

60%

# Các kiểu dữ liệu nâng cao và Phân loại

Chỉ dành cho Trung tâm Aptech sử dụng

## Phiên 11



# Mục tiêu - 1

---

Giải thích các cấu trúc và cách sử dụng của chúng Định

nghĩa các cấu trúc Khai báo các

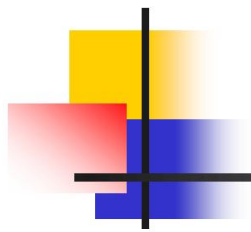
biến cấu trúc Giải thích cách truy cập các phần tử cấu trúc Giải thích cách khởi tạo các cấu

trúc Giải thích cách sử dụng các câu lệnh gán với cấu trúc

Giải thích cách các cấu trúc có thể được truyền như là đối số cho chức năng

Sử dụng mảng cấu trúc

Giải thích việc khởi tạo mảng cấu trúc



# Mục tiêu - 2

---

Giải thích các con trỏ tới các cấu trúc

Giải thích cách con trỏ cấu trúc có thể được truyền như đối số  
cho các hàm Chỉ dành cho Trung tâm Aptech sử dụng

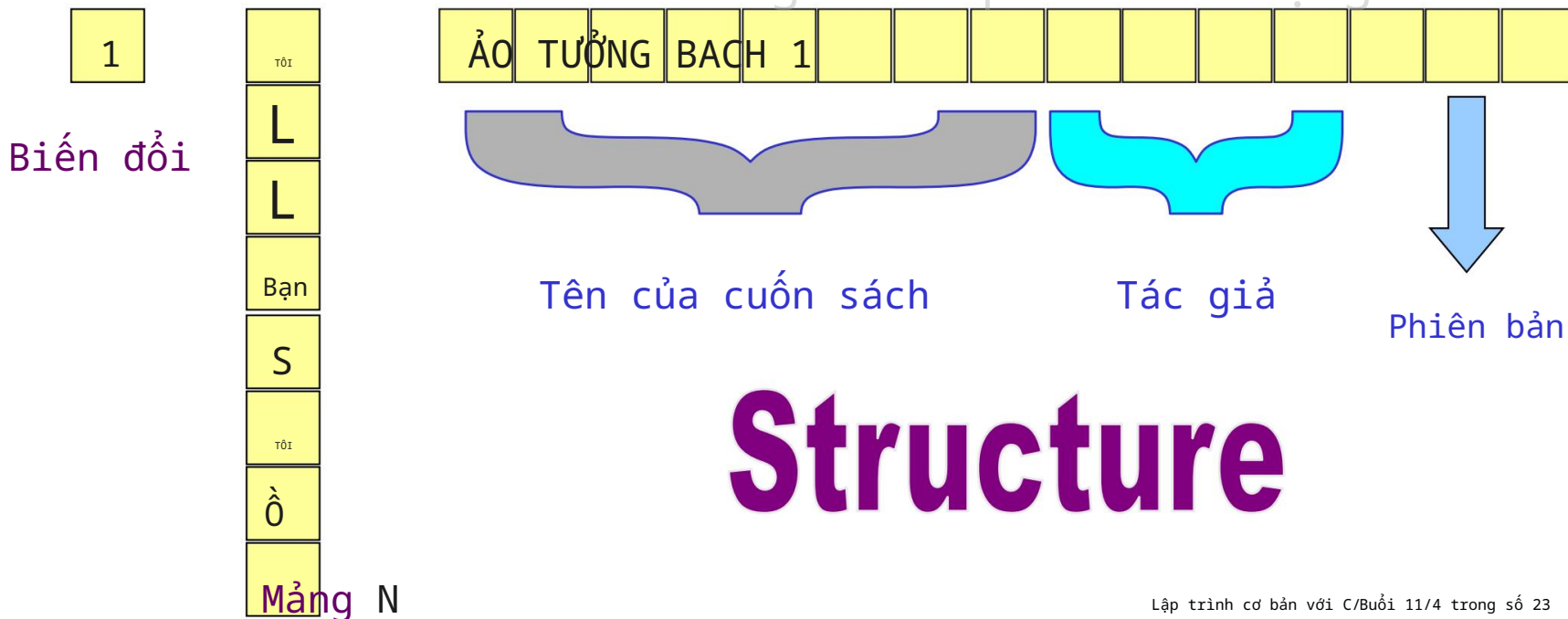
Giải thích từ khóa typedef

Giải thích sắp xếp mảng bằng sắp xếp chọn lọc và  
Phương pháp sắp xếp nổi bọt

# Cấu trúc

Một cấu trúc bao gồm một số mục dữ liệu, không nhất thiết phải cùng một kiểu dữ liệu, được nhóm lại với nhau. Cấu trúc có thể chứa nhiều mục này tùy ý.

Chỉ dành cho Trung tâm Aptech sử dụng





# Xác định một cấu trúc

---

Định nghĩa cấu trúc tạo thành một khuôn mẫu để tạo các biến cấu trúc

Các biến trong cấu trúc được gọi là **cấu trúc**  
**các thành phần** hoặc **cấu trúc thành viên**

Ví dụ:

**cấu trúc mèo**

```
{    char bk_name [25]; char tác giả  
    [20]; int edn; giá thả nổi;  
  
};
```



# Khai báo biến cấu trúc

Sau khi cấu trúc đã được xác định, một hoặc nhiều biến của kiểu đó có thể được khai báo

Ví dụ: `struct cat books1;`

Câu lệnh này dành đủ bộ nhớ để lưu giữ tất cả các mục trong cấu trúc

Other ways

Vì

```
struct cat { char bk_name[25];
             char tác giả[20];
             ấn bản quốc tế;
             giá cả nổi;
} sách1, sách2;
```

cấu trúc cat books1, books2;

hoặc `struct cat books1;`  
cấu trúc cat books2;



# Truy cập các phần tử cấu trúc

---

Các phần tử cấu trúc được tham chiếu thông qua việc sử dụng toán tử dấu chấm (.), còn được gọi là toán tử thành viên

Cú pháp: Chỉ dành cho Trung tâm Aptech sử dụng

`structure_name.element_name`

Ví dụ:

```
scanf( "%s" , books1.bk_name);
```



# Khởi tạo cấu trúc

---

Giống như biến và mảng, biến cấu trúc có thể được khởi tạo tại thời điểm khai báo **struct**

```
employee { int
```

```
no; Chỉ dành cho Trung tâm Aptech sử dụng
```

```
tên ký tự [20];
```

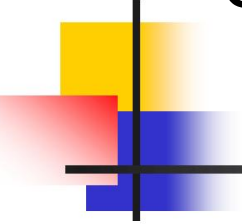
```
};
```

Biến emp1 và emp2 có kiểu nhân viên có thể được khai báo và khởi tạo như

sau: **struct employee emp1 = {346, "Abraham"};**

**cấu trúc nhân viên emp2 = {347, "John"};**





# Các câu lệnh gán được sử dụng với Structures-1

---

Có thể gán giá trị của một biến cấu trúc cho một biến khác cùng kiểu bằng cách sử dụng một câu lệnh gán đơn giản

Chỉ dành cho Trung tâm Aptech sử dụng

Ví dụ, nếu books 1 và books2 là các biến cấu trúc cùng loại, thì câu lệnh sau là hợp lệ

```
sách2 = sách1;
```

## Các câu lệnh gán được sử dụng với cấu trúc - 2

---

Trong trường hợp không thể gán trực tiếp, có thể sử dụng hàm `memcpy()` tích hợp sẵn

Cú pháp: *Chỉ dành cho Trung tâm Aptech sử dụng*

```
memcpy (char * đích, char & nguồn, int nbyte);
```

Ví dụ:

```
memcpy (&books2, &books1, sizeof(cấu trúc cat));
```



# Cấu trúc trong Cấu trúc

---

Có thể có một cấu trúc bên trong một cấu trúc khác.

Một cấu trúc không thể được lồng vào bên trong chính nó

vấn đề cấu trúc

{

```
char người vay [20]; char  
dt_of_issue[8]; struct cat  
books; }issl;
```

Chỉ dành cho Trung tâm Aptech sử dụng

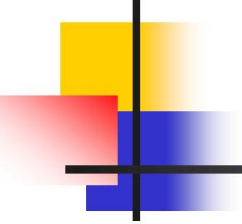
Để truy cập các thành phần của cấu trúc, định dạng sẽ tương tự như

cái được sử dụng với các cấu trúc bình thường,

`issl.người vay`

Để truy cập các thành phần của cấu trúc cat, là một phần của vấn đề cấu trúc khác,

`issl.books.author`



# Cấu trúc vượt qua như Lập luận

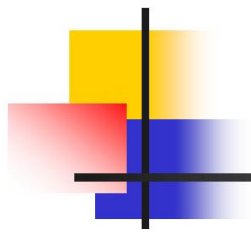
---

Một biến cấu trúc có thể được truyền như một đối số cho một hàm

Chỉ dành cho Trung tâm Aptech sử dụng

Tiện ích này được sử dụng để truyền các nhóm dữ liệu có liên quan về mặt logic với nhau thay vì truyền từng dữ liệu một

Kiểu của đối số phải phù hợp với kiểu của tham số



# Mảng cấu trúc

---

Một cách sử dụng phổ biến của cấu trúc là trong các mảng cấu trúc. Đầu tiên, một cấu trúc được định nghĩa, sau đó một biến mảng có kiểu đó được khai báo. Ví dụ: `Chữ dành cho Trung tâm Aptech sử dụng`

```
struct cat books[50];
```

Để truy cập biến tác giả của phần tử thứ tư trong mảng books:

```
sách[4].tác giả
```

# Khởi tạo cấu trúc

## Mảng

Mảng cấu trúc được khởi tạo bằng cách bao quanh danh sách giá trị của các phần tử của nó trong một cặp dấu ngoặc nhọn

Ví dụ:

đơn vị cấu trúc

```
{      char ch;
```

```
      số nguyên i;
```

```
};
```

cấu trúc đơn vị chuỗi [3] =

```
{      {'một', 100}
```

```
      {'b', 200}
```

```
      {'c', 300}
```

```
};
```



# Con trỏ đến Cấu trúc

---

Con trỏ cấu trúc được khai báo bằng cách đặt dấu hoa thị (\*) trước tên biến cấu trúc

Toán tử -> được sử dụng để truy cập các phần tử của một cấu trúc bằng cách sử dụng một con trỏ. Ví dụ:

```
cấu trúc cat *ptr_bk; ptr_bk  
= &books; printf("%s",  
ptr_bk->author);
```

Con trỏ cấu trúc được truyền dưới dạng đối số cho hàm cho phép hàm sửa đổi trực tiếp các phần tử cấu trúc



# Từ khóa `typedef`

---

Tên kiểu dữ liệu mới có thể được định nghĩa bằng cách sử dụng từ khóa `typedef`. Nó

không tạo ra một kiểu dữ liệu mới, nhưng định nghĩa một tên mới cho một kiểu hiện có. Cú pháp:

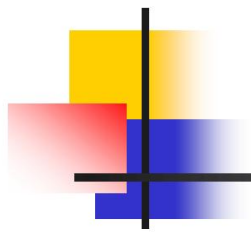
```
typedef tên kiểu;
```

Ví dụ:

```
typedef float deci;
```

`typedef` không thể được sử dụng với các lớp lưu trữ





# Sắp xếp mảng

---

Sắp xếp bao gồm việc sắp xếp dữ liệu mảng theo thứ tự được chỉ định như tăng dần hoặc giảm dần. Dữ liệu trong một mảng dễ tìm kiếm hơn khi mảng

đã được sắp xếp

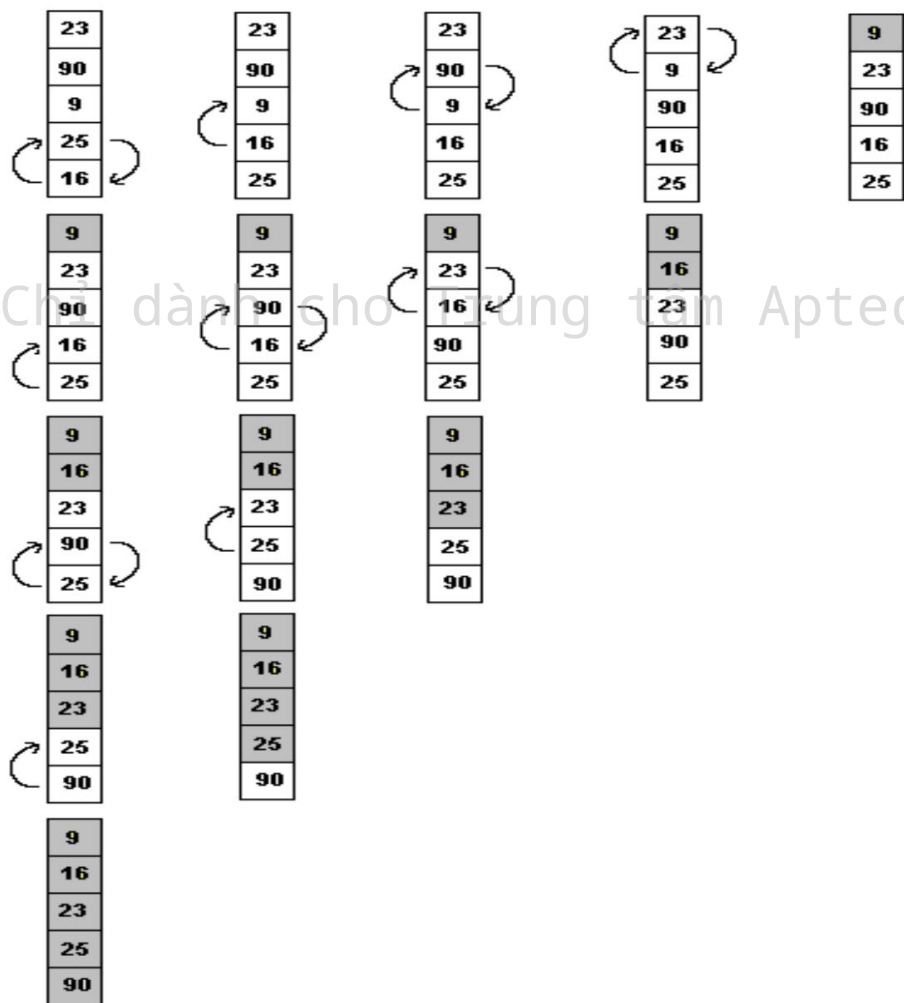
Chỉ dành cho Trung tâm Aptech sử dụng

Có hai phương pháp để sắp xếp mảng - Sắp xếp chọn lọc và

Sắp xếp theo bong bóng

Trong phương pháp sắp xếp chọn lọc, giá trị có trong mỗi phần tử được so sánh với các phần tử tiếp theo trong mảng để có được giá trị nhỏ nhất/lớn nhất. Trong phương pháp sắp xếp nổi bọt, các phép so sánh bắt đầu từ phần tử dưới cùng và các phần tử nhỏ hơn nổi lên trên cùng.

# Sắp xếp bong bóng-1





# Sắp xếp bong bóng-2

---

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

```
hàm main() {
```

```
    int i, j, temp, arr_num[5] = { 23, 90, 9, 25, 16};
```

```
    clrscr();
```

```
    for(i=3;i>=0;i--) /* Theo dõi mọi lần đi qua */
```

```
        for(j=4;j>=4-i;j--) /* So sánh các phần tử */
```

```
            {        nếu(arr_num[j]<arr_num[j-1])
```

```
                { temp=arr_num[j];
```

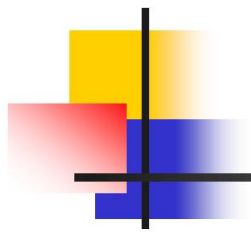
```
                    arr_num[j]=arr_num[j-1];
```

```
                    arr_num[j-1]=temp; }
```

```
    }
```

Ví dụ

Tiếp theo. . .



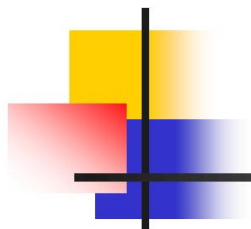
# Sắp xếp bong bóng-3

---

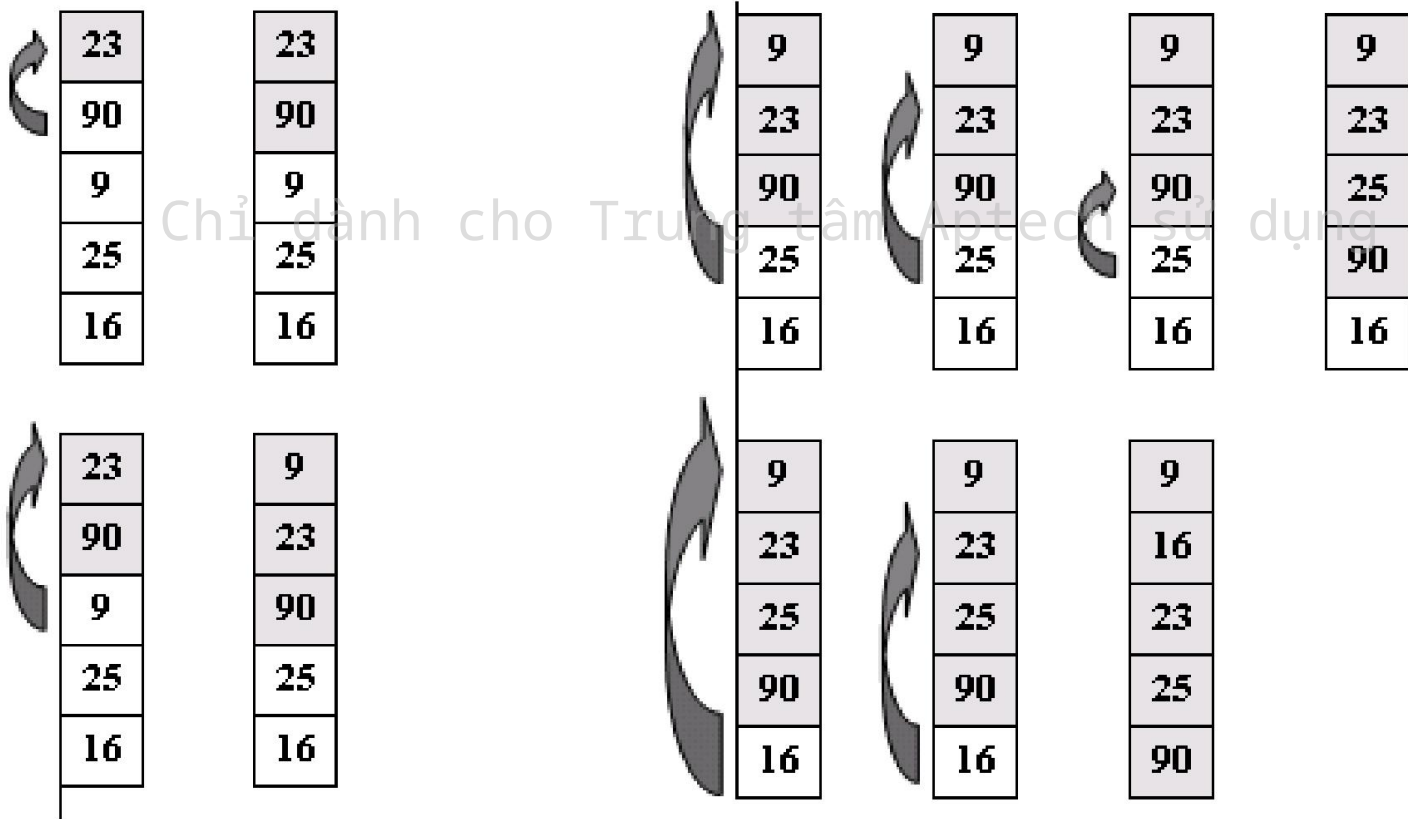
```
printf("\nMảng đã được sắp  
xếp");  
    for(i=0;i<5;i++) printf("\n%d", arr_num[i]);
```

Ví dụ

lấy(); } Chỉ dành cho Trung tâm Aptech sử dụng



# Sắp xếp chèn-1





# Sắp xếp chèn-2

---

```
#include<stdio.h>

void main() {

    int i, j, arr[5] = { 23, 90, 9, 25, 16 };
    char c;
    clrscr();
    /*Lặp lại để so sánh từng phần tử của phần chưa được sắp xếp của mảng*/
    for(i=1; i<5; i++)
        /*Lặp lại cho từng phần tử trong phần được sắp xếp của mảng*/
        for(j=0, flag='n'; j<i && flag=='n'; j++) {

            nếu(arr[j]>arr[i])
                /*Gọi hàm để chèn số*/ { insertnum(arr, i, j);
            flag='y';

        }

    }

    printf("\n\nMảng đã được sắp
    xếp\n"); for(i=0;
        i<5; i++) printf("%d\t", arr[i]);

    lấy();

}
```



# Sắp xếp chèn-3

---

```
insertnum(int arrnum[], int x, int y)
{
    int nhiệt độ;
    /*Lưu số cần chèn*/ temp=arrnum[x];

    /*Lặp để đẩy phần đã sắp xếp của mảng xuống từ vị trí cần chèn số*/
    for(;x>y; x--) arrnum[x]=arrnum[x-1];

    /*Chèn số*/ arrnum[x]=temp;
}
```