BACHKHOA EDUCATION APTECH
Làm trước học sau

Aptech COMPUTER EDUCATION

TRƯỜNG ĐÀO TẠO LẬP TRÌNH VIÊN VÀ QUẢN TRỊ MẠNG QUỐC TẾ BACHKHOA-APTECH

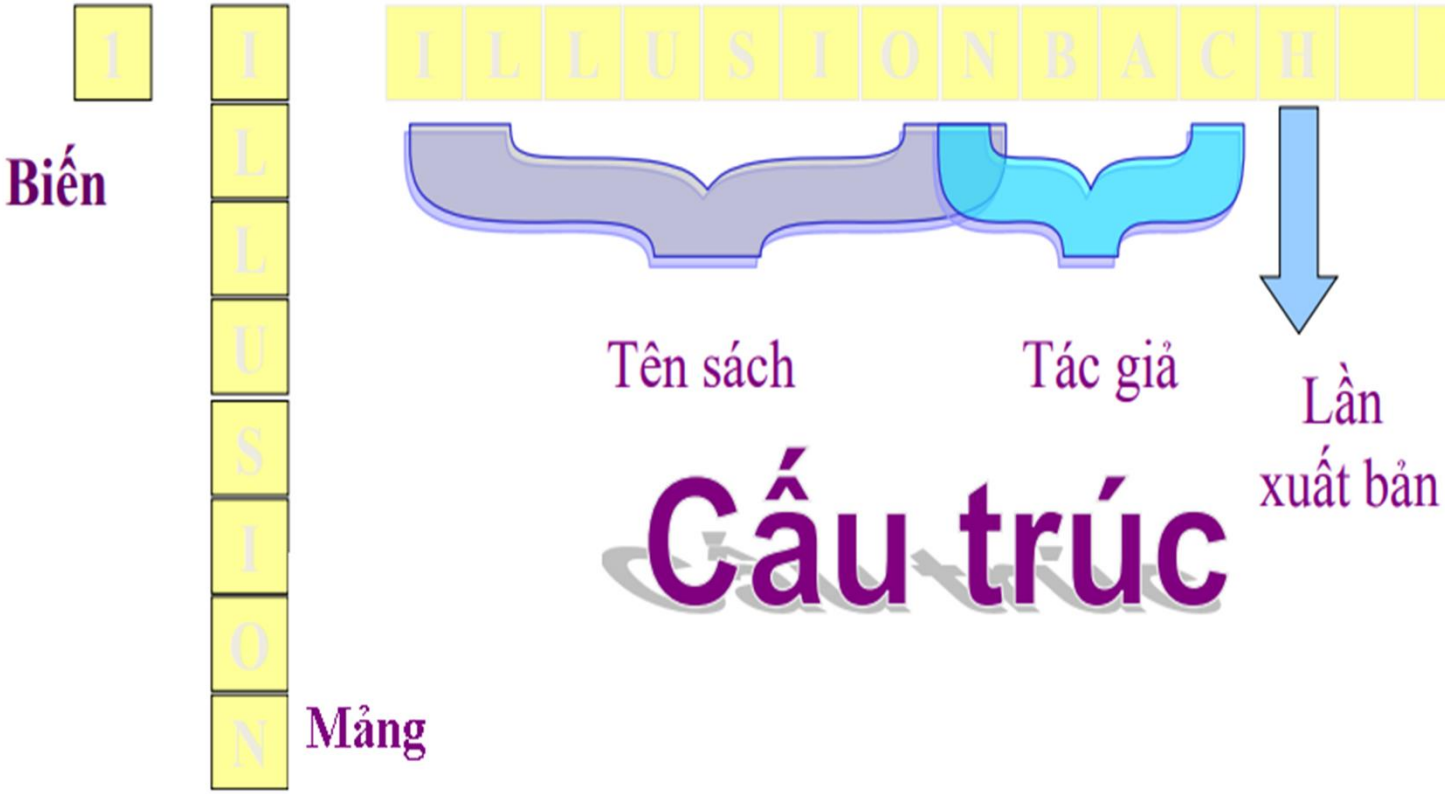
BÀI 11:
CÁC KIỂU DỮ LIỆU NÂNG CAO – SẮP XẾP

MỤC TIÊU BÀI HỌC

- ✓ Tìm hiểu kiểu dữ liệu cấu trúc và công dụng
- ✓ Định nghĩa cấu trúc
- ✓ Khai báo các biến kiểu cấu trúc
- ✓ Cách truy cập vào các phần tử của cấu trúc
- ✓ Khởi tạo biến cấu trúc
- ✓ Sử dụng biến cấu trúc trong câu lệnh gán
- ✓ Cách truyền tham số cấu trúc
- ✓ Sử dụng mảng các cấu trúc
- ✓ Tìm hiểu cách khởi tạo mảng các cấu trúc
- ✓ Con trỏ cấu trúc
- ✓ Cách truyền tham số kiểu con trỏ cấu trúc
- ✓ Tìm hiểu từ khóa typedef
- ✓ Sắp xếp mảng bằng phương pháp Bubble sort và Insertion sort.

CẤU TRÚC

- Một cấu trúc bao gồm các mẫu dữ liệu, không nhất thiết cùng kiểu, được nhóm lại với nhau.
- Một cấu trúc có thể bao gồm nhiều mẫu dữ liệu như vậy.



ĐỊNH NGHĨA CẤU TRÚC

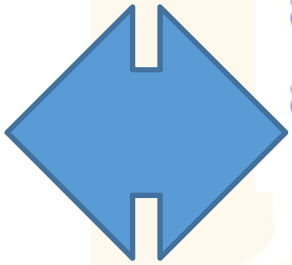
- Việc định nghĩa cấu trúc sẽ tạo ra kiểu dữ liệu mới cho phép người dùng sử dụng chúng để khai báo các biến kiểu cấu trúc .
- Các biến trong cấu trúc được gọi là các phần tử của cấu trúc hay thành phần của cấu trúc
- Ví dụ:

```
struct book{
    char bk_name[25];
    char author[20];
    int edn;
    float price;
};
```

KHAI BÁO BIẾN CẤU TRÚC

- ✓ Khi một cấu trúc đã được định nghĩa, chúng ta có thể khai báo một hoặc nhiều biến kiểu này.
- ✓ Ví dụ: **struct cat books1;**
- ✓ Câu lệnh này sẽ dành đủ vùng nhớ để lưu trữ tất cả các mục trong một cấu trúc.

```
struct book{
    char bk_name[25];
    char author[20];
    int edn;
    float price;
} book1, book2;
```



```
struct book book1, book2;
struct book book1;
struct book book2;
```

TRUY CẬP PHẦN TỬ CỦA CẤU TRÚC

- ✓ Các phần tử của cấu trúc được truy cập thông qua việc sử dụng **toán tử chấm** (.), toán tử này còn được gọi là **toán tử thành viên – membership**
- ✓ Cú pháp:

  structure_name.element_name

- ✓ Ví dụ:

  scanf(“%s”, books1.bk_name);

KHỞI TẠO CẤU TRÚC

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

```
struct employee{
    int no;
    char name[20];
};
```

- ✓ Các biến emp1 và emp2 có kiểu employee có thể được khai báo và khởi tạo như sau:

struct employee emp1 = {346, “Abraham”};

struct employee emp2 = {347, “John”};

CÂU LỆNH GÁN SỬ DỤNG CÁC CẤU TRÚC

- ✓ Có thể sử dụng câu lệnh gán đơn giản để gán giá trị của một biến cấu trúc cho một biến khác có cùng kiểu
- ✓ Chẳng hạn, nếu **books1** và **books2** là các biến cấu trúc có cùng kiểu, thì câu lệnh sau là hợp lệ: **books2 = books1;**
- ✓ Trong trường hợp không thể dùng câu lệnh gán trực tiếp, thì có thể sử dụng hàm tạo sẵn **memcpy()**
- ✓ Cú pháp: **memcpy (char * destn, char &source, int nbytes);**
- ✓ Ví dụ:: **memcpy (&books2, &books1, sizeof(struct cat));**

CẤU TRÚC LỒNG TRONG CẤU TRÚC

- Một cấu trúc có thể lồng trong một cấu trúc khác. Tuy nhiên, một cấu trúc không thể lồng trong chính nó.

```
struct book{
    int name[20];
    char author[20];
};
struct issue{
    char borrower[20];
    char dt_of_issue[8];
    struct books;
} issl;
```

- c phần tử của cấu trúc này tương tự như với cấu trúc bình thường khác,

  issl.borrower

- Để truy cập vào phần tử của cấu trúc cat là một phần của cấu trúc issl ,

  issl.books.author

TRUYỀN THAM SỐ KIỂU CẤU TRÚC

- ✓Tham số của hàm có thể là một cấu trúc.
- ✓Là một phương tiện hữu dụng khi muốn truyền một nhóm các thành phần dữ liệu có quan hệ logic với nhau thông qua một biến thay vì phải truyền từng thành phần một
- ✓Kiểu của tham số thực phải trùng với kiểu của tham số hình thức.

MẢNG CẤU TRÚC

- ✓ Một áp dụng thường gặp là mảng cấu trúc
- ✓ Một kiểu cấu trúc phải được định nghĩa trước, sau đó một biến mảng có kiểu đó mới được khai báo
- ✓ Ví dụ: **struct cat books[50];**
- ✓ Để truy cập vào thành phần author của phần tử thứ tư của mảng **books:**
- ✓ **books[4].author**

KHỞI TẠO MẢNG CẤU TRÚC

- ✓ Mảng cấu trúc được khởi tạo bằng cách liệt kê danh sách các giá trị phần tử của nó trong một cặp dấu móc
- ✓ Ví dụ:

```
struct book{
    char name[20];
    float price;
};
struct book bookstore[3] = {{'abc',100},{'bcd',200},{'def',150}};
```

CON TRỎ ĐẾN CẤU TRÚC

- ✓Con trỏ cấu trúc được khai báo bằng cách đặt dấu * trước tên của biến cấu trúc.
- ✓Toán tử -> được dùng để truy cập vào các phần tử của một cấu trúc sử dụng một con trỏ
- ✓Con trỏ cấu trúc được truyền vào hàm, cho phép hàm thay đổi trực tiếp các phần tử của cấu trúc.

CON TRỎ ĐẾN CẤU TRÚC

```
1  #include <stdio.h>
2  #include <string.h>
3
4  struct Books{
5     char  name[50];
6     char  author[50];
7     char  catalog[100];
8     int   id;
9  };
10
11 /* khai bao ham */
12 void showBookInfo( struct Books *book );
13 int main( )
14 {
15    struct Books Book1;        /* Khai bao Book1 la cua kieu Book */
16    struct Books Book2;        /* Khai bao Book2 la cua kieu Book */
17
18    /* thong tin chi tiet quyen sach thu nhat */
      strcpy( Book1.name, "C Programing Language");
      strcpy( Book1.author, "Jackpot");
      strcpy( Book1.catalog, "Information Technology");
22    Book1.id = 1234567;
23
24    /* thong tin chi tiet quyen sach thu hai */
      strcpy( Book2.name, "Gone With The Wind");
      strcpy( Book2.author, "Margaret Mitchell");
      strcpy( Book2.catalog, "Oscar");
28    Book2.id = 6677028;
```

CON TRỎ ĐẾN CẤU TRÚC

```
29
30      /* in thong tin Book1 bang cach truyen dia chi cua Book1 */
31      showBookInfo( &Book1 );
32
33      /* in thong tin Book2 bang cach truyen dia chi cua Book2 */
34      showBookInfo( &Book2 );
35
36      return 0;
37  }
38  void showBookInfo( struct Books *book )
39  {
40      printf( "Tieu de sach: %s\n", book->name);
41      printf( "Tac gia: %s\n", book->author);
42      printf( "Chu de: %s\n", book->catalog);
43      printf( "Book ID: %d\n", book->id);
44  }
```

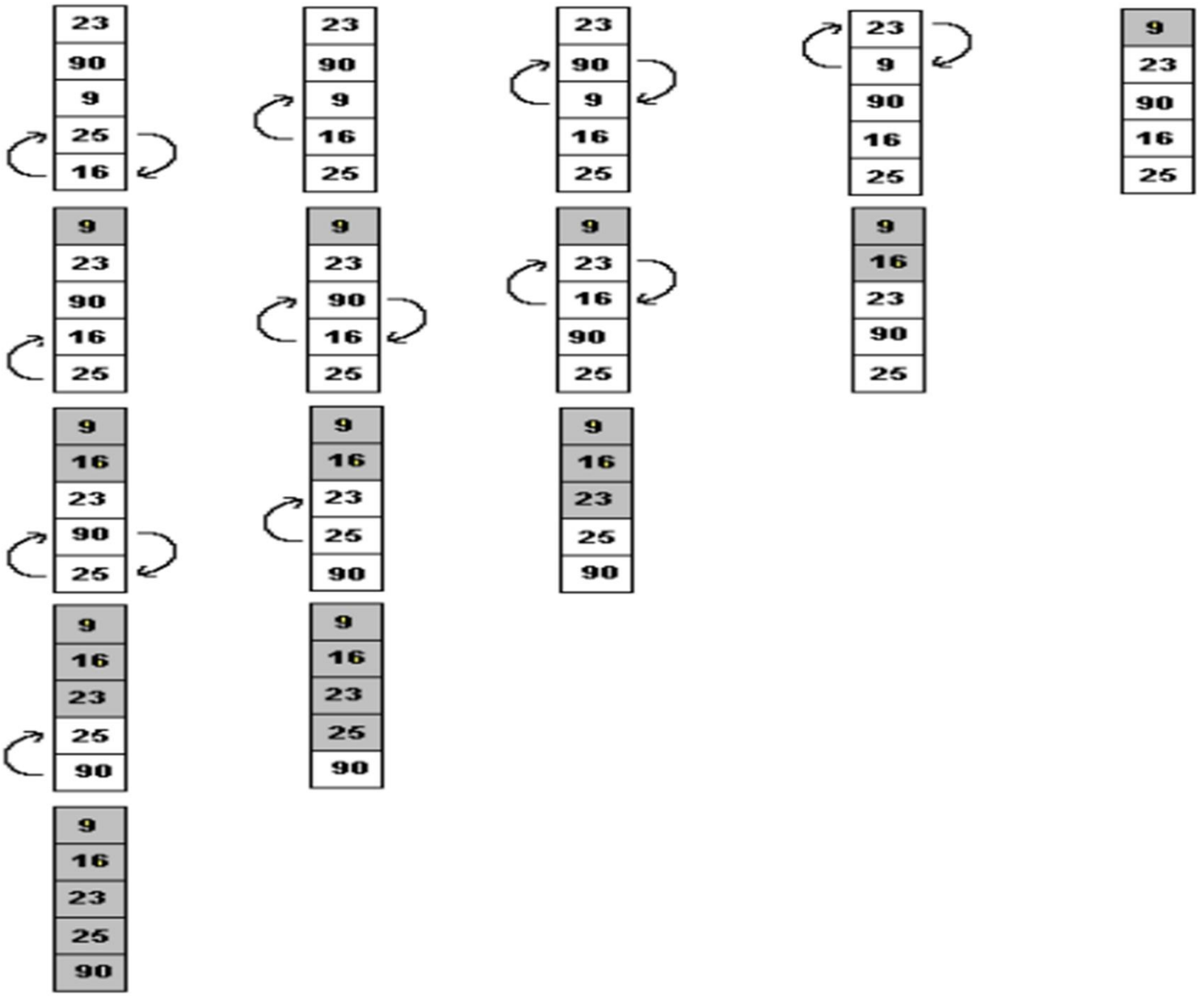
TỪ KHÓA typedef

- ✓Một kiểu dữ liệu có thể được định nghĩa bằng cách sử dụng từ khóa typedef
- ✓Nó không tạo ra một kiểu dữ liệu mới, mà định nghĩa một tên mới cho một kiểu đã có.
- ✓Cú pháp: **typedef type name;**
- ✓Ví dụ: **typedef float deci;**
- ✓**typedef** không thể sử dụng với *storage classes*

SẮP XẾP MẢNG

- ✓ Sắp xếp liên quan đến việc thay đổi vị trí các phần tử theo thứ tự xác định như tăng dần hay giảm dần
- ✓ Dữ liệu trong mảng sẽ dễ dàng tìm thấy hơn nếu mảng được sắp xếp
- ✓ Hai phương pháp sắp xếp mảng được trình bày: Bubble Sort và Insertion Sort
- ✓ Trong phương pháp Bubble sort, việc so sánh bắt đầu từ phần tử dưới cùng và phần tử có giá trị nhỏ hơn sẽ chuyển dần lên trên (nổi bọt)
- ✓ Trong phương pháp Insertion sort, mỗi phần tử trong mảng được xem xét, và đặt vào vị trí đúng của nó giữa các phần tử đã được sắp xếp

BUBBLE SORT 1-4



BUBBLE SORT 2-4

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

#define MAX 10

int list[MAX] = {1, 8, 4, 6, 0, 3, 5, 2, 7, 9};

void display() {
    int i;
    printf("[");
    for (i = 0; i < MAX; i++) {
        printf("%d ", list[i]);
    }
    printf("]\n");
}

```

BUBBLE SORT 3-4

```
17  void bubbleSort() {
18      int temp;
19      int i, j;
20      bool swapped = false;
21      for (i = 0; i < MAX - 1; i++) {
22          swapped = false;
23          for (j = 0; j < MAX - 1 - i; j++) {
24              printf("So sanh cac phan tu: [ %d, %d ] ", list[j], list[j + 1]);
25              if (list[j] > list[j + 1]) {
26                  temp = list[j];
27                  list[j] = list[j + 1];
28                  list[j + 1] = temp;
29                  swapped = true;
30                  printf(" => trao doi [%d, %d]\n", list[j], list[j + 1]);
31              } else {
32                  printf(" => khong can trao doi\n");
33              }
34
35          }
36          if (!swapped) {
37              break;
38          }
39          printf("Vong lap thu %d#: ", (i + 1));
40          display();
41      }
42  }
43
```

BUBBLE SORT 4-4

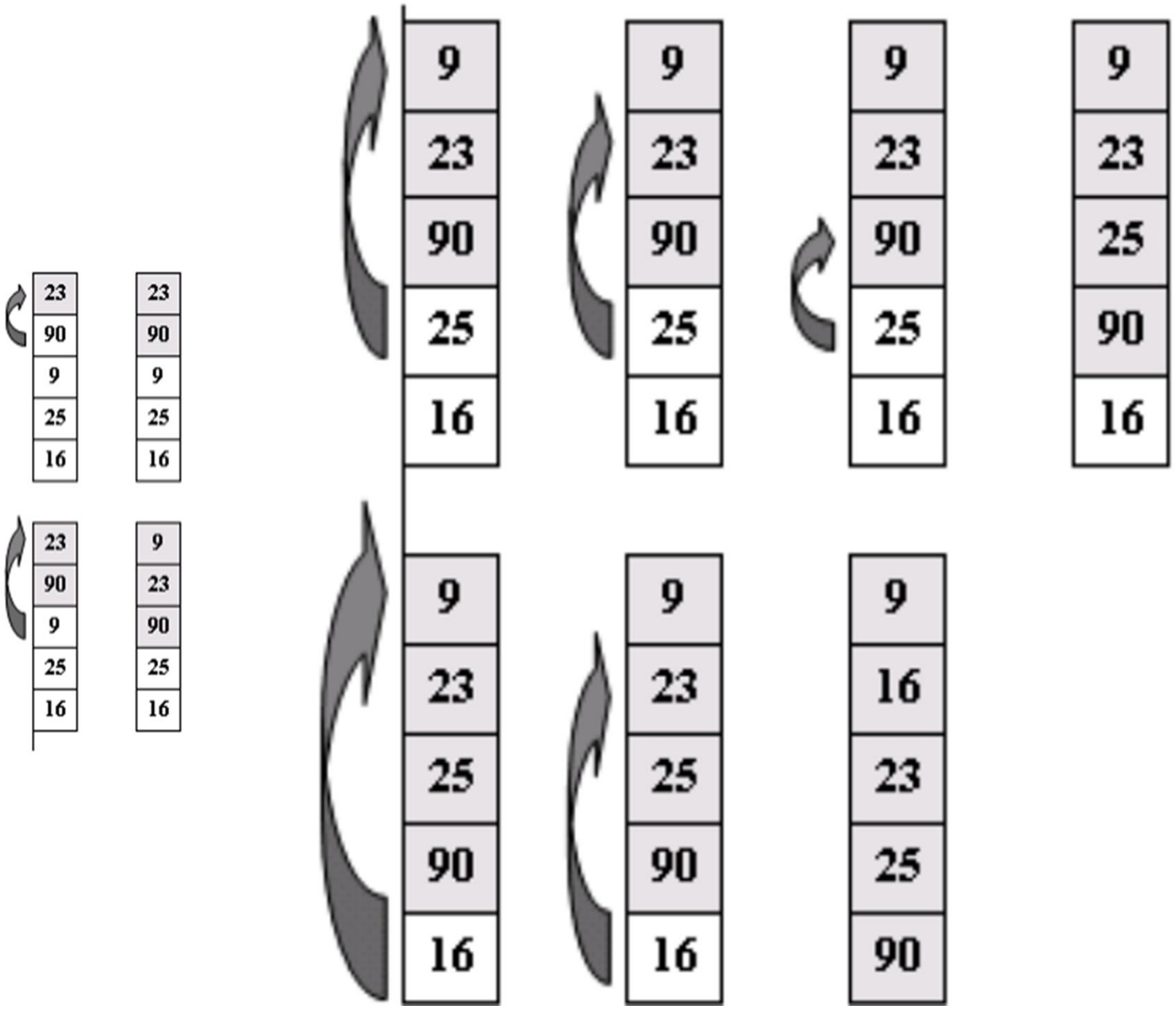
```
43
44  main() {
45      printf("Mang du lieu dau vao: ");
46      display();
        printf("\n");
48      bubbleSort();
49      printf("\nMang sau khi da sap xep: ");
50      display();
51  }
```

```
Mang du lieu dau vao: [1 8 4 6 0 3 5 2 7 9 ]

So sanh cac phan tu: [ 1, 8 ]  => khong can trao doi
So sanh cac phan tu: [ 8, 4 ]  => trao doi [4, 8]
So sanh cac phan tu: [ 8, 6 ]  => trao doi [6, 8]
So sanh cac phan tu: [ 8, 0 ]  => trao doi [0, 8]
So sanh cac phan tu: [ 8, 3 ]  => trao doi [3, 8]
So sanh cac phan tu: [ 8, 5 ]  => trao doi [5, 8]
So sanh cac phan tu: [ 8, 2 ]  => trao doi [2, 8]
So sanh cac phan tu: [ 8, 7 ]  => trao doi [7, 8]
So sanh cac phan tu: [ 8, 9 ]  => khong can trao doi
Vong lap thu 1#: [1 4 6 0 3 5 2 7 8 9 ]
So sanh cac phan tu: [ 1, 4 ]  => khong can trao doi
So sanh cac phan tu: [ 4, 6 ]  => khong can trao doi
So sanh cac phan tu: [ 6, 0 ]  => trao doi [0, 6]
So sanh cac phan tu: [ 6, 3 ]  => trao doi [3, 6]
So sanh cac phan tu: [ 6, 5 ]  => trao doi [5, 6]
So sanh cac phan tu: [ 6, 2 ]  => trao doi [2, 6]
So sanh cac phan tu: [ 6, 7 ]  => khong can trao doi
So sanh cac phan tu: [ 7, 8 ]  => khong can trao doi
Vong lap thu 2#: [1 4 0 3 5 2 6 7 8 9 ]
So sanh cac phan tu: [ 1, 4 ]  => khong can trao doi
So sanh cac phan tu: [ 4, 0 ]  => trao doi [0, 4]
So sanh cac phan tu: [ 4, 3 ]  => trao doi [3, 4]
So sanh cac phan tu: [ 4, 5 ]  => khong can trao doi
So sanh cac phan tu: [ 5, 2 ]  => trao doi [2, 5]
So sanh cac phan tu: [ 5, 6 ]  => khong can trao doi
So sanh cac phan tu: [ 6, 7 ]  => khong can trao doi
Vong lap thu 3#: [1 0 3 4 2 5 6 7 8 9 ]
So sanh cac phan tu: [ 1, 0 ]  => trao doi [0, 1]
So sanh cac phan tu: [ 1, 3 ]  => khong can trao doi
So sanh cac phan tu: [ 3, 4 ]  => khong can trao doi
So sanh cac phan tu: [ 4, 2 ]  => trao doi [2, 4]
So sanh cac phan tu: [ 4, 5 ]  => khong can trao doi
So sanh cac phan tu: [ 5, 6 ]  => khong can trao doi
Vong lap thu 5#: [0 1 2 3 4 5 6 7 8 9 ]
So sanh cac phan tu: [ 0, 1 ]  => khong can trao doi
So sanh cac phan tu: [ 1, 2 ]  => khong can trao doi
So sanh cac phan tu: [ 2, 3 ]  => khong can trao doi
So sanh cac phan tu: [ 3, 4 ]  => khong can trao doi

Mang sau khi da sap xep: [0 1 2 3 4 5 6 7 8 9 ]
Press [Enter] to close the terminal ...
```

INSERTION SORT 1-2



INSERTION SORT 2-2

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

int main() {
    int length = 5;
    int array[] = {5, 2, 4, 3, 1};
    int unsorted_id, i;
    for (unsorted_id = 1; unsorted_id < length; unsorted_id++) {
        int element = array[unsorted_id];
        int sorted_id = unsorted_id - 1;
        while (sorted_id >= 0 && element > array[sorted_id]) {
            array[sorted_id + 1] = array[sorted_id];
            sorted_id--;
        }
        array[sorted_id + 1] = element;
    }

    for (i = 0; i < length; ++i) {
        printf("%d ", array[i]);
    }
    getch();
    return 0;
}
```

TÓM TẮT BÀI HỌC

- ✓ Một cấu trúc là tập các biến có thể có kiểu dữ liệu khác nhau được nhóm lại với nhau dưới cùng một tên
- ✓ **Định nghĩa cấu trúc** sẽ tạo ra kiểu dữ liệu mới cho phép người dùng sử dụng chúng để khai báo các biến kiểu cấu trúc
- ✓ **Toán tử chấm (.)**, hay còn được gọi là **toán tử thành viên**
- ✓ Có thể có một cấu trúc nằm trong một cấu trúc khác
- ✓ Một biến cấu trúc có thể được truyền vào một hàm như là một tham số
- ✓ Cách sử dụng thông dụng nhất của cấu trúc là dưới hình thức mảng cấu trúc
- ✓ Toán tử -> được sử dụng để truy cập vào các phần tử của một cấu trúc thông qua một con trỏ trỏ đến cấu trúc đó
- ✓ Một kiểu dữ liệu mới có thể được định nghĩa bằng từ khóa **typedef**
- ✓ Hai phương pháp dùng để sắp xếp một mảng là **bubble sort** và **insertion sort**
- ✓ Trong **bubble sort**, giá trị của các phần tử được so sánh với giá trị của phần tử kế tiếp. Trong phương pháp này, các phần tử nhỏ hơn nổi lên dần, và cuối cùng mảng sẽ được sắp xếp
- ✓ Trong **insertion sort**, ta xét mỗi phần tử trong mảng và chèn vào vị trí đúng của nó giữa các phần tử đã được sắp xếp

BACHKHOA EDUCATION APTECH
Làm trước học sau

Aptech COMPUTER EDUCATION

TRƯỜNG ĐÀO TẠO LẬP TRÌNH VIÊN VÀ QUẢN TRỊ MẠNG QUỐC TẾ BACHKHOA-APTECH

THANK FOR WATCH!