CHƯƠNG 5: KIỂU DỮ LIỆU CẤU TRÚC

5.1. Khai báo cấu trúc

5.1.1. Khai báo kiểu dữ liệu cấu trúc

```
struct <tên kiểu cấu trúc>
      {
         <kiểu dữ liệu 1>  <tên trường 1> ;
         <kiểu dữ liệu 2>  <tên trường 2> ;
            . . . .
         <kiểu dữ liệu n>  <tên trường n> ;
      };
```

Ví dụ:

```
struct  hocsinh
      {
           char  hoten[25] ;
           float  diemthi ;
      } ;
```

CHƯƠNG 5: KIỂU DỮ LIỆU CẤU TRÚC

```
struct  ngay
      {
           int  ngaythu ;
           char  thang [10] ;
           int  nam ;
      } ;

struct congnhan
      {
           char  hoten [20] ;
           char  diachi [30] ;
           float  bacluong ;
           struct  ngay  ngaysinh ;
      } ;
```

CHƯƠNG 5: KIỂU DỮ LIỆU CẤU TRÚC

5.1.2. Khai báo biến cấu trúc

Khai báo biến cấu trúc:

struct < tên cấu trúc > < danh sách biến > ;

Ví dụ:

```
struct  hocsinh  hsinh [ 100 ] , hs ;
struct  congnhan  nhancong[1000] , nguoi1, nguoi2 ;
```

Khai báo con trỏ cấu trúc:

struct < tên cấu trúc > *< danh sách biến > ;

Ví dụ:

```
struct  hocsinh  * hs1,  *hs2 ;
struct congnhan  * cn1 ;
```

CHƯƠNG 5: KIỂU DỮ LIỆU CẤU TRÚC

Khai báo đồng thời kiểu cấu trúc và biến cấu trúc:

struct < tên cấu trúc >
{ < khai báo các thành phần >
} < danh sách biến cấu trúc > ;

Ví dụ:

```
struct  ngay
      {   int  ngaythu ;
          char  thang [10] ;
          int  nam ;
      }   ngaydi , ngayden ;
struct congnhan
      {  char  hoten [20] ;
          char  diachi [30] ;
          float  bacluong ;
          struct  ngay  ngaysinh ;
      }  nguoi1, nguoi2 ;
```

CHƯƠNG 5: KIỂU DỮ LIỆU CẤU TRÚC

5.2. Sử dụng typedef với cấu trúc

```
typedef   struct
      {
         <kiểu dữ liệu 1>  <tên trường 1> ;
         <kiểu dữ liệu 2>  <tên trường 2> ;
            . . . .
         <kiểu dữ liệu n>  <tên trường n> ;
      }        <tên kiểu cấu trúc> ;
```

CHƯƠNG 5: KIỂU DỮ LIỆU CẤU TRÚC

Ví dụ:

```
typedef  struct
      {
           int  ngaythu ;
           char  thang [10] ;
           int  nam ;
      }  ngay ;
typedef  struct
{
           char  hoten [20] ;
           char  diachi [30] ;
           float  bacluong ;
           ngay  ngaysinh ;
      }  congnhan ;
```

CHƯƠNG 5: KIỂU DỮ LIỆU CẤU TRÚC

5.3. Thao tác trên biến cấu trúc

- Truy nhập tới thành phần trong cấu trúc:

<tên biến cấu trúc > . <tên thành phần>

Ví dụ:

```
typedef  struct          {   float  x , y ;
                         }  diem  ;
typedef  struct          { int  n ;
                         diem  dinh [20] ;
                         }  dagiac ;
diem  p ;
dagiac  dg1 ;
p.x = 5 ; p.y = 10 ;
dg1. n = 10 ;
dg1.dinh [1].x = 100 ; dg1.dinh[1].y = 200;
```

CHƯƠNG 5: KIỂU DỮ LIỆU CẤU TRÚC

- Truy nhập tới thành phần cấu trúc từ con trỏ cấu trúc:

<tên biến con trỏ cấu trúc> -> <tên thành phần>

Ví dụ:

```
diem *ptr_p ;
dagiac  *pdg ;
ptr_p->x = 15 ;
ptr_p->y = 20 ;
pdg->n = 5
pdg->dinh[1].x = 50 ;
pdg->dinh[1].y = 100 ;
```

CHƯƠNG 5: KIỂU DỮ LIỆU CẤU TRÚC

- Nhập dữ liệu cho biến cấu trúc:

Để nhập dữ liệu cho biến cấu trúc, ta phải tự thực hiện nhập dữ liệu cho các thành phần trong biến cấu trúc. Muốn nhập một danh sách (một dãy cấu trúc) thì phải dùng lệnh lặp để nhập từng cấu trúc.

- Đưa cấu trúc ra màn hình: Phải đưa dữ liệu của từng trường ra. Muốn đưa ra một danh sách thì phải dùng lệnh lặp để đưa ra từng cấu trúc.

- Xử lý mảng các cấu trúc:

* Tìm các cấu trúc thỏa mãn điều kiện nào đó.

* Chèn thêm, xóa bớt một vài cấu trúc nào đó.

* Sắp xếp lại các cấu trúc theo một tiêu chuẩn nào đó.

* Tính toán trong danh sách các cấu trúc (tính tổng, tính trung bình cộng...).

- Phép gán giữa các biến cấu trúc: Có thể thực hiện phép gán nội dung của một biến cấu trúc cho một biến cấu trúc khác cùng kiểu.

CHƯƠNG 5: KIỂU DỮ LIỆU CẤU TRÚC

VD:Vào số liệu và in ra danh sách sinh viên có điểm từ 5 trở lên

```
#include <stdio.h>
#include <conio.h>
#include <string.h>
typedef struct
      {  char hoten[25];
          float diemthi;
      } sinhvien ;
void main()
{    int n , k , i ;
    sinhvien sv[300] ;
    char ten[25]; float diem;
    clrscr();
```

CHƯƠNG 5: KIỂU DỮ LIỆU CẤU TRÚC

```
n=0;
   do
   { printf("Nhap vao sinh vien thu %d\n", n+1);
     printf("Ho va ten: ");
     fflush(stdin); gets(ten);
     if (strcmp(ten,"") != 0)
        { strcpy(sv[n].hoten , ten) ;
          printf("Diem thi: ");
          scanf("%f", &diem);
          sv[n++].diemthi = diem;
        }
   } while (strcmp(ten , "") != 0 ) ;
```

CHƯƠNG 5: KIỂU DỮ LIỆU CẤU TRÚC

```
if (n==0)
{
  printf("Chua nhap sinh vien nao!");
  getch();
  return;
}
else {
  k=0;
  printf("\nDanh sach sinh vien tu trung binh tro len:\n");
  for(i=0; i<n; i++)
  if (sv[i].diemthi >= 5)
  printf("%d %s %6.2f\n",++k,sv[i].hoten,sv[i].diemthi);
  if (k==0) printf("Khong co sinh vien thi dat diem >= 5");
 }
 getch();
}
```

CHƯƠNG 5: KIỂU DỮ LIỆU CẤU TRÚC

5.4. Truyền biến cấu trúc cho hàm

Việc khai báo tham số hình thức của hàm là một biến cấu trúc sẽ ảnh hưởng nhiều đến tốc độ thực hiện chương trình

Để nâng cao tốc độ thực hiện chương trình, thay vì truyền nội dung cấu trúc cho hàm người ta thường truyền địa chỉ của biến cấu trúc cho hàm.

Truyền biến cấu trúc bằng tham biến: Việc truyền địa chỉ biến cấu trúc có hai cái lợi:

- Dù kích thước biến lớn nhường nào thì địa chỉ của biến vẫn chỉ là 4 byte.

- Ta có thể sử dụng toán tử "->" để thực hiện các truy xuất dữ liệu lên biến cấu trúc.

CHƯƠNG 5: KIỂU DỮ LIỆU CẤU TRÚC

Ví dụ: Kiểm tra xem điểm M có nằm ở trong tam giác ABC không?

```
#include <stdio.h>
#include <conio.h>
#include <math.h>
typedef struct
{
  float x,y;
} diem;
void NhapDiem(diem *S)
{
   float tam;
   printf("Nhap toa do x: "); scanf("%f", &tam); S->x=tam;
   printf("Nhap toa do y: "); scanf("%f", &tam); S->y=tam;
}
```

CHƯƠNG 5: KIỂU DỮ LIỆU CẤU TRÚC

```
// Ham tinh khoang cach giua hai diem
float KhCach(diem A, diem B)
{ float tam;
   tam = sqrt(pow(A.x-B.x, 2) + pow(A.y-B.y, 2));
   return tam;
}
// Ham tinh dien tich tam giac
float Dtich(diem A, diem B, diem C)
{ float s,da,db,dc,p;
   da = KhCach(B, C);     db = KhCach(A, C);     dc =
KhCach(B, A);
   p = (da+db+dc)/2.0;
   s = sqrt(p*(p-da)*(p-db)*(p-dc));
   return s; }
```

CHƯƠNG 5: KIỂU DỮ LIỆU CẤU TRÚC

```
void main()
{ diem A,B,C,M;
   float s,s1,s2,s3;
   printf("Nhap diem A:\n"); NhapDiem(&A);
   printf("Nhap diem B:\n"); NhapDiem(&B);
   printf("Nhap diem C:\n"); NhapDiem(&C);
   printf("Nhap diem M:\n"); NhapDiem(&M);
   s = Dtich(A,B,C);     s1 = Dtich(M,B,C);     s2 =
Dtich(M,A,C);
   s3 = Dtich(M,B,A);
   if (s * 1.0000001 > s1+s2+s3)
            printf("M nam trong tam giac ABC");
   else     printf("M khong nam trong tam giac ABC");
   getch();
}
```

CHƯƠNG 5: KIỂU DỮ LIỆU CẤU TRÚC

5.5. Kiểu hợp (union)

union là gì?

Cũng như cấu trúc, *union* gồm nhiều thành phần nhưng chúng khác nhau ở chỗ: các thành phần của cấu trúc có những vùng nhớ khác nhau, còn các thành phần của *union* được cấp phát một vùng nhớ chung, độ dài của *union* bằng độ dài của thành phần lớn nhất.

Khai báo kiểu dữ liệu *union*:

Việc định nghĩa một kiểu *union*, việc khai báo biến *union*, mảng *union*, con trỏ *union*, cũng như cách truy nhập đến các thành phần *union* được thực hiện hoàn toàn tương tự như đối với cấu trúc. Một cấu trúc có thể có thành phần kiểu *union*, ngược lại các thành phần của *union* lại có thể là cấu trúc.