

NỘI DUNG CHÍNH BUỔI 9

HÀM (tt)

(6) Xây dựng thư viện hàm

- Xây dựng thư viện chứa các hàm tự định nghĩa để tái sử dụng khi cần
 - ❖ **Bước 1:** Tạo file header **ten_thu_vien_ham.h** để khai báo các nguyên hàm:

```
#ifndef ten_thu_vien
#define ten_thu_vien
Nguyên hàm 1;
Nguyên hàm 2;
...
Nguyên hàm n;
#endif
```

VD: Tạo file header **kiem_tra_so.h** như sau:

```
#ifndef ktso
#define ktso
int KiemTraSNT(int n);
int KiemTraSCP(int n);
int KiemTraSHT(int n);
int KiemTraSDX(int n);
#endif
```

- ❖ **Bước 2:** Tạo file source **ten_thu_vien_ham.cpp** để định nghĩa các hàm tương ứng với các nguyên hàm đã khai báo:

```
#include <...>
Hàm 1;
Hàm 2;
...
Hàm n;
```

VD: Tạo file source **kiem_tra_so.cpp** như sau:

```
#include<cmath>

int KiemTraSNT(int n)
{
    int snt=1;
    if(n<2)
        snt=0;
    else
        for(int i=2;i<n;i++)
            if(n%i==0)
            {
                snt=0;
                break;
            }
    return snt;
}

int KiemTraSCP(int n)
{
    int scp=0;
    if(n<1)
        scp=0;
    else
        for(int i=1;i<=n/2;i++)
            if(i*i==n)
            {
                scp=1;
                break;
            }
    return scp;
}
```

```
int KiemTraSHT(int n)
{
    int sht;
    if(n<6)
        sht=0;
    else
    {
        int tongus=0;
        for(int i=1;i<=n/2;i++)
            if(n%i==0)
                tongus+=i;
        sht=(tongus==n)?1:0;
    }
    return sht;
}

int KiemTraSDX(int n)
{
    int sdx;
    if(n<1)
        sdx=0;
    else
    {
        int m=n,dao=0,dv;
        while(m>0)
        {
            dv=m%10;
            dao=dao*10+dv;
            m/=10;
        }
        sdx=(dao==n)?1:0;
    }
    return sdx;
}
```

❖ Bước 3: Sử dụng thư viện hàm đã xây dựng:

```
#include <ten_thu_vien_chuan>
#include "ten_thu_vien_ham.h"
```

Lưu ý:

- Sử dụng cặp dấu <...> đối với các **thư viện chuẩn có sẵn** trong C => Khi biên dịch, CT sẽ tìm thư viện trong ổ đĩa cài đặt phần mềm (C:\Microsoft Visual Studio\VC\include) để liên kết.
- Sử dụng cặp dấu "..." đối với các **thư viện tự định nghĩa** => Khi biên dịch, CT sẽ tìm thư viện trong thư mục lưu dự án hiện hành, nếu không tìm thấy CT sẽ tiếp tục tìm thư viện trong ổ đĩa cài đặt phần mềm (C:\Microsoft Visual Studio\VC\include) để liên kết.
- Trường hợp file sử dụng thư viện không cùng vị trí lưu trữ với file thư viện thì phải sử dụng đường dẫn tuyệt đối / tương đối đến file thư viện

VD:

```
#include "D:\KTLT\KiemTraSo\kiem_tra_so.h" /*đường dẫn tuyệt đối tính từ ổ đĩa gốc
truy xuất đến vị trí file thư viện*/
#include "..\KTLT\KiemTraSo\kiem_tra_so.h" /*đường dẫn tương đối tính từ vị trí file
sử dụng thư viện truy xuất đến vị trí
file thư viện*/
```

VD: Sử dụng thư viện kiểm tra số đã khai báo và định nghĩa như sau:

```
#include<stdio.h>
#include<conio.h>
#include"kiem_tra_so.h"
void main()
{
    int n;
    //Nhập n
    if(KiemTraSNT(n)==1)
        printf("%d la SNT",n);
    else
        printf("%d khong la SNT",n);
    ...
}
```

MẢNG 1 CHIỀU

(1) Khái niệm

- Mảng là một tập hợp **nhiều phần tử (biến)** có **cùng một kiểu** và **chung một tên** được **sắp xếp liên tiếp nhau trong bộ nhớ**.
- Mỗi phần tử của mảng có một đại lượng xác định vị trí tương đối của phần tử đó so với các phần tử khác trong mảng, gọi là **chỉ số**.

(2) Cú pháp khai báo

<kiểu_mảng> <tên_mảng>[<kích_thước>];

- Mỗi phần tử của mảng được truy nhập thông qua tên mảng cùng với chỉ số đặt giữa hai ngoặc vuông.
- Chỉ số là một số nguyên được đánh số từ 0 đến <kích thước> - 1.

VD:

```
int a[5]; /*Khai báo một mảng a gồm 5 phần tử kiểu int, bao gồm:  
a[0], a[1], a[2], a[3], a[4] */
```

(3) Khởi tạo

- Để khởi tạo mảng ta liệt kê danh sách các giá trị của mảng trong cặp dấu ngoặc "{" và "}".
- Kích thước mảng nếu có luôn \geq số phần tử mảng trong danh sách khởi tạo.

VD:

```
int a[5] = {20, -5, 8, 40, 10};  
int b[] = {100, -25, 18, 10, 18};
```

(4) Nhập / xuất mảng 1 chiều

VD:

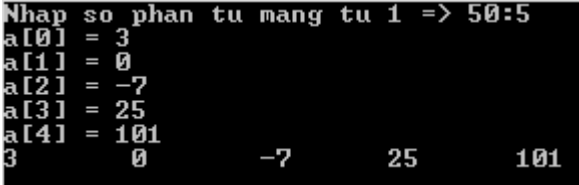
```
#include<stdio.h>
#include<conio.h>

void main()
{
    int a[50]; //Khai báo mảng chứa tối đa 50 phần tử kiểu int
    int n;     //Lưu số phần tử thực có của mảng

    //Nhập số phần tử thực có của mảng
    do
    {
        printf("Nhap so phan tu mang tu 1 => 50:");
        scanf("%d", &n);
    } while(n < 1 || n > 50);

    //Nhập giá trị cho các phần tử của mảng
    for(int i = 0; i < n; i++)
    {
        printf("a[%d] = ", i);
        scanf("%d", &a[i]);
    }

    //In giá trị của các phần tử mảng
    for(int i = 0; i < n; i++)
        printf("%d\t", a[i]);
    _getch();
}
```



```
Nhap so phan tu mang tu 1 => 50:5
a[0] = 3
a[1] = 0
a[2] = -7
a[3] = 25
a[4] = 101
3      0      -7      25      101
```

(5) Dùng mảng một chiều làm tham số truyền cho hàm

- Tên mảng là một hằng địa chỉ và nó chính là địa chỉ phần tử đầu tiên của mảng.
Như vậy, **a** \Leftrightarrow **&a[0]**
- Khi dùng tên mảng làm tham số thực truyền cho hàm thì thực chất là địa chỉ phần tử đầu tiên của mảng được truyền cho hàm => tham số hình thức tương ứng trong định nghĩa hàm phải viết dưới dạng con trỏ.
- **Lưu ý:** Tham số hình thức tương ứng với tham số thực là tên mảng cũng có thể viết như sau:

```
void NhapMang(int a[], int *n)
void XuatMang(int a[], int n)
```

VD: Xây dựng 2 hàm nhập / xuất mảng 1 chiều

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

void NhapMang(int *a, int *n)
{
    //Nhập số phần tử thực có của mảng
    do
    {
        printf("Nhap so phan tu mang tu 1 => 50:");
        scanf("%d", n); //hoặc: scanf("%d", &(*n));
    } while(*n < 1 || *n > 50);

    //Nhập giá trị cho các phần tử của mảng
    for(int i = 0; i < *n; i++)
    {
        printf("a[%d] = ", i);
        scanf("%d", &a[i]);
    }
}
```

```
void XuatMang(int *a, int n)
{
    //In giá trị của các phần tử mảng
    for(int i = 0; i < n; i++)
        printf("%d\t", a[i]);
}

void main()
{
    int a[50]; //Khai báo mảng chứa tối đa 50 phần tử kiểu int
    int n;     //Lưu số phần tử thực có của mảng
    NhapMang(a, &n); //Gọi hàm nhập mảng 1 chiều
    XuatMang(a, n);  //Gọi hàm xuất mảng 1 chiều
    _getch();
}
```

Hoặc viết cách khác như sau:

```
#include<stdio.h>
#include<conio.h>
void NhapMang(int a[], int &n)
{
    //Nhập số phần tử thực có của mảng
    do
    {
        printf("Nhap so phan tu mang tu 1 => 50:");
        scanf("%d", &n);
    } while(n < 1 || n > 50);

    //Nhập giá trị cho các phần tử của mảng
    for(int i = 0; i < n; i++)
    {
        printf("a[%d] = ", i);
        scanf("%d", &a[i]);
    }
}
```



```
void XuatMang(int a[],int n)
{
    //In giá trị của các phần tử mảng
    for(int i = 0; i < n; i++)
        printf("%d\t", a[i]);
}

void main()
{
    int a[50]; //Khai báo mảng chứa tối đa 50 phần tử kiểu int
    int n;     //Luu số phần tử thực có của mảng
    NhapMang(a,n); //Gọi hàm nhập mảng 1 chiều
    XuatMang(a,n); //Gọi hàm xuất mảng 1 chiều
    _getch();
}
```