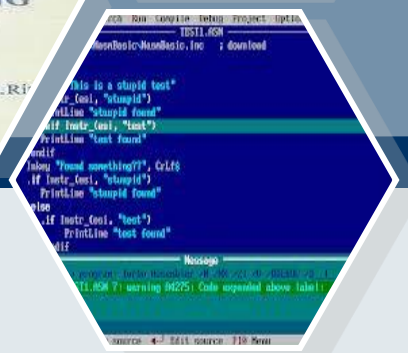
HUTECH
Đại học Công nghệ Tp.HCM

KHOA CÔNG NGHỆ THÔNG TIN

Bài giảng:

KỸ THUẬT LẬP TRÌNH



Bài 1:

MẢNG MỘT CHIỀU

C Ngôn ngữ lập trình số 1 thế giới

Giảng viên: Th.S Dương Thành Phết

Email: phetcm@gmail.com

Website: http://www.thayphet.net

Mobile: 0918158670

MỤC TIÊU

- ✓ Trình bày được khái niệm về kiểu dữ liệu mảng, mảng ký tự (chuỗi k1y tự) và các ứng dụng;
- ✓ Thực hiện được khai báo biến kiểu mảng và các phép toán trên các phần tử của mảng;
- ✓ Hiện thực được các giải thuật trên mảng 1 chiều như: Tìm kiếm, tính toán, sắp xếp, thêm/xóa phần tử...
- ✓ Thực hiện các giải thuật trên mảng 1 chiều các ký tự: Tìm kiếm, sao chép, cắt, ghép,...

NỘI DUNG

1. Khái niệm mảng 1 chiều
2. Chuỗi ký tự - mảng 1 chiều các ký tự
3. Bài tập

1.1. KHÁI NIỆM MẢNG 1 CHIỀU

1.1.1. Mảng một chiều

1.1.2. Cách khai báo mảng một chiều

1.1.3. Truy cập vào các phần tử của mảng

1.1.4. Nhập dữ liệu cho mảng một chiều

1.1.5. Xuất dữ liệu cho mảng một chiều

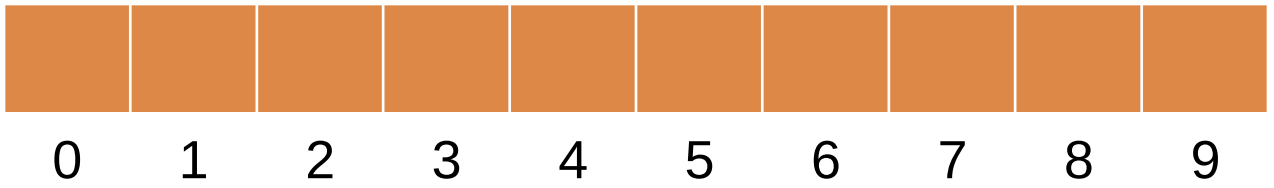
1.1.6. Một vài thuật toán trên mảng một chiều

1.1.1. MẢNG MỘT CHIỀU

- ✓ Là tập hợp các phần tử cố định có cùng kiểu dữ liệu
- ✓ Kiểu dữ liệu của mảng là:
  - Kiểu dữ liệu chuẩn
  - Kiểu do người dùng định nghĩa
  - Kiểu mảng (mỗi phần tử là 1 mảng)
- ✓ Kiểu dữ liệu mảng được sử dụng khá phổ biến để lưu các tập giá trị có kiểu kiểu dữ liệu
  - Mảng 1 chiều
  - Mảng nhiều chiều (>=2 chiều)

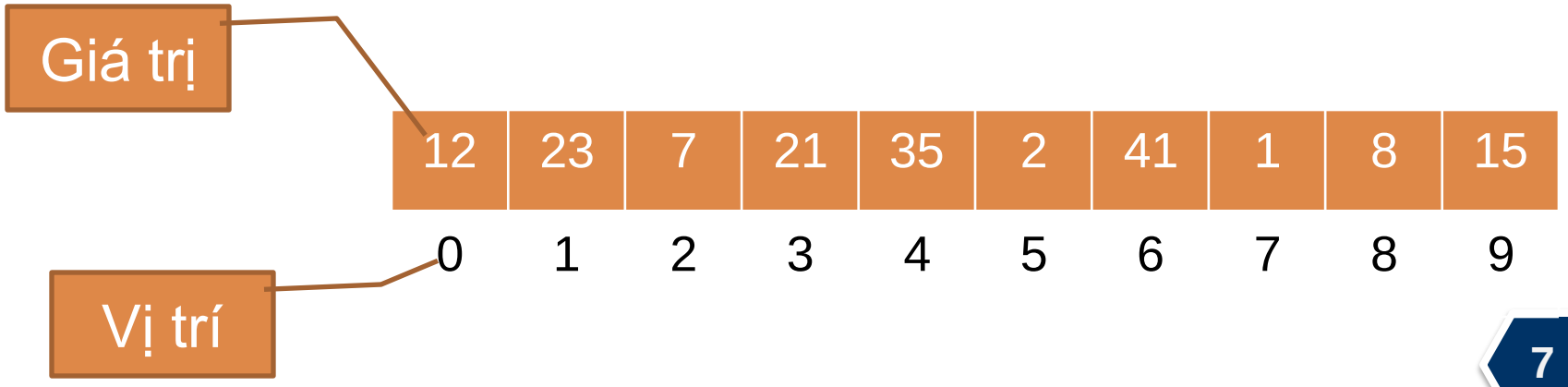
1.1.1. MẢNG MỘT CHIỀU

- ✓ Mảng là kiểu dữ liệu được sử dụng rất phổ biến như:
  - Cần quản lý một danh sách sinh viên trong một lớp,
  - Hay lưu trữ các từ khóa của ngôn ngữ lập trình C
  - …
- ✓ Kích thước của mảng là số phần tử của mảng. Được biết khi khai báo mảng..



1.1.1. MẢNG MỘT CHIỀU

- Mảng 1 chiều là một biến bao gồm nhiều biến thành phần được cấp phát bộ nhớ liên tục.
- 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 (Vị trí).



1.1.2. CÁCH KHAI BÁO MẢNG MỘT CHIỀU

1.1.2.1. Khai báo tường minh (số phần tử xác định):

< Kiểu dữ liệu > < Tên mảng >[< Số p.tử tối đa>] ;

int a[100]; //Khai bao mang so nguyen a gom 100 phan tu

float b[50]; //Khai bao mang so thuc b gom 50 phan tu

char str[30]; //Khai bao mang ky tu str gom 30 ky tu

1.1.2. CÁCH KHAI BÁO MẢNG MỘT CHIỀU

1.1.2.2. Khai báo không tường minh:

Cách 1: Vừa khai báo vừa gán giá trị ban đầu

```
int a[5] = {3, 6, 8, 1, 12};
```

→ a[0] = 3, a[1] = 6, a[2] = 8, …

```
int a[10] = {0};
```

→ a[0]=a[1]=a[2]=a[3]=…=a[9]=0

Cách 2. Khai báo mảng là tham số hình thức của hàm

Không cần chỉ định số phần tử của mảng là bao nhiêu.

Ví dụ :

```
void  Nhapmang ( int a[ ], int n )
```

1.1.3. TRUY CẬP VÀO CÁC PHẦN TỬ CỦA MẢNG

✓ Các phần tử trong mảng sẽ cùng tên khác chỉ số.

Ví dụ mảng A có 10 phần tử:

A[0]➔ 12

…

A[5] ➔ 2

…

A[9] ➔ 15

1.1.4. NHẬP DỮ LIỆU CHO 1 CHIỀU

- ✓ Khai báo mảng a 100 phần tử là các số nguyên, máy tính sẽ cấp phát 200 byte để lưu trữ : int a [100]

```
// Nhập từng phần tử của mảng số nguyên
for ( int i =0 ; i<n ; i++)
{
        printf ("Nhap a[%d] :", i );
        scanf ("%d", & a[ i ]);
}
```

Hàm nhập mảng 1 chiều a có n số nguyên

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

1.1.4. NHẬP DỮ LIỆU CHO 1 CHIỀU

✓ Phát sinh mảng 1 chiều các số nguyên ngẫu nhiên

- Khởi tạo số ngẫu nhiên dùng để giảm bớt công đoạn nhập số cho mảng một chiều.
- Để khởi tạo số ngẫu nhiên ta cần biết đến hàm srand() và rand() trong stdlib.h. time.h (trong C++ trong iostream, time.h)

Hàm phát sinh mảng 1 chiều a có n số nguyên ngẫu nhiên:

```
void Phatsinhmang (int a[], int n)
{
        srand(time(NULL)); //moi lan chay 1 gia tri khac nhau
        for (int i = 0; i < n; i ++)
                a[i] = rand();
}
```

1.1.5. XUẤT DỮ LIỆU CHO 1 CHIỀU

✓ Lần lượt duyệt từng phần tử trong mảng và xuất ra.

```
// Xuất từng phần tử của mảng số nguyên
for ( int i =0 ; i<n ; i++)
        printf ("%3d", a[ i ] );
```

Hàm xuất mảng 1 chiều a có n số nguyên

```
void Xuatmang(int a[], int n)
{
        for ( int i =0 ; i<n ; i++)
                printf ("%3d", a[ i ] );
}
```

1.1.6. MỘT SỐ THUẬT TOÁN TRÊN MẢNG 1 CHIỀU

✓ Xuất các phần tử thỏa điều kiện

```
void LietKeXXX(int a[], int n)
{
    for (int i = 0; i<n; i++)
        if (a[i] thỏa điều kiện)
            Xuất a[i];
}
```

```
void LietKeXXX(int a[], int n, int x)
{
   for (int i = 0; i<n; i++)
        if (a[i] thỏa điều kiện so với x)
            Xuất a[i];
}
```

1.1.6. MỘT SỐ THUẬT TOÁN TRÊN MẢNG 1 CHIỀU

✓ Đếm các phần tử thỏa điều kiện

```
int DemXXX(int a[], int n){
    int d = 0;
    for (int i = 0; i<n; i++)
        if (a[i] thỏa điều kiện)
            d++;
        return d;
}
```

1.1.6. MỘT SỐ THUẬT TOÁN TRÊN MẢNG 1 CHIỀU

✓ Tìm kiếm

Tìm và trả về vị trí phần tử có giá trị lớn nhất

```
int TimVTMax(int a[], int n)
{
    int vtmax = 0;
    for (int i = 0; i < n; i++)
        if (a[i] > a[vtmax])
            vtmax = i;
        return vtmax;
}
```

1.1.6. MỘT SỐ THUẬT TOÁN TRÊN MẢNG 1 CHIỀU

✓ Tìm kiếm

Tìm vị trí phần tử có giá trị x *(nếu x không xuất hiện trong mảng trả về -1)*

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

1.1.6. MỘT SỐ THUẬT TOÁN TRÊN MẢNG 1 CHIỀU

✓ Kiểm tra mảng có thỏa điều kiện cho trước

TH1: Kiểm tra tồn tại một phần tử trong mảng thỏa điều kiện nào đó cho trước.

```
bool KiemTraTonTaiXXX(int a[], int n)
{
        for (int i = 0; i<n; i++)
                if (a[i] thỏa điều kiện)
                        return true;
        return false;
}
```

```
bool KiemTraTonTaiLe(int a[], int n)
{
    foreach (int giatri in a)
            if (giatri % 2 != 0)
                return true;
    return false;
}
```

1.1.6. MỘT SỐ THUẬT TOÁN TRÊN MẢNG 1 CHIỀU

✓ Kiểm tra mảng có thỏa điều kiện cho trước

TH2: Kiểm tra tất cả các phần tử thỏa điều kiện nào đó cho trước.

```
bool  KiemTraXXX(int a[], int n)
{
        for (int i = 0; i<n; i++)
                if (a[i] không thỏa điều kiện)
                        return false;
        return true;
 }
```

```
bool KiemTraToanAm(int a[], int n)
{
      for (int i  = 0; i<n; i++)
              if (a[i] >= 0)
                      return false;
      return true;
}
```

1.1.6. MỘT SỐ THUẬT TOÁN TRÊN MẢNG 1 CHIỀU

✓ **Tính tổng các phần tử thỏa điều kiện:**

```
int TongXXX(int a[], int n)
{
    int s = 0;
    for (int i = 0; i<n; i++)
        if (a[i] thỏa điều kiện)
            s += a[i];
    return s;
}
```

```
int TongLe(int a[], int n)
{
    int s = 0;
    for (int i = 0; i<n; i++)
        if (a[i] %2!=0)
            s += a[i];
    return s;
}
```

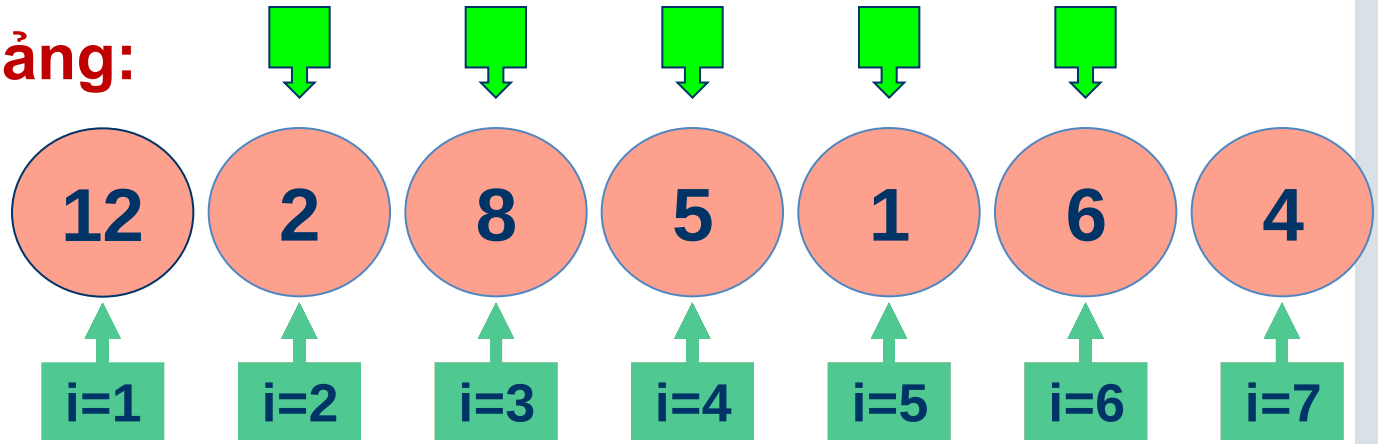
1.1.6. MỘT SỐ THUẬT TOÁN TRÊN MẢNG 1 CHIỀU

✓ **Tính trung bình cộng các phần tử thỏa điều kiện:**

```
float  TrungBinhXXX(int a[], int n)
{
    int s = 0;
    int d = 0;
    for (int i = 0; i<n; i++)
        if (a[i] thỏa điều kiện)
        {
            s += giatri;
            d ++;
        }
    if (d==0)
        return 0;
    return (float) s / d;
}
```

1.1.6. MỘT SỐ THUẬT TOÁN TRÊN MẢNG 1 CHIỀU

Sắp xếp mảng:




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

1.1.6. MỘT SỐ THUẬT TOÁN TRÊN MẢNG 1 CHIỀU

Chèn 1 phần tử vào mảng:

Chèn phần tử tại vị trí cho trước:

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

1.1.6. MỘT SỐ THUẬT TOÁN TRÊN MẢNG 1 CHIỀU

Xóa 1 phần tử vào mảng:

Xóa phần tử tại vị trí cho trước:

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

1.1.6. MỘT SỐ THUẬT TOÁN TRÊN MẢNG 1 CHIỀU

Tách mảng:

Tách mảng thành 2 mảng chẵn lẻ:

```
void Tachmang(int a[], int b[], int c[], int n, int &m, int &k)
{
        int m=0,k=0 ;
        for (int i=0; i<n; i++)
        {
                if (a[i]%2==0){
                        b[m]=a[i];
                        m++;
                }
                else{
                        c[k]=a[i];
                        k++;
                }
        }
}
```

1.1.6. MỘT SỐ THUẬT TOÁN TRÊN MẢNG 1 CHIỀU

Ghép mảng:

Ghép 2 mảng thành 1 mảng:

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

1.2. CHUỖI KÝ TỰ

1.2.1. Ký tự (character)

1.2.2. Chuỗi

1.2.3. Các thao tác trên chuỗi ký tự

1.2.4. Một số hàm xử lý chuỗi

1.2.1. KÝ TỰ (CHARACTER)

1.2.1.1. Khái niệm

- ✓ **Ký tự " in được " gồm:** 52 chữ cái ('a', 'b', ...'z', 'A', 'B',...'Z'); 10 chữ số ('0','1','2',....'9'); Khoảng trắng;

  các ký tự: ! " # $ % & ' ( ) * + , - . / : ; < = > ? @ [\ ] ^ _ { | } ~
- ✓ **Các ký tự " không in được ":** tab, lert (bell), newline, formfeed,...
- ✓ **Các ký tự " in được " đặc biệt:** '\\', '\'', '\"'
- ✓ **Các ký tự "không in được" đặc biệt:**

\n	new line;	\a	bell;
\0	null character;	\b	backspace;
\t	horizontal tab		

1.2.1. KÝ TỰ (CHARACTER)

1.2.1.2. NHẬP KÝ TỰ

✓ Hàm scanf : Sử dụng thư viện <stdio.h>

Ví dụ : char ch; scanf("%c", &ch);

✓ Hàm getch : Nhận một ký tự từ bộ đệm bàn phím và không cho hiện ký tự này lên màn hình.

Cú pháp : int getch (void)

Hàm trả về ký tự nhận được. ch = getch ();

+ Nếu ký tự có sẵn trong bộ đệm bàn phím thì hàm nhận một ký tự trong đó.

+ Nếu bộ đệm rỗng thì máy tạm dừng.

1.2.1. KÝ TỰ (CHARACTER)

1.2.1.2. NHẬP KÝ TỰ

✓ Hàm getche : Nhận một ký tự từ bộ đệm bàn phím và cho hiển thị ký tự này lên màn hình.

Cú pháp : int getche(void)

Hàm này có công dụng giống như hàm getch, nhưng cho hiển thị ký tự nhập vào lên màn hình.

```
ch = getchar();
```

1.2.1. KÝ TỰ (CHARACTER)

1.2.1.3. XUẤT KÝ TỰ

✓ Hàm putch : Xuất một ký tự ra màn hình.

Cú pháp : int putch (int ch)

Khai báo thư viện string.h

+ Đối số ch chứa ký tự cần hiển thị.

+ Hàm có công dụng xuất ký tự ch ra màn hình.

+ Ký tự sẽ được hiển thị theo màu xác định trong hàm textcolor.

+ Hàm trả về ký tự đã hiển thị.

1.2.1. KÝ TỰ (CHARACTER)

1.2.1.3. XUẤT KÝ TỰ

✓ Hàm printf : Xuất ký tự ch ra màn hình.

Cú pháp: printf (“ %c”, ch)

khai báo thư viện < stdio.h >

✓ Hàm putc(): Xuất ký tự ch ra màn hình.

Cú pháp: putc (ch)

Khai báo thư viện < string.h >

1.2.2. CHUỖI KÝ TỰ

1.2.2.1. KHÁI NIỆM

- ✓ Chuỗi là một dãy ký tự.
- ✓ Trong C không có kiểu chuỗi, chuỗi được thể hiện bằng mảng các ký tự, kết thúc bằng ký tự '\0' (NULL).
- ✓ Các hằng chuỗi được đặt trong cặp dấu nháy kép " ".

***Chú ý**: Các thao tác trên mảng có thể áp dụng đối với chuỗi ký tự.*

1.2.2. CHUỖI KÝ TỰ

1.2.2.3. Lỗi khi tạo một chuỗi

- ✓ Không sử dụng toán tử gán = để chép nội dung.

```
Char    a[4]="hi";
char    b[4];
b = a;  //??? Máy báo lỗi
```

- ✓ Không dùng toán tử == để so sánh nội dung

```
char a[]="hi";
char b[] = "there";
if(a==b) //??? Máy báo lỗi
        { ... }
```

- ✓ *Phép* gán trong kiểu dữ liệu chuỗi như thế này là sai

```
Char   ten[10];
ten = "hoahong"
```

1.2.3. CÁC THAO TÁC TRÊN CHUỖI KÝ TỰ

1.2.3.1. NHẬP XUẤT CHUỖI DÙNG THƯ VIỆN < STDIO.H>

- ✓ **Nhập: scanf**

scanf (“%s” , & Hoten);

Đối với hàm scanf khi gặp phím space, tab, new line, Enter thì dừng, ➔ dùng hàm scanf để nhập chuỗi không có khoảng trắng.

- ✓ **Xuất: printf**

printf (“%s” , Hoten); //không xuống dòn

printf (“%s \n ” , Hoten); //Xuất xong xuống dòng

1.2.3. CÁC THAO TÁC TRÊN CHUỖI KÝ TỰ

1.2.3.2. NHẬP XUẤT CHUỖI DÙNG THƯ VIỆN <STRING.H>

✓ Hàm nhập : gets

gets (Hoten);

+ Tiếp nhận được space, tab, new line.

+ Nhận Enter thì dừng, phải khai báo hàm xóa bộ đệm bàn phím trước khi dùng hàm gets :

fflush (stdin) hay flushall ( ).

✓ Hàm xuất puts:

puts (Hoten);

+ Xuất chuỗi xong tự động xuống dòng

1.2.4. MỘT SỐ HÀM XỬ LÝ CHUỖI (string.h)

✓ Hàm kbhit: Kiểm tra bộ đệm bàn phím.

Cú pháp : int kbhit (void)

Hàm trả về giá trị khác không nếu bộ đệm bàn phím khác rỗng, trả về giá trị không nếu ngược lại.

✓ Hàm clrscr (): Dùng để xoá màn hình.

Cú pháp: void clrscr (void)

✓ Hàm gotoxy(): Dùng di chuyển con trỏ đến vị trí (x,y).

Cú pháp: void gotoxy(int x,int y)

x là số hiệu cột (1 ➔80), y là số hiệu dòng (1 ➔ 25).

1.2.4. MỘT SỐ HÀM XỬ LÝ CHUỖI (string.h)

✓ Nhập xuất chuỗi với hàm scanf và printf

```
#include <stdio.h>
#include <string.h>
void main (void)
{
    char name[20];
    printf (“Enter a name: “);
    scanf (“%s”, name); // there is no & before name
    printf (“Hello %s\n”, name);
 }
```

1.2.4. MỘT SỐ HÀM XỬ LÝ CHUỖI (string.h)

✓ Hàm strcat: Dùng để nối hai chuỗi lại với nhau.

*Cú pháp : char * strcat(char* s1, char* s2)*

Công dụng ghép chuỗi s2 vào s1

```
#include "stdio.h"
#include "string.h"
void main()
{
        char  s1[50],s2[50];
        printf("\nNhap chuoi 1 : ");      gets(s1);
        printf("Nhap chuoi 2 : ");    gets(s2);
        strcat(s1,s2);
        printf("Xuat chuoi 1 : %s",s1);
        printf("\nXuat chuoi 2 : %s",s2);
}
```

1.2.4. MỘT SỐ HÀM XỬ LÝ CHUỖI (string.h)

✓ Hàm strchr: Lần xuất hiện đầu tiên của ký tự trong chuỗi.

Cú pháp: char strchr (char* ch, int kt)*

+ Tìm lần xuất hiện đầu tiên của ký tự kt trong chuỗi ch.

+ Trả về địa chỉ của ký tự được tìm thấy ngược lại trả về NULL.

```
#include "stdio.h"
#include "string.h"
void main()
 {
        char  s[50], ch, *p;
        printf("\nNhap chuoi : ");          gets(s);
        printf("Nhap ky tu : ");            ch=getche();
        p=strchr(s,ch);
        if (p != NULL)
                printf("\nchi so cua ky tu : %d",(int)(p-s));
        else
                printf("\nKhong tim thay!");
}
```

1.2.4. MỘT SỐ HÀM XỬ LÝ CHUỖI (string.h)

✓ Hàm strcmp: so sánh hai chuỗi.

Cú pháp: int strcmp (char s1, char* s2)*

So sánh hai chuỗi s1 và s2.

Nếu trả về giá trị <0 thì chuỗi s1 nhỏ hơn chuỗi s2.

Nếu trả về giá trị 0 nếu chuỗi s1 bằng chuỗi s2.

Nếu trả về giá trị >0 thì chuỗi s1 lớn hơn chuỗi s2.

1.2.4. MỘT SỐ HÀM XỬ LÝ CHUỖI (string.h)

✓ Hàm stricmp() : So sánh chuỗi

*Cú pháp: int stricmp (const char *s1, const char *s2)*

So sánh trong n ký tự đầu tiên của 2 chuỗi s1 và s2, kho6gn phân biệt chữ thường và chữ hoa.

Nếu trả về giá trị <0 thì chuỗi s1 nhỏ hơn chuỗi s2.

Nếu trả về giá trị 0 nếu chuỗi s1 bằng chuỗi s2.

Nếu trả về giá trị >0 thì chuỗi s1 lớn hơn chuỗi s2.

1.2.4. MỘT SỐ HÀM XỬ LÝ CHUỖI (string.h)

✓ Hàm strcpy : Sao chép chuỗi.

Cú pháp: char *strcpy (char *Des, const char *Source)

➔Sao chép toàn bộ nội dung chuỗi nguồn vào chuỗi đích.

```
#include "stdio.h"
#include "string.h"
void main()
{
        char  s[50];
        strcpy(s,"Truong Dai hoc Cong Nghe");
        printf("\nXuat chuoi : %s",s);
}
```

1.2.4. MỘT SỐ HÀM XỬ LÝ CHUỖI (string.h)

✓ Hàm strncpy () : Sao chép một phần chuỗi

Cú pháp:

*char *strncpy(char *Des, const char *Source, size_t n)*

➔ Chép n ký tự đầu tiên của chuỗi nguồn sang chuỗi đích.

1.2.4. MỘT SỐ HÀM XỬ LÝ CHUỖI (string.h)

✓ Hàm strchr() : Trích một phần chuỗi

*Cú pháp : char *strchr (const char *str, int c)*

➔ Trích một chuỗi con của một chuỗi từ ký tự chỉ định đến hết chuỗi.

Ghi chú: Nếu ký tự chỉ định không có trong chuỗi, kết quả trả về là NULL.

1.2.4. MỘT SỐ HÀM XỬ LÝ CHUỖI (string.h)

✓ Hàm strstr() : Tìm kiếm nội dung chuỗi

Cú pháp:

```
char    *strstr(const char *s1, const char *s2)
```

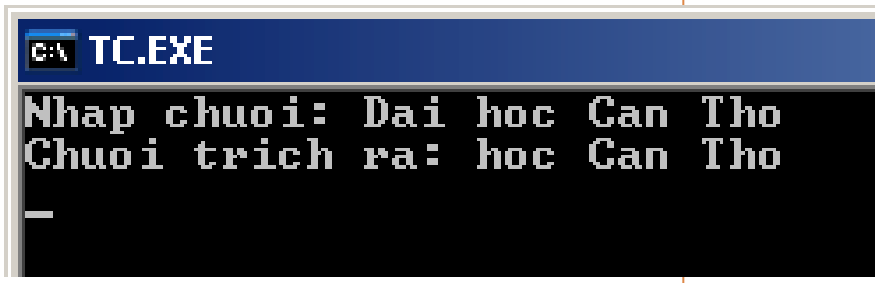
➔ Tìm kiếm chuỗi s2 trong chuỗi s1.

➔ Trả về con trỏ chỉ đến phần tử đầu tiên của chuỗi s1 có chứa chuỗi s2 hoặc giá trị NULL nếu chuỗi s2 không có trong chuỗi s1.

1.2.4. MỘT SỐ HÀM XỬ LÝ CHUỖI (string.h)

Ví dụ: Viết chương trình sử dụng hàm strstr() để lấy ra một phần của chuỗi gốc bắt đầu từ chuỗi "*hoc*".

```
#include<conio.h>
#include<stdio.h>
#include<string.h>
void main()
{
        char Chuoi[255] , *s ;
        printf("Nhap chuoi: ");gets(Chuoi);
        s=strstr(Chuoi,"hoc");
        printf("Chuoi trich ra: ");puts(s);
        getch();
}
```



1.2.4. MỘT SỐ HÀM XỬ LÝ CHUỖI (string.h)

✓ Hàm strlen() : Lấy chiều dài chuỗi với

*Cú pháp : int strlen (const char *s);*

```
#include<conio.h>
#include<stdio.h>
#include<string.h>
void main()
{
        char Chuoi [255];
        int  Dodai;
        printf("Nhap chuoi: ");gets(Chuoi);
        Dodai = strlen(Chuoi)
        printf("Chuoi vua nhap: ");puts(Chuoi);
        printf(“Co do dai %d”,Dodai);
        getch();
}
```

1.2.4. MỘT SỐ HÀM XỬ LÝ CHUỖI (string.h)

Ví dụ: Gán một chuỗi vào chuỗi khác. Gán từng ký tự.

```
#include<conio.h>
#include<stdio.h>
#include<string.h>
void  main()
{
        clrscr() ;
        char newstr [35];
        char str[] = " Trường ĐH Công Nghệ Tp.HCM";
        for(int i = 0; i<strlen(str); i++)
                newstr[i] = str[i];
        newstr[i] = '\0';
        getch () ;
}
```

1.2.4. MỘT SỐ HÀM XỬ LÝ CHUỖI (string.h)

✓ Hàm toupper(): Chuyển 1 ký tự thường thành ký tự hoa

Cú pháp: *char toupper (char c)*

➔Chuyển đổi một ký tự thường thành ký tự hoa.

➔ Sử dụng thư viện ctype.h

1.2.4. MỘT SỐ HÀM XỬ LÝ CHUỖI (string.h)

✓ Hàm strupr(): Chuyển chuỗi thường thành chuỗi hoa

Cú pháp: char *strupr(char *s)

```
#include<conio.h>
#include<stdio.h>
#include<string.h>
void main()
{
        char Chuoi[255],*s;
        printf("Nhap chuoi thường: "); gets(Chuoi);
        s=strupr(Chuoi) ;
        printf("Chuoi chu hoa: "); puts(s);
        getch();
}
```

1.2.4. MỘT SỐ HÀM XỬ LÝ CHUỖI (string.h)

✓ Hàm strlwr(): Chuyển chuỗi hoa thành chuỗi thường

Cú pháp: char *strlwr(char *s)

```
#include<conio.h>
#include<stdio.h>
#include<string.h>
void main()
{
        char Chuoi[255],*s;
        printf("Nhap chuoi hoa: "); gets(Chuoi);
        s=strlwr(Chuoi) ;
        printf("Chuoi chu thường: "); puts(s);
        getch();
}
```

1.2.4. MỘT SỐ HÀM XỬ LÝ CHUỖI (string.h)

✓ Đổi từ chuỗi ra số, hàm atoi(), atof(), atol()

Cú pháp :

int atoi(const char *s): chuyển thành số nguyên

long atol(const char *s): chuyển thành số nguyên dài

float atof(const char *s) : chuyển thành số thực

Nếu chuyển không thành công, kết quả trả về là 0.

Sử dụng thư viện stdlib.h

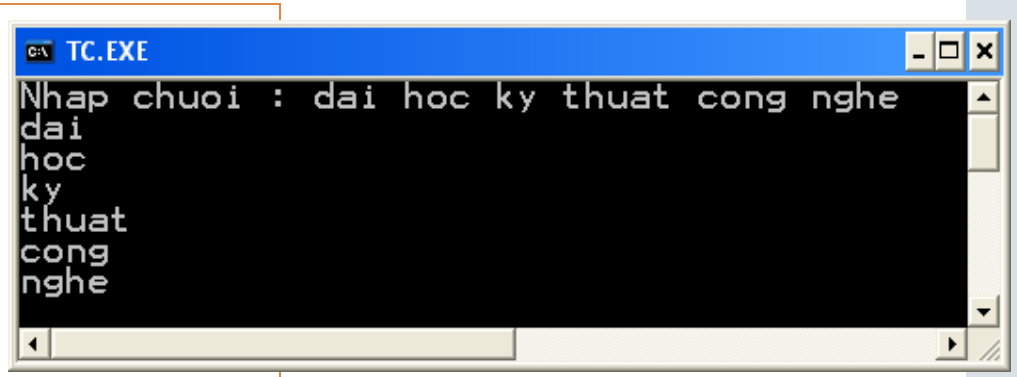
1.2.4. MỘT SỐ HÀM XỬ LÝ CHUỖI (string.h)

✓ Tách chuỗi thành các chuỗi con:

char* strtok (char *s1 , const char *s2)

Tách s1 thành từng chuỗi con, ngăn cách nhau bởi s2.

```
#include <string.h>
void main()
{
        char s[80] , *p ;
        gets(s);
        p = strtok(s," ");
        if (p) printf("%s" , p);
        while(p)
        {
                p = strtok(NULL," ");
                if (p) printf("%s" , p);
        }
}
```



1.2.4. MỘT SỐ HÀM XỬ LÝ CHUỖI (string.h)

✓ Đảo ngược chuỗi :

Cú pháp: char* strrev(char *s)

```
#include <string.h>
void main()
{
        char *s , *p ;
        gets(s);
        p = strrev(s);
        puts(p);
}
```

1.3. BÀI TẬP

1. **Viết chương trình thao tác trên mảng 1 chiều, n số nguyên**
   - Nhập mảng
   - Xuất mảng đã nhập
   - Xuất phần tử lớn nhất
   - Xuất phần tử nhỏ nhất
   - Xuất những phần tử là số nguyên tố
   - Tính và xuất tổng giá trị các phần tử
   - Tính và xuất tổng bình phương các phần tử âm.
   - Đếm và xuất các phần tử chẵn

1.3. BÀI TẬP

2. Viết chương trình thao tác trên mảng 1 chiều ngẫu nhiên có n phần tử số nguyên:

- Phát sinh mảng ngẫu nhiên n phần tử (0-100)
- Xuất mảng đã phát sinh
- Đảo các phần tử trong mảng (3 4 2 → 2 4 3)
- Sắp xếp mảng theo thứ tự tăng dần
- Xuất mảng đã sắp xếp
- Loại bỏ các phần tử trùng nhau
- Xuất mảng sắp xếp tăng đã loại bỏ các phần tử trùng nhau

1.3. BÀI TẬP

3. Viết chương trình thao tác trên chuỗi:

- Nhập chuỗi
- Xuất chuỗi
- Nhập vào 2 chuỗi, xuất chuỗi theo thứ tự từ điển
- Đếm số ký tự ‘ a’ có trong chuỗi
- Cắt khoảng trắng có trong chuỗi
- Đếm khoảng trắng trong chuỗi .
- Đếm số từ có trong chuỗi
- Sắp xếp chuỗi tăng dần
- Nhập 3 chuỗi, xuất chuỗi theo thứ tự từ điển.

HUTECH
Đại học Công nghệ Tp.HCM

KHOA CÔNG NGHỆ THÔNG TIN

Bài giảng:

KỸ THUẬT LẬP TRÌNH

HẾT BÀI 1
MẢNG MỘT CHIỀU

C Ngôn ngữ lập trình số 1 thế giới