جامعة المستقبل
AL MUSTAQBAL UNIVERSITY

كلية العلوم

قسم الأمن السيبراني

Lecture: 7

Structures

Subject: Structured Programming
First Stage
Lecturer: Asst. Prof. Dr. Ali Kadhum Al-Quraby

LECTURE [illegible]

1. Structures:

Structures are typically used to group several data items together to form a single entity. It is a collection of variables used to group variables into a single record. Thus a structure (the keyword **struct** is used in C++) is used. Keyword struct is a data-type, like the following C++ data-types (int, float, char, etc…). This is unlike the array, which all the variables must be the same type. The data items in a structure are called the members of the structure.

General Form of Structure:

```
struct  struct-name
{
   variables …
};
```

2. The Three Ways for Declare the Structure:

A.

```
#include <iostream.h>

struct  data
{
   char *name;
   int age;
};

void  main()
{
   struct  data  student;
```

B.

```
struct  data
{
      char *name;
      int age;
  } student;
```

C.

```
  typedef  struct
  {
      char *name;
      int age;
  } student;
```

- ➢ The above three ways are called structure specifier (tells how the structure is organized) or it is called structure declaration.
- ➢ To access elements in a structure, use a record selector **(.)**.

```
     student . name="ahmed";
     student . age=20;
        :
  }
```

Note: we can assign more than one name as a structure-name, to the one structure. For example:

```
  typedef  struct
  {
      char *name;
      int age;
  } student , lecturer;
```

Example 1:

This example uses parts inventory to demonstrate structures.

```
#include<iostream.h>

Struct part    //   specify a structure
{
Int   model_no;
Int   part_no;
Float  cost;
}

Void main()
{
Part p1;     //  define a structure variable.
P1.model_no=6244;
P1.part_no=373;
P1.cost=217.55;
Cout<<”/n model”<<p1.model_no;
Cout<<”, part”<<p1.part_no;
Cout<<”, cost”<<p1.cost;
}
```

The above program has three main aspects: specifying the structure, defining a structure variable, and accessing the members of the structure.

3. A measurement example:

Let’s see how a structure can be used to group a different kind of information. If you have ever looked at an architectural drawing, you know that distances are measured in feet and inches. The length of a living room, for example, might be given as 12’-8’’, meaning 12 feet 8 inches. The hyphen is not a negative sign; it merely separates the feet from the inches. This is part of the English of measurement. The following program will show how two measurements of type distance can be added together.

Example 2:

Write C++ program to find the distance in English system.

```
#include<iostream.h>

struct distance
{
int feet;
float inches;
}
Void main ()
{
distance d1,d3;
distance d2={11,6.25};
cout<<”\n Enter feet:”;
cin>>d1.feet;
cout<<”\n Enter inches:”;
cin>>d1.inches;
d3.inches=d1.inches+d2.inches;
d3.feet=0;
If (d3.inches >=12.0)
{
```

```
d3.inches -=12.0;
d3.feet ++;
}
d3.feet +=d1.feet + d2.feet;
cout<<d1.feet<<”\’-“<<d1.inches<<”\”+”;
cout<<d2.feet<<”\’-“<<d2.inches<<”\”=”;
cout<<d3.feet<<”\’-“<<d1.inches<<”\”\n”;
}
```

4. Structures within Structures:

You can nest structures within other structures. Here’s a variation on the English system program that shows how this looks. In the bellow program we want to create a data structure that stores the dimensions of a typical room: its length and width. Since we’re working with English distances, we’ll use two variables of type distance as the length and width variables.

Example 3:

Write C++ program to find the area of the room in English system.

```
#include<iostream.h>

struct distance
{
int feet;
float inches;
}

struct room
{
distance length;
distance width;
};

Void main ()
{
room dining;
dining.length.feet=13;
dining.length.inches=6.5;
dining.width.feet=10;
dining.width.inches=0.0;
float L=dining.length.feet+dining.length.inches/12;
```