BÀI 7
CẤU TRÚC VÀ HỢP

MỤC TIÊU

Người học sau khi học xong bài 7 sẽ có các khai niệm cơ bản về các vấn đề sau:

- Kiểu cấu trúc
- Khai báo biến kiểu cấu trúc
- Truy cập đến các thành phần cấu trúc
- Thành phần kiểu field (nhóm bit)
- Con trỏ cấu trúc và địa chỉ cấu trúc
- Cấu trúc con trỏ và danh sách liên kết
- Kiểu hợp (union)

KIẾN THỨC CẦN CÓ

Các kiến thức cần thiết:

- Học xong bài 6
- Phân tích cấu trúc chương trình
- Thực hiện các bài toán về cấu trúc, mảng…
- Khuyến nghị học môn Tin Học Cơ Bản.

NỘI DUNG

1. Kiểu cấu trúc
2. Khai báo biến cấu trúc
3. Truy nhập đến các thành phần của cấu trúc
4. Mảng cấu trúc
5. Khởi gán cho cấu trúc
6. Phép gán cấu trúc
7. Con trỏ cấu trúc và địa chỉ cấu trúc
8. Hàm trên các cấu trúc
9. Cấp phát bộ nhớ động
10. Cấu trúc tự trỏ và danh sách liên kết

7.1. KIỂU CẤU TRÚC

- Kiểu cấu trúc là một nhóm các biến, mảng, hằng… đi kèm với nhau và được đặt tên riêng.
- Khi định nghĩa kiểu cấu trúc, cần chỉ rõ tên kiểu và các thành phần của nó:

```
struct tên_kiểu_cấu_trúc
{
Khai báo các thành phần của nó
};
```

  - struct: từ khóa khai báo
  - tên_kiểu_cấu_trúc : tuân theo quy tắc đặt tên biến.
  - Khai báo các thành phần của nó: Có thể là biến, mảng, nhóm bit, hợp…
  - ; : kết thúc khai báo cấu trúc bằng dấu ;

7.1. KIỂU CẤU TRÚC (tiếp theo)

Ví Dụ: khai báo cấu trúc ngày và cấu trúc nhân công

```
struct ngay
{
int ngay_thu;
char ten_thang[10];
int nam;
};
```

```
struct nhan_cong
{
char ten[15];
char dia_chi[20];
double bac_luong;
struct ngay ngay_sinh;
};
```

7.2. KHAI BÁO BIẾN CẤU TRÚC

- Khai bao biến cấu trúc nhằm xây dựng những cấu trúc theo các kiểu đã thiết kế ở phần trên.
- Quá trình khai báo biến cấu trúc tương đương như khai báo biến và mảng. Khi khai báo một biến cần chỉ ra tên và kiểu của nó:

```
struct tên_kiểu_cấu_trúc danh_sách_tên_cấu_trúc;
```

Ví Dụ:

```
struct ngay ngay_di, ngay_den;
```

```
struct nhan_cong nguoi_a,nguoi_b;
```

- ngay_di, ngay_den là 2 kiểu cấu trúc kiểu ngày. Điều này tương đương với 2 kiểu cấu trúc nguoi_a và nguoi_b

7.2. KHAI BÁO BIẾN CẤU TRÚC (tiếp theo)

Khi chưa khai báo kiểu cấu trúc thì chưa thể gán tên kiểu cho nó được. Tuy nhiên, ta có thể đồng thời thực hiện khai báo cấu trúc và gán cấu trúc luôn cho nó.

Ví Dụ:

```
struct ngay
{
int ngay_thu;
char ten_thang[10];
int nam;
} ngay_di, ngay_den;
```

- Thực hiện khai báo cấu trúc ngày
- Gán 2 kiểu cấu trúc là ngay_di và ngay_den

7.3. TRUY NHẬP NHANH ĐẾN CÁC THÀNH PHẦN CỦA CẤU TRÚC

Để truy cập đến 1 thành phân cơ bản của cấu trúc, ta dùng các lệnh sau:

```
tên_cấu_trúc.tên_thành_phần;
tên_cấu_trúc.tên_cấu_trúc.tên_thành_phần;
tên_cấu_trúc….. tên_cấu_trúc.tên_cấu_trúc.tên_thành_phần;
```

- tên_cấu_trúc: Các tên cấu trúc đã được định nghĩa. Nếu có nhiều kiểu cấu trúc được định nghĩa lồng nhau, thì chúng phải được gọi lần lượt từ ngoài vào trong
- tên_thành_phần: nội dung của biến, mảng, thành phần có trong cấu trúc.

7.3. TRUY NHẬP NHANH ĐẾN CÁC THÀNH PHẦN CỦA CẤU TRÚC (tiếp theo)

Ví Dụ 1: Tính tổng bậc lương của 2 người a và b trong cấu trúc đã được khai báo trước. Lưu kết quả vào biến s

```
s = nguoi_a.bacluong + nguoi_b.bacluong
```

Ví Dụ 2: Đưa tên người a ra màn hình

```
printf("%s",nguoi_a.ten);
```

Ví Dụ 3: Dùng con trỏ địa chỉ để nhập ngày vào cơ quan cho người b

```
int x;
scanf("%d",&x);
nguoi_b.ngay_vao_co_quan = x;
```

7.4. MẢNG CẤU TRÚC

- Khi sử dụng mảng cấu trúc cần chú ý đến kiểu giá trị đi kèm của nó. Mỗi mảng cấu trúc đều có 1 kiểu xác định, ví dụ như int, float…
- Ta có thể sư dụng một kiểu cấu trúc đã mô tả để khai báo các cấu trúc và mảng cấu trúc khác.

Ví Dụ:

```
struct nhan_cong nguoi_a,nguoi_b,to_1[10],to_2[20];
```

- Hai biến có cấu trúc là nguoi_a và nguoi_b;
- Hai mảng có cấu trúc là:
  - to_1 : kích thước 10 phần tử, kiểu nhan_cong
  - to_2: kích thước 20 phần tử, kiểu nhan_cong

7.5. KHỞI GÁN CHO CẤU TRÚC

Có thể khởi đầu cho cấu trúc ngoài, cấu trúc tĩnh, mảng cấu trúc ngoài và mảng cấu trúc tĩnh bằng cách viết thêm câu lệnh khi khai báo cấu trúc.

Ví Dụ:

```
struct date
{
int day;
int month;
int year;
char * monthname;
};
struct date dd = {4,8,1990,"Thang8"};
```

- dd: cấu trúc có kiểu date với dữ liệu khởi đầu là ngày 4, tháng 8, năm 1990, tên của tháng là Thang8

7.5. KHỞI GÁN CHO CẤU TRÚC (tiếp theo)

Ví Dụ: Khai báo 2 mảng cấu trúc lồng nhau

- Khai báo mảng cấu trúc year gồm 12 phần tử
- Mỗi phần tử của mảng lại là 1 cấu trúc month
- Đây là ví dụ điển hình của việc khai báo cấu trúc lồng nhau

```
struct month
{
    int number;
    char * name;
} year[12] =
{
    {1,"thang1"},
    {2,"thang2"},
    ----
    {12,"thang12"},
};
```

7.6. PHÉP GÁN CẤU TRÚC

Phép gán cấu trúc được thực hiện bằng cách gán các biến và các phần tử mảng của các cấu trúc cùng kiểu:

- Gán 2 biến cho nhau
- Gán biến cho phần tử của mảng
- Gán phần tử mảng cho biến
- Gán 2 phần tử mảng cho nhau

Ví Dụ: Sắp xếp thứ tự thí sinh theo tiêu chí điểm giảm.Dùng các mảng cấu trúc tiến hành so sánh và đổi vị trí các phần tử

```
struct thi_sinh
{
char ht[25];
float td;
} tg,ts[1000];

int i,j,n;
for (i = 1;i<=n-1;i++)
for (i = 1;i<=n-1;i++)
  if(ts[i].td < ts[j].td)
{
    tg = ts[i];
    ts[i] = ts[j];
    ts[j] = tg;
}
```

7.7. CON TRỎ CẤU TRÚC VÀ ĐỊA CHỈ CẤU TRÚC

Giả sử ta có 2 kiểu cấu trúc như sau:

```
struct ngay
{
int ngay_thu;
char ten_thang[10];
int nam;
};
```

```
struct nhan_cong
{
char ten[15];
char dia_chi [20];
double bac_luong;
--------
};
```

Khi đó con trỏ cấu trúc kiểu nhan_cong có thể được khai báo như sau:

```
struct nhan_cong *p,*p1,*p2,nc1,nc2,ds[100];
```

- p,p1,p2 là các con trỏ có kiểu cấu trúc là nhan_cong
- nc1,nc2: biến cấu trúc thông thường
- ds: mảng cấu trúc với kích thước là 100

7.7. CON TRỎ CẤU TRÚC VÀ ĐỊA CHỈ CẤU TRÚC (tiếp theo)

Để truy cập đến các thành phần thông qua con trỏ, ta sử dụng các cách sau:

```
tên_con_trỏ → tên_thành_phần;
```

```
(*tên_con_trỏ ).tên_thành_phần;
```

Ví Dụ:

Để truy cập vào trường thông tin ngày sinh của nc1 và p2 thì ta có thể thực hiện như sau:

```
nc1.ngay_sinh.nam ;
(*p2).ngay_sinh.ten_thang;
```

7.7. CON TRỎ CẤU TRÚC VÀ ĐỊA CHỈ CẤU TRÚC (tiếp theo)

Quá trình gán qua con trỏ phải được thực hiện gián tiếp thông qua địa chỉ của các con trỏ.

- **Ví Du:** Gửi địa chỉ của nc1 vào p1

```
p1 = &nc1;
```

- **Ví Dụ:** gửi địa chỉ ds[2] vào p2

```
p2 = &ds[2];
```

- Sau khi dùng con trỏ để lấy địa chỉ,ta có thể thực hiện các phép cộng tương ứng với nó.

  Ví Dụ: Lệnh p3 = p2 +3 sẽ trỏ đến địa chỉ của ds[5];

7.8. HÀM TRÊN CÁC CẤU TRÚC

Với 1 hàm số thì đối số của nó có thể là 1 trong các yếu tố sau:

- Biến cấu trúc: Khi đó tham số thực của hàm sẽ là 1 giá trị cấu trúc
- Con trỏ cấu trúc: Khi đó tham số thực là địa chỉ của biến cấu trúc
- Mảng cấu trúc hình thức hoặc con trỏ cấu trúc : Khi đó tham số thực tương ứng là tên mảng cấu trúc
- Giá trị trả về của hàm cấu trúc có thể là:
  - Giá trị cấu trúc: có kiểu là kiểu của hàm;
  - Con trỏ cấu trúc: Bao gồm địa chỉ và giá trụ dữ liệu tương ứng của con trỏ.

7.8. HÀM TRÊN CÁC CẤU TRÚC (tiếp theo)

Ví Dụ: Xét kiểu cấu trúc person gồm 3 thành phần

- ht: họ tên, kiểu mảng char
- ns : năm sinh, kiểu struct date
- bl: bậc lương, kiểu float

- Với các thông tin trên, ta có thể tác động vào đó các hàm sau
- Hàm ptim
- Hàm tim
- Hàm hv
- Hàm sapxep
- Hàm vao
- Hàm in

7.8. HÀM TRÊN CÁC CẤU TRÚC (tiếp theo)

Hàm ptim:

Cú pháp:

```
person *ptim(char *ht,person h[], int n);
```

Tác dụng: Hàm ptim là tim trong danh sách n nhân viên lưu trong mảng h người có tên trong ht. Hàm trả về con trỏ tới người tìm được hoặc trả về NULL nếu không tìm được

Mã nguồn:

```
person *ptim(char *ht,person h[], int n)
{
int i;
for(i=1;i<=n;++i)
   if (strcmp(ht,h[i].ht)==0)
      return(&h[i]);
      return (NULL);
}
```

7.8. HÀM TRÊN CÁC CẤU TRÚC (tiếp theo)

Hàm tim:

Cú pháp: person tim(char *ht,person h[], int n);

Tác dụng: Tương đương như hàm ptim, nhưng kết quả trả về là cấu trúc chứa thông tin người tìm được.

Mã nguồn:

```
person tim(char *ht,person h[], int n)
{
int i; person ps;
ps.ns.ngay = ps.ns.thang = ps.ns.nam = 0;
ps.bl = 0; ps.ht[0] = 0;
for(i=1;i<=n;++i)
   if (strcmp(ht,h[i].ht)==0)
      return(h[i]);
      return (ps);
}
```

7.8. HÀM TRÊN CÁC CẤU TRÚC (tiếp theo)

Hàm hv:

Cú pháp:

```
void hv(person *p1, person *p2);
```

Tác dụng: Hoán vị 2 cấu trúc p1 và p2

Mã nguồn:

```
void hv(person *p1, person *p2)
{
person h;
h = *p1;
*p1 = *p2;
*p2 = h;
}
```

7.8. HÀM TRÊN CÁC CẤU TRÚC (tiếp theo)

Hàm vao:

Cú pháp:

```
void vao(person *p);
```

Tác dụng: Nhập dữ liệu cho đối tượng p theo cú pháp sau

```
scanf("%f%*c", &h.bl);
```

Mã nguồn:

```
void vao(person *p)
{
person h; float bl;
printf("\n Ho ten"); gets(h.ht);
printf("\n Ngay Sinh"); gets(h.ns);
scanf("%d%d%d%*c", &h.ns.ngay, & h.ns.thang, & h.ns.nam);
printf("\n Bac Luong"); scanf("%f%*c", &bl); h.bl = bl; *p = h;
}
```

7.8. HÀM TRÊN CÁC CẤU TRÚC (tiếp theo)

Hàm sapxep:

Cú pháp:

```
void sapxep(person *p, int n);
```

Tác dụng: Sắp xếp n phần tử theo thứ tự tăng của năm sinh

Mã nguồn:

```
void sapxep(person *p, int n)
{  int i,j;
for (i =1; i<=n-1;++i)
for (j =1; j<=n;++j)
    if(p[i].ns.nam > p[j].ns.nam)
    {
            hv(&p[i], &p[j]);
    }
}
```

7.8. HÀM TRÊN CÁC CẤU TRÚC (tiếp theo)

Hàm in:

Cú pháp:

```
void in(person p);
```

Tác dụng: In ra màn hình cấu trúc p

Mã nguồn:

```
void in(person p)
{
printf("\n Ho ten %s Sinh ngay %d thang %d nam %d
bac luong %.1f",
p.ht,p.ns.ngay,p.ns.thang,p.ns.nam,p.bl);
}
```

7.9. CẤP PHÁT BỘ NHỚ ĐỘNG

- Trong thực tế khi làm việc với các kiểu dữ liệu chưa xác định về độ lớn nội dung cũng như số lượng phần tử sẽ gặp khó khăn nếu dùng cơ chế cấp bộ nhớ cố định.
- Để dễ dàng thay đổi số lượng trường thông tin cũng như tiết kiệm bộ nhớ, nên sử dụng cơ chế cấp phát bộ nhớ tự động.
- Trong quá trình cấp phát bộ nhớ tự động, các mảng cấu trúc sẽ được cung cấp bộ nhớ vừa đủ để lưu trữ các thông tin thuộc tính của mình.

7.10. CẤU TRÚC TỰ TRỎ VÀ DANH SÁCH LIÊN KẾT

Cấu trúc tự trỏ là cấu trúc có ít nhất 1 thành phần là con trỏ kiểu cấu trúc.

Có 3 cách định nghĩa cấu trúc tự trỏ như sau:

Cách 1

```
typedef struct pp
{
char  ht[25];
char qq[20];
int tuoi;
struct pp *tiep;
} person;
```

Cách 2

```
typedef struct pp person;
struct pp
{
char     ht[25];
char    qq[20];
int tuoi;
struct pp *tiep;
};
```

Cách 3

```
struct pp
{
char  ht[25];
char qq[20];
int tuoi;
struct pp *tiep;
};
typedef pp person ;
```

7.10. CẤU TRÚC TỰ TRỎ VÀ DANH SÁCH LIÊN KẾT (tiếp theo)

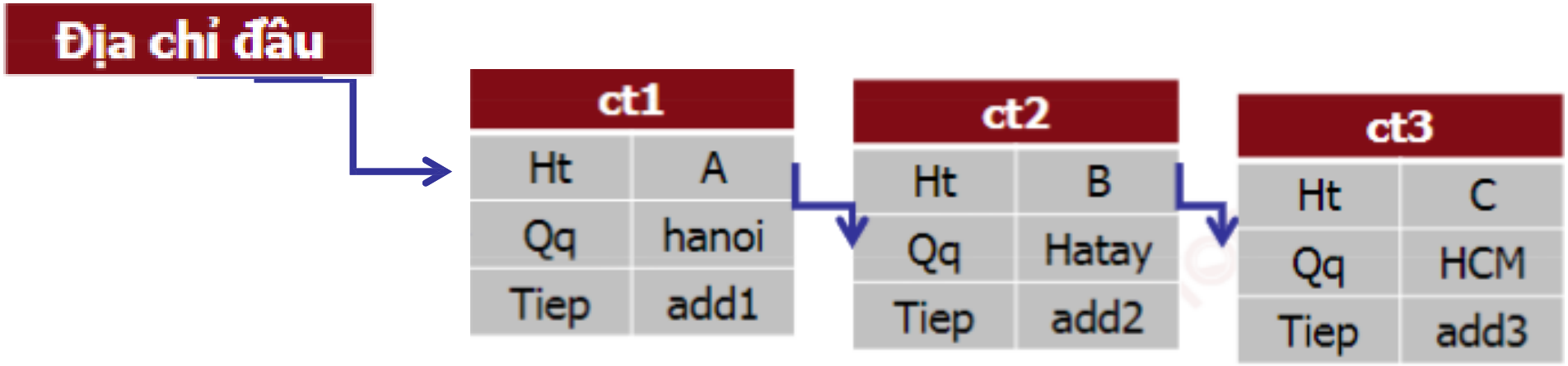
- Cấu trúc tự trỏ được sử dụng để xây dựng danh sách liên kết với các tính chất cơ bản sau:
  - Xác định rõ địa chỉ đầu của cấu trúc đang được lưu trữ
  - Trong mỗi cấu trúc, biết được các địa chỉ tiếp theo có trong danh sách
  - Cấu trúc cuối cùng chứa hằng NULL
- Khi thông tin được lưu trữ dưới dạng cấu trúc con trỏ sẽ rất dễ dàng cho việc truy xuất, thêm bớt, chỉnh sửa…
- Cấu trúc con trỏ cho phép sử dụng tương đương như một dạng cơ sở dữ liệu

7.10. CẤU TRÚC TỰ TRỎ VÀ DANH SÁCH LIÊN KẾT (tiếp theo)

Ví Dụ: Xét cấu trúc lưu trữ thông tin cá nhân gồm các trường thông tin sau:

- ht: họ tên
- qq: Quê quán
- tiep: Địa chỉ

- Bắt đầu từ địa chỉ đầu, thông tin có cấu trúc lần lượt trỏ tới các cấu trúc tiếp theo là ct1,ct2 và ct3



7.10. CẤU TRÚC TỰ TRỎ VÀ DANH SÁCH LIÊN KẾT (tiếp theo)

Ví Dụ:

Minh họa cấu trúc con trỏ thông qua chương trình minh họa các chi tiết sau:

- Nhập một số người và lưu vào bộ nhớ dưới dạng danh sách móc nối thuận.
- In danh sách móc nối ra màn hình
- Tìm kiếm trên danh sách móc nối
- Xóa một người khỏi danh sách
- Bổ sung vào cuối danh sách
- Chèn một người vào giữa danh sách

7.10. CẤU TRÚC TỰ TRỎ VÀ DANH SÁCH LIÊN KẾT (tiếp theo)

Các thủ thuật làm việc trên danh sách móc nối bao gồm:

- Tạo danh sách mới:
  - Cấp phát bộ nhớ cho 1 cấu trúc
  - Nhập 1 người vào vùng nhớ vừa cấp
  - Gán địa chỉ cho người sau
- Duyệt các phần tử có trong danh sách:
  - Gán con trỏ p = pdau
  - Chuyển đến người tiếp theo p->tiep
  - Kiểm tra người cuối cùng nhờ thông tin NULL

7.10. CẤU TRÚC TỰ TRỎ VÀ DANH SÁCH LIÊN KẾT (tiếp theo)

- Loại bớt thông tin về người:
  - Lưu trữ địa chỉ cần loại vào con trỏ
  - Giải phóng con trỏ free(p)
- Chèn thêm thông tin về người:
  - Cấp phát thêm bộ nhớ
  - Nhập bổ sung thông tin
  - Sửa thành phần con trỏ, cho phép truy cập tới người tiếp theo
- Xem ví dụ minh họa cụ thể thông qua chương trình C

TÓM LƯỢC CUỐI BÀI

- Bài 7 trình bày về cấu trúc và hợp. Nó cho phép lưu trữ các thông tin định dạng văn bản và bảng biểu theo cấu trúc.
- Có thể dùng cấu trúc để lưu trữ thông tin, đóng vai trò như 1 cơ sở dữ liệu của chương trình
- Việc cấp phát bộ nhớ cấu trúc động sẽ cho phép xử lý dữ liệu vào và ra linh hoạt hơn, đồng thời tiết kiệm bộ nhớ.