Các kiểu dữ liệu nâng cao và Phân loại

Phiên 11

Mục tiêu - 1

- Giải thích các cấu trúc và cách sử dụng của chúng
- Xác định cấu trúc
- Khai báo các biến cấu trúc
- Giải thích cách truy cập các thành phần cấu trúc
- Giải thích cách khởi tạo cấu trúc
- Giải thích cách sử dụng các câu lệnh gán với các cấu trúc
- Giải thích cách các cấu trúc có thể được truyền như là đối số cho các hàm
- Sử dụng mảng cấu trúc
- Giải thích việc khởi tạo các mảng cấu trúc

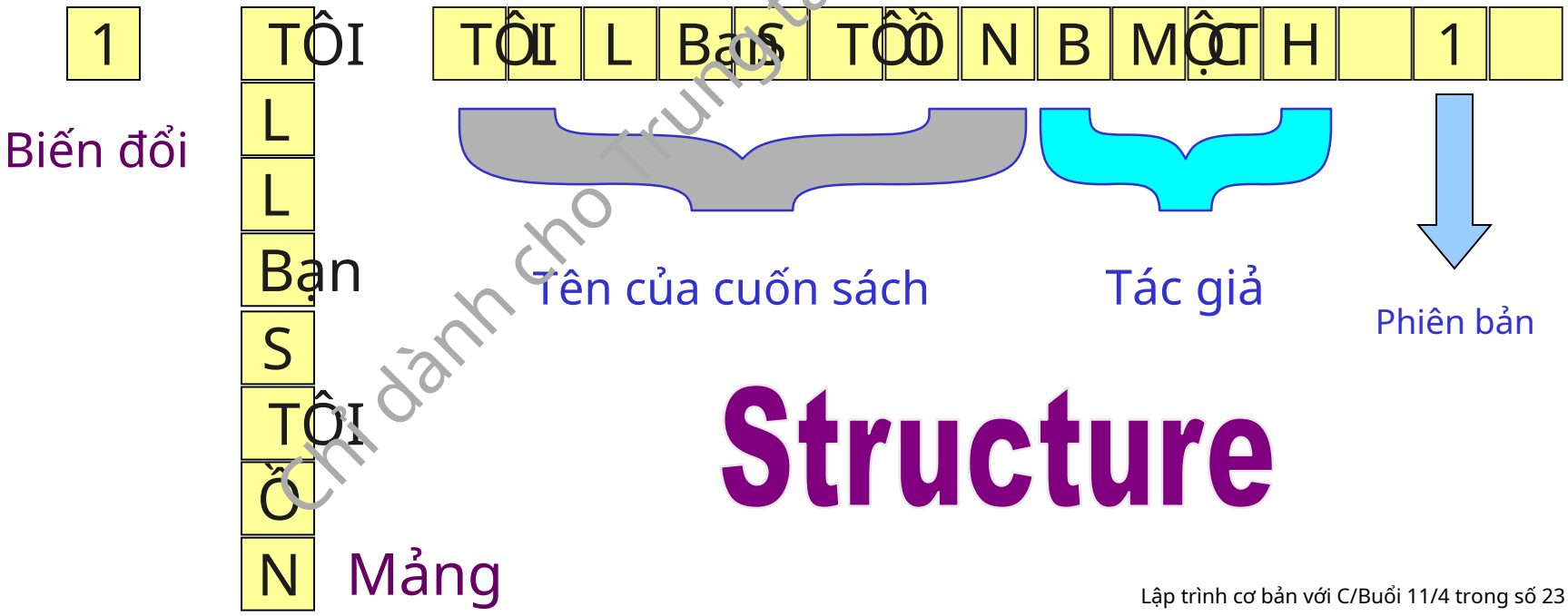
Mục tiêu - 2

- 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
- Giải thích từ khóa typedef
- Giải thích về sắp xếp mảng bằng phương pháp Selection Sort và Bubble Sort

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 này có thể chứa nhiều vật phẩm như mong muốn



Xác định một cấu trúc

- Định nghĩa cấu trúc tạo thành một mẫu để tạo các biến cấu trúc
- Các biến trong cấu trúc được gọi làcác yếu tố cấu trúc hoặcthành viên cấu trúc
- Ví dụ:

```
cấu trúc mèo
    {       char   tên_bk     [25];
            char   tác giả    [20];
            số nguyên   biên tập;
            trôi nổi   giá;
    };
```

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:**cấu trúc cat books1;**
- Câu lệnh này dành đủ bộ nhớ để lưu trữ tất cả các mục trong cấu trúc

Other ways

cấu trúc mèo {
char bk_name[25];
char tác giả[20];
ấn bản quốc tế;
giá thả nổi;
} sách1, sách2;

cấu trúc cat books1, books2;

hoặc

cấu trúc cat books1;
cấu trúc cat books2;

Truy cập các phần tử cấu trúc

- Các thành phần cấu trúc được tham chiếu thông qua việc sử dụng **toán tử chấm**(.),còn được gọi là**nhà điều hành thành viên**
- Cú pháp:

structure_name.element_name

- Ví dụ:

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

Khởi tạo cấu trúc

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

```
cấu trúc nhân viên
  { số nguyên;
    tên ký tự [20];
  };
```

- Biến số **emp1** Và **emp2** của loại **người lao động** có thể được khai báo và khởi tạo như sau:

```
cấu trúc nhân viên emp1 = {346, “Abraham”};
cấu trúc nhân viên emp2 = {347, “John”};
```

Các câu lệnh gán được sử dụng với Cấu trúc-1

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

- Ví dụ, nếu**sách 1**Và**sách2**là các biến cấu trúc cùng loại, câu lệnh sau là hợp lệ

sách3 = sách1;

Các câu lệnh gán được sử dụng với Cấu trúc - 2

- Trong trường hợp không thể gán trực tiếp, hàm tích hợp **memcpy()**có thể được sử dụng

- Cú pháp:

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

- **Để truy cập các thành phần của cấu trúc, định dạng sẽ tương tự như định dạng được sử dụng với các cấu trúc thông 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

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

- Kiểu của đối số phải khớp với kiểu của tham số

Mảng cấu trúc

- Một cách sử dụng phổ biến của các 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:

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

- Để truy cập biến tác giả của phần tử thứ tư của mảng **sách**:

```
sách[4].tác giả
```

Khởi tạo cấu trúc Mảng

- Mảng cấu trúc được khởi tạo bằng cách bao gồm danh sách các 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] = {'a',
{       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
- Các->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 = &sách;
printf("%s", ptr_bk->tác giả);
```

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

Cácđịnh nghĩa kiểutừ khóa

- Tên kiểu dữ liệu mới có thể được định nghĩa bằng cách sử dụng từ khóa định nghĩa kiểu
- Nó không tạo ra một kiểu dữ liệu mới, nhưng định nghĩa một tên mới cho một kiểu dữ liệu hiện có
- Cú pháp:

  typedef tên kiểu;

- Ví dụ:

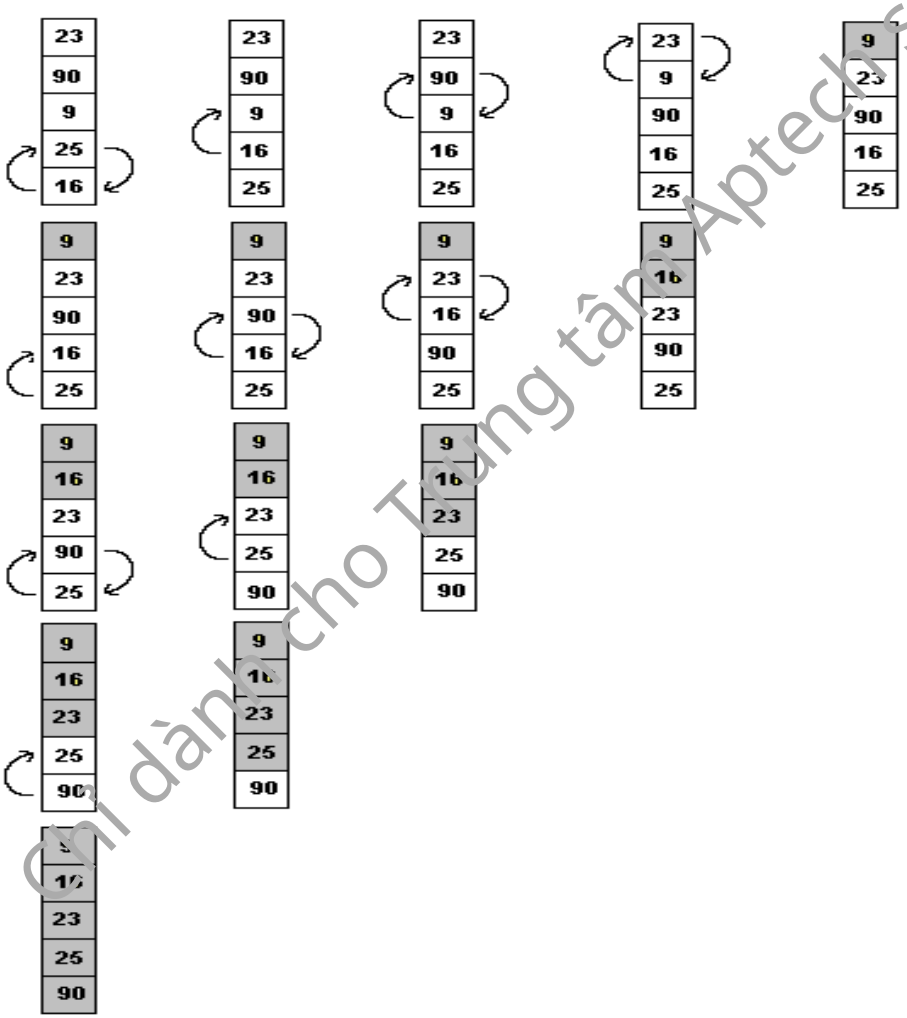
  typedef float deci;

- 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 sẽ dễ tìm kiếm hơn khi mảng được sắp xếp
- Có hai phương pháp để sắp xếp mảng – Selection Sort và Bubble Sort
- Trong phương pháp sắp xếp lựa chọn, 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 bọt lên trên cùng

Sắp xếp bong bóng-1



Sắp xếp bong bóng-2

Ví dụ

```
# bao gồm <stdio.h>

hàm main() không có giá trị
{
   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;
                           }
                  }
```

Tiếp theo.....

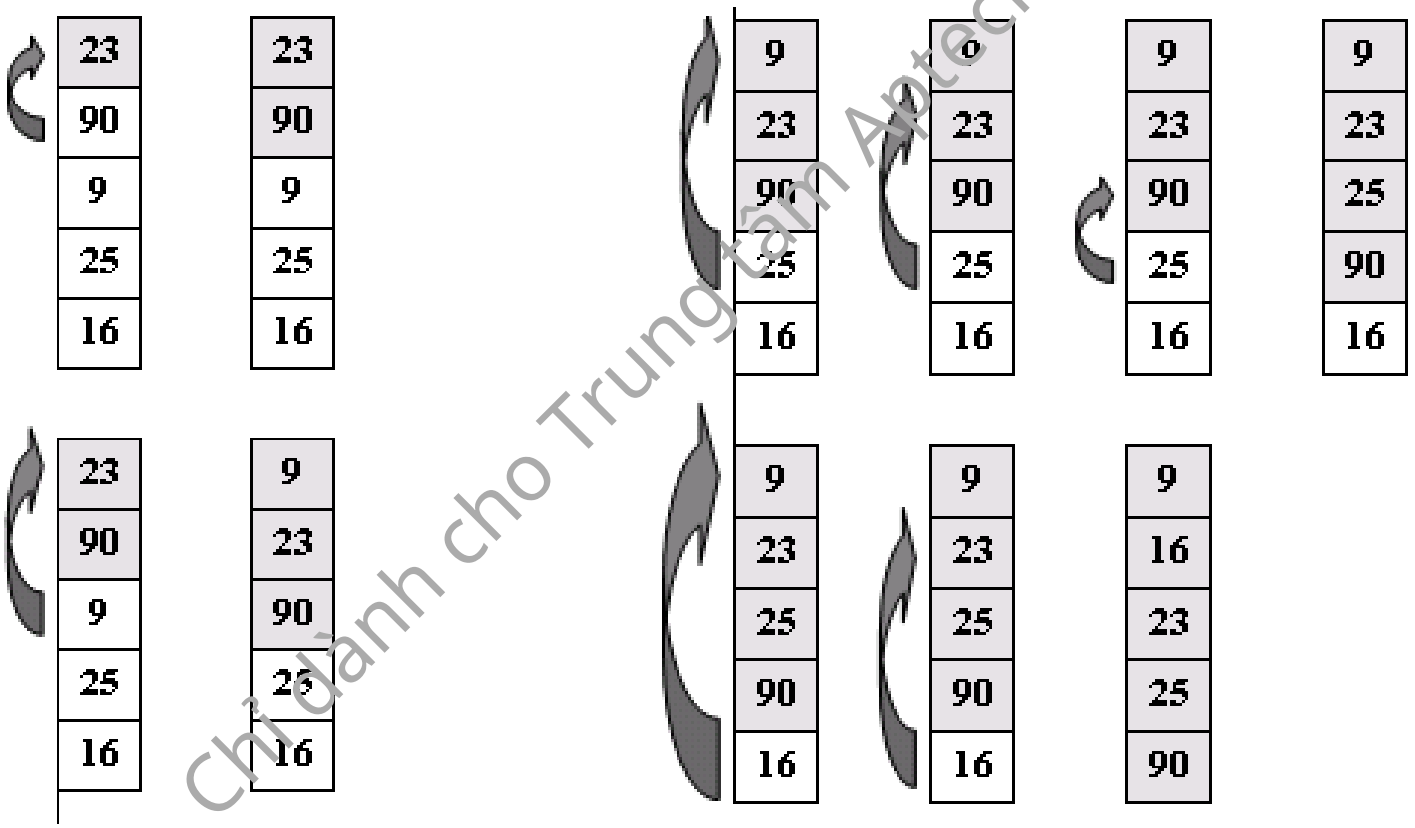
Sắp xếp bong bóng-3

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

    lấy();
}
```

Ví dụ

Sắp xếp chèn-1



Sắp xếp chèn-2

```
# bao gồm <stdio.h>

hàm main() không có giá trị
{
        int i, j, arr[5] = { 23, 90, 9, 25, 16 }; cờ
        char;
        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 đã 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(mảng, i, j);
                                cờ='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 lại để đẩy phần đã sắp xếp của mảng xuống từ vị trí cần chèn
        số*/
        đối với(;x>y; x--)
                arrnum[x]=arrnum[x-1];
        /*Nhập số*/
        arrnum[x]=temp;
}
```