen          Van       A
NguyenVanA
```

- Ý tưởng thuật toán: Duyệt từng kí tự của xâu, nếu thấy kí tự dấu cách sẽ xóa kí tự đó. Thuật toán xóa của xâu tương tự thuật toán xóa đi 1 phần tử của mảng 1 chiều.

- Biến xâu str coi như mảng khi truyền vào hàm không cần dấu &

C. Bài tập:

1) Cộng hai số sử dụng con trỏ

2) Viết chương trình tính tổng các phần tử trong mảng bởi sử dụng con trỏ

Bài 10: Bài tập tổng hợp

A. Mẫu câu hỏi:

1)

Lập chương trình thực hiện các công việc sau:
a) Nhập một số nguyên N (0 < N < 10) từ bàn phím.
b) Nhập một mảng có N số nguyên từ bàn phím. In ra màn hình mảng vừa nhập như một dãy số.
c) In ra số lẻ hoàn toàn
Số lẻ hoàn toàn là số mà các chữ số của nó đều <u>không chia hết</u> cho 2.
Ví dụ n=1357 là số "lẻ hoàn toàn".

2)

a) Viết hàm tính phương trình: $f(x)=x^5+\sqrt[5]{x}$
b) Viết chương trình cho phép nhập 3 số thực a, b, c và tính trung bình cộng của f(a), f(b), f(c)

3)

Viết chương trình thực hiện các công việc sau:
Nhập vào **số nguyên N (0 < N < 100000)**.
Đưa ra kết quả **số ngược** của **N**.
Ví dụ: 1234 có số ngược là **4321**

4)

Viết chương trình thực hiện các công việc sau:
Nhập vào **số nguyên dương N**.
Kiểm tra số vừa nhập có phải **số lùi** không.
Hiển thị thông báo ra màn hình.
Biết: Số lùi là số mà các chữ số theo thứ tự giảm dần
Ví dụ: abcd là số lùi khi **a>b>c>d**.

5)

Viết chương trình tính tiền điện hàng tháng:
Nhập một **số nguyên 0<N<10000** là số kW điện đã sử dụng.
Tính số tiền gia đình phải trả chưa bao gồm thuế ở bảng dưới đây

Các bậc	Giá tiền
Bậc 1: Cho kWh từ 0-50	1484
Bậc 2: Cho kWh từ 51-100	1533
Bậc 3: Cho kWh từ 101-200	1786
Bậc 4: Cho kWh từ 201-300	2242
Bậc 5: Cho kWh từ 301-400	2503
Bậc 6: Cho kWh từ 401 trở lên	2587

6)

Lập chương trình thực hiện các công việc sau:
Nhập một **số nguyên N ≥ 0** bất kì thuộc hệ đếm **cơ số 2** từ bàn phím.
Chuyển số đó thành số trong hệ đếm**cơ số 10**
In kết quả ra màn hình.

7)

Lập chương trình thực hiện các công việc sau:
a) Nhập một giá trị thực **x *radian*** **(0 ≤ x < 10)** từ bàn phím.
b) Tính **sin(x)** với độ chính xác **0.0001** dựa vào công thức sau:

$$\sin(x) = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \ldots + (-1)^n \frac{x^{2n+1}}{(2n+1)!}$$

Gợi ý: Sử dụng vòng lặp **while** để tính với điều kiện dừng khi $\frac{x^{2n+1}}{(2n+1)!} < epsilon = 0.0001$

8)

Lập chương trình thực hiện các công việc sau:
a) Nhập một **số nguyên N (0 < N < 10)** từ bàn phím.
b) Nhập một mảng có **N số nguyên** từ bàn phím. In ra màn hình mảng vừa nhập như một dãy số.
c) In ra **vị trí** các **số nguyên tố** và đếm số lượng **số nguyên tố nhỏ hơn 2016**.

B. Bài tập lớn: (phần phụ)

Đề tài 1: Viết game

- VD tham khảo: Game giả lập chiếc nón kì diệu

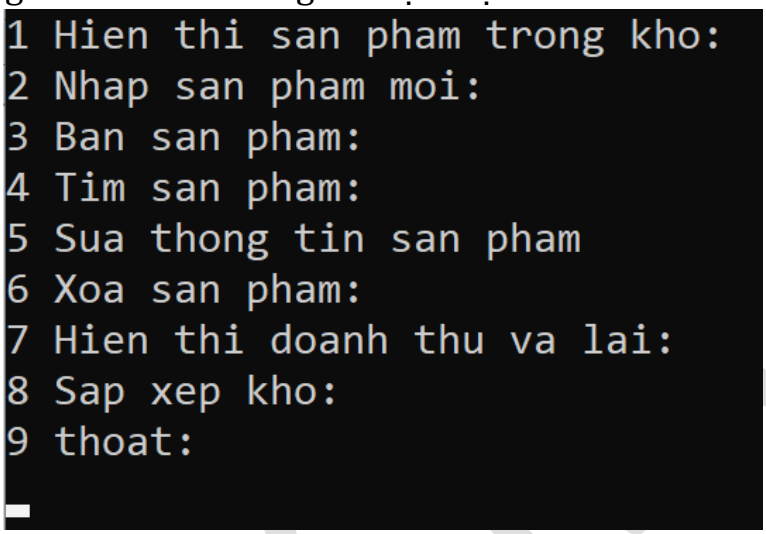
```
Diem: 0
***** ******** ** *** **** ******** ***** *** *** **** ***
Ban doan chu cai nao?
```

Sau khi đoán chữ ‘t’

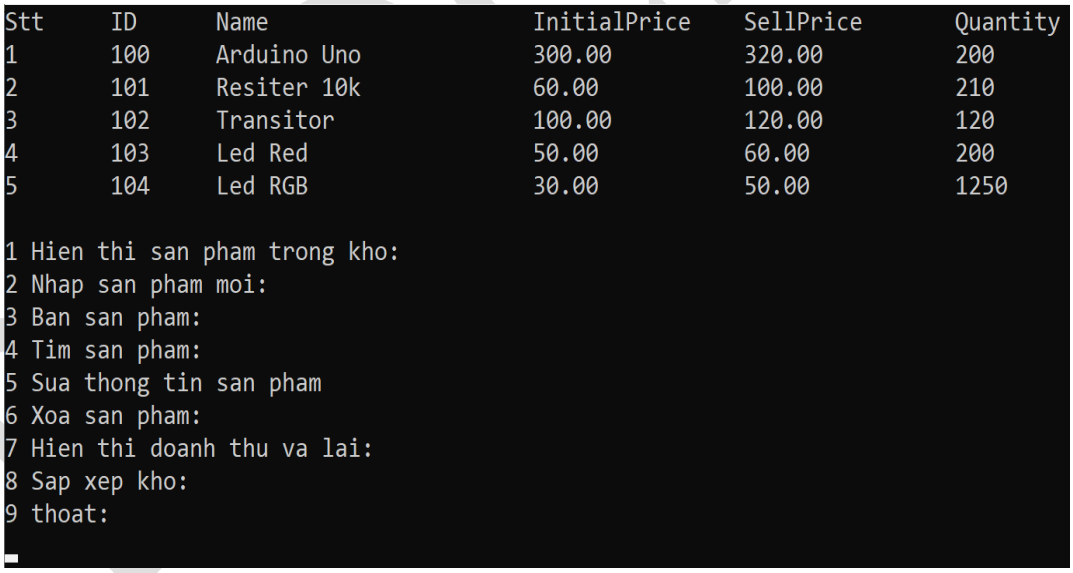
```
Co 3 chu t
Diem: 1000
***** ***t**** t* *** ***t ******** ***** *** *** **** ***
Ban doan chu cai nao?
```

Đề tài 2: Viết ứng dụng quản lý

- VD tham khảo: Ứng dụng quản lý kho hàng
  - Ban đầu có menu tạo bởi switch-case cho người dùng nhập từ 1-9 tương ứng với các chức năng sẽ thực hiện



  - Sau khi nhập 1, hiện danh sách trong kho



  ➔ Tương tự với các chức năng còn lại