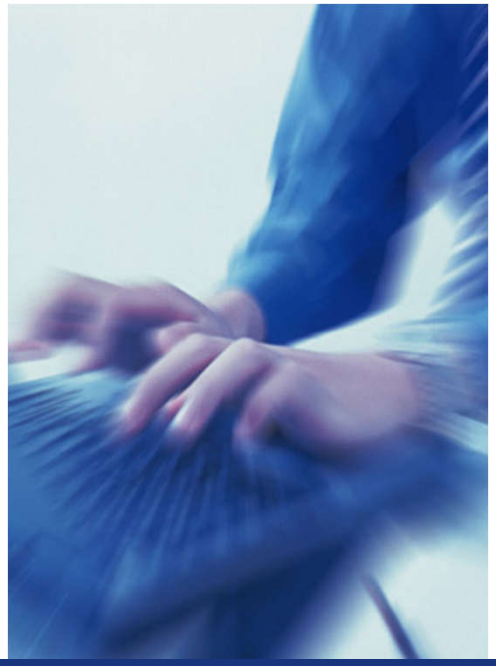
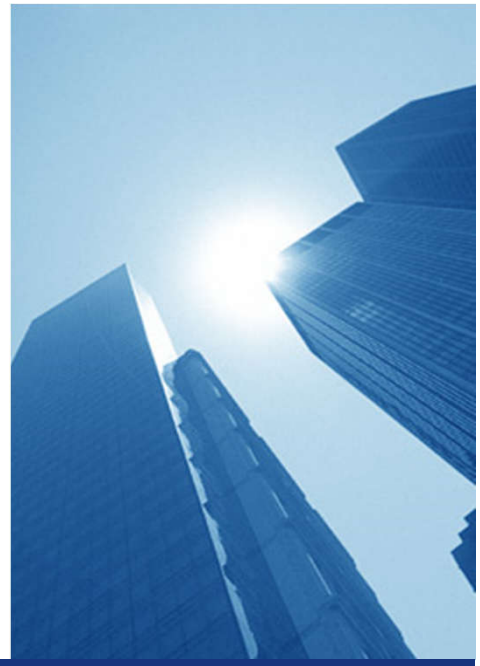




Phần 6: Struct – Cấu trúc

Xét bảng thông tin sinh viên

MSV	HVT	LOP	DIEM
20002106	Phạm Tiến Anh	K65KTDTTH	9.5

- Bảng trên có 4 cột thông tin để mô tả một sinh viên.
  - Mã sinh viên, Họ và tên, Lớp, Điểm
- Hàng đầu tiên chứa các *tiêu đề* đại điện cho các cột là:
  - MSV, HVT, LOP, DIEM
  - Các tiêu đề đó còn được gọi là các *trường* thông tin
- Hàng phía dưới chứa *giá trị* của các *trường*.

Bảng trên là một ví dụ về *dữ liệu có cấu trúc*. Một đối tượng trong bảng có nhiều trường thông tin.

Kiểu dữ liệu của các trường thông tin

MSV	HVT	LOP	DIEM
20002106	Phạm Tiến Anh	K65KTDTTH	9
20002147	Trương Ngôn Nghĩa	K65KTDTTH	8
20002133	Nguyễn Quang Huy	K65KTDTTH	8.5
20002112	Đào Ánh Dương	K65KTDTTH	7.5

- *int* MSV
- *char* HVT
- *char* LOP
- *float* DIEM

Mô tả dữ liệu có cấu trúc ở bảng trên như sau

Ngôn ngữ C hỗ trợ một số cú pháp để mô tả dữ liệu cấu trúc trên

Cú pháp 1:

```
struct      sinhvien
{
        int     MSV;
        char    HVT[50];
        char    LOP[15];
        float   DIEM;
};
```

Cú pháp 2:

```
typedef     struct
{
        int     MSV;
        char    HVT[50];
        char    LOP[15];
        float   DIEM
} sinhvien ;
```


Nội dung chính

1 **Struct – Cấu trúc**

1. Cấu trúc

- Là sự mở rộng của mảng, cho phép ta lưu trữ và xử lý thông tin phức tạp hơn.
- Là một tập hợp các biến, mảng,… với các kiểu dữ liệu khác nhau, mỗi thành phần này được gọi là *trường*. Chúng được nhóm lại và biểu thị bởi một tên duy nhất (tên của cấu trúc).

- Là kiểu dữ liệu phức hợp do người dùng tự định nghĩa khi mà một kiểu dữ liệu sơ cấp của C không thể mô tả được.

Ví dụ: nhân_viên: họ_tên(char), năm_sinh(int), lương(float)

✓ Các thành phần của một cấu trúc có thể là một cấu trúc khác.

2. Cú pháp định nghĩa dữ liệu kiểu struct

- Cú pháp 1:

```
struct <tên_kiểu_cấu_trúc>
{
<tên_kiểu_dữ_liêu>   tên_trường_1;
<tên_kiểu_dữ_liệu>   tên_trường_2;
…
<tên_kiểu_dữ_liệu>   tên_trường_n;
} ;
----------Ví dụ--------------
struct   student
{
char        name[100];
int         age;
float       score;
};
```

- Cú pháp 2:

```
typedef    struct
{
<tên_kiểu_dữ_liệu>   tên_trường_1;
<tên_kiểu_dữ_liệu>   tên_trường_2;
…
<tên_kiểu_dữ_liệu>   tên_trường_n;
} <tên_kiểu_cấu_trúc> ;
----------Ví dụ--------------
typedef    struct
{
char        name[100];
int         age;
float       score;
} student;
```

3. Khai báo biến kiểu cấu trúc

- Sau khi định nghĩa xong kiểu dữ liệu mới. Ta có thể khai báo các biến có kiểu dữ liệu như đã được định nghĩa theo các cách sau:

  - ✓ Khai báo biến có kiểu dữ liệu được định nghĩa theo cú pháp 1:
    struct <tên_kiểu_cấu_trúc> <danh_sách_tên_các_biến_cấu_trúc>
    ví dụ: ***struct student a;***
  - ✓ Khai báo biến có kiểu dữ liệu được định nghĩa theo cú pháp 2:
    <tên_kiểu_cấu_trúc> <danh_sách_tên_các_biến_cấu_trúc>
    ví dụ: ***student a;***

- Cách sau đây vừa định nghĩa cấu trúc, vừa khai báo các biến có kiểu cấu trúc đang được định nghĩa:

```
struct      <tên_kiểu_cấu_trúc>
            {
            <tên_kiểu_dữ_liệu>  tên_trường_1;
            …
            <tên_kiểu_dữ_liệu>  tên_trường_n;
            }  <danh_sách_tên_các_biến_cấu_trúc> ;
```

Ví dụ:

```
float     x;
int       y;

student   z;          // khai báo biến đơn
student   a[10];      // khai báo biến mảng
student   *p;         // khai báo con trỏ
```

4. Truy cập các thành phần cấu trúc

- Thành phần cơ bản (trường) của một cấu trúc có thể là *biến đơn*, *mảng* hoặc có thể là *một cấu trúc* khác. Do đó khi xử lý một biến kiểu cấu trúc bao giờ cũng phải được thực hiện thông qua các trường của nó. Để truy cập đến các trường của một biến cấu trúc ta dùng một trong các cách viết sau.

✓ Truy cập các trường của *biến đơn* cấu trúc như sau:

Ví dụ: *khai báo* tên_cấu_trúc biến

truy cập biến . tên_trường_i

✓ Truy cập các trường của *biến mảng* cấu trúc như sau:

Ví dụ: *khai báo* tên_cấu_trúc biến[100]

truy cập biến[chỉ_số].tên_trường_i

✓ Nếu biến cấu trúc là *con trỏ*, ta truy cập các trường của con trỏ cấu trúc thông qua phép toán: →

Ví dụ: *khai báo* tên_cấu_trúc *biến

truy cập biến → tên_trường_i

Ví dụ: Định nghĩa một cấu trúc toado2d

Định nghĩa một cấu trúc *toado2d* với 2 trường thông tin là *x* và *y*

```
typedef         struct
{
        float   x;
        float   y;
}       toado2d;
```

```
// khai bao bien co kieu toado2d
toado2d         a, b;
```

Tóm tắt cách truy cập các thành phần biến cấu trúc

```
// định nghĩa kiểu cấu trúc student
typedef   struct
{
        char   name[20];
        int      age;
        float   score;
} student ;
```

```
// khai báo một số biến kiểu student

student   x , a[50], *p;
```

//Bảng dưới đây mô tả tóm tắt cách truy xuất các thành phần của các biến có kiểu cấu trúc *student.*

Biến cấu trúc	Lấy địa chỉ		Lấy giá trị	
Biến đơn	&biến . tên_trường		biến . tên_trường	
Ví dụ: **x**	*&x .score*		*x .score*	
Biến mảng	&biến[chỉ_số] . tên_trường		biến[chỉ_số] . tên_trường	
	&(biến + chỉ số) →tên_trường		(biến + chỉ số) →tên_trường	
Ví dụ: **a**	*&a[i].score*	*&(a+i) →score*	*a[i].score*	*(a+i) →score*
Biến con trỏ	&biến → tên_trường		biến → tên_trường	
Ví dụ: **p**	*&p→score*		*p→score*	

5. Xuất, nhập dữ liệu với biến cấu trúc

Ví dụ: Nhập và in thông tin cho 1 biến kiểu *student*

- Định nghĩa một cấu trúc:

```
typedef  struct  {
                 int  age;  char  name[20];  float score;
                 } student;
```

- Khai báo biến có kiểu dữ liệu student

```
student  q;
```

- Nhập dữ liệu cho biến q

```
scanf("%d%s%f",&q.age, &q.name, &q.score);
```

- Xuất dữ liệu của biến q:

```
printf("%d%s%f",q.age, q.name, q.score);
```

Ví dụ: Cách dùng biến đơn cấu trúc

```
// Chương trình này nhập vào thông tin của một biến đơn kiểu cấu trúc

#include <stdio.h>
typedef   struct
    {
    int  age;  char  name[20];  float score;
    } student;

main()
 {
 student   q;
 printf("Nhap vao thong tin: \n");
 scanf("%d%s%f", &q.age, &q.name, &q.score);

 printf("\nAGE = %d  NAME = %s  SCORE = %f ", q.age, q.name, q.score);
 }
```

Định nghĩa một kiểu dữ liệu mới có tên là **student**

Khai báo biến **q** có kiểu **student**

Ví dụ: Cách dùng con trỏ cấu trúc

```
#include<stdio.h>
#include<stdlib.h>
typedef   struct
{
                int   ns;              //nam sinh
                char  nm[100];         //ho ten
                float  dtb;            //diem trung binh
} student ;
main()
{
                student     *a;
                int   i, N;
                printf("Nhap tong so sinh vien: ");
                scanf("%d%*c",&N);
                puts("Nhap thong tin cua sinh vien");
                a = (student*)malloc(N*sizeof(student));
                for(i=0; i< N; i++)
                {
                                printf("Ho ten:\t");
                                gets(a[i].nm);
                                printf("Nam sinh:\t");
                                scanf("%d",&((a+i)->ns));
                                printf("Diem:\t");
                                scanf("%f%*c",&(a+i)->dtb);
                }
                printf("-------------------------\n");
                printf("Thong tin vua nhap\n");
                printf("Ho va ten\tDiem\tNam sinh\n");
                for(i=0; i< N; i++)
                    printf("%s\t%.2f\t%d\n",(a+i)->nm,(a+i)->dtb,(a+i)->ns);
                free(a);
}
```

- Chương trình bên nhập và in thông tin của N sinh viên.

- Kết quả chạy chương trình.

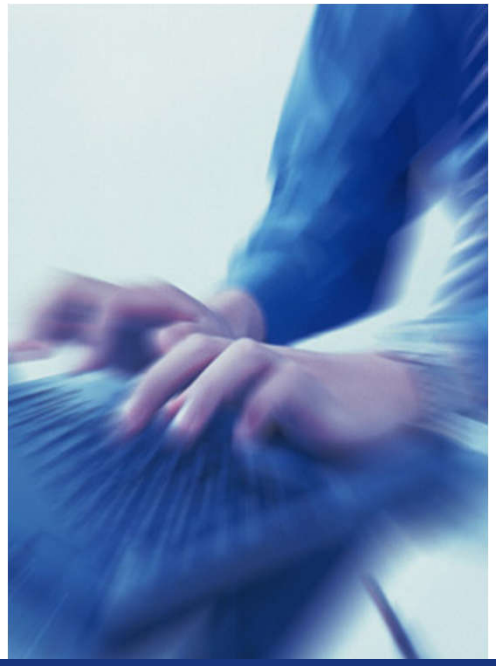
```
Nhap tong so sinh vien: 3
Nhap thong tin cua sinh vien
Ho ten: Nguyen Hung
Nam sinh:       1990
Diem:   5.6
Ho ten: Pham Quang
Nam sinh:       1992
Diem:   6.4
Ho ten: Hoang Minh
Nam sinh:       1994
Diem:   7.5
-------------------------
Thong tin vua nhap
Ho va ten       Diem    Nam sinh
Nguyen Hung     5.60    1990
Pham Quang      6.40    1992
Hoang Minh      7.50    1994
```

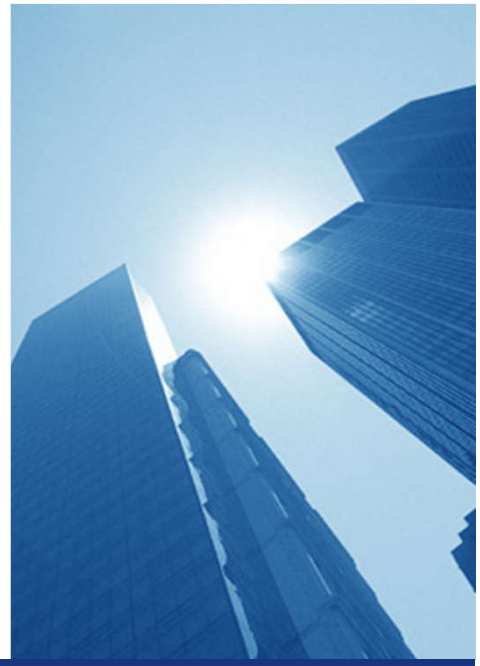

Nội dung chính

2 **Bài tập**


Bài tập

1. Nhập vào danh sách N sinh viên gồm các thông tin sau: mã sinh viên, họ tên, ngày tháng năm sinh, điểm trung bình, quê quán, giới tính. In ra sinh viên có điểm lớn nhất.
2. Sắp xếp danh sách sinh viên trên theo điểm và in ra màn hình.





Thank You !