هياكل البيانات

المرحلة الثانية

محاضرة(7)

م.م فرح معاذ جاسم

Farahmaath86@uoanbar.edu.iq

1. POINTER

Pointer is a variable in C++ that holds the address of another variable. They have data type just like variables, for example an integer type pointer can hold the address of an integer variable and an character type pointer can hold the address of char variable.

Syntax of pointer

data_type *pointer_name;

```
int    *ip;    // pointer to an integer
double *dp;    // pointer to a double
float  *fp;    // pointer to a float
char   *ch     // pointer to character
```

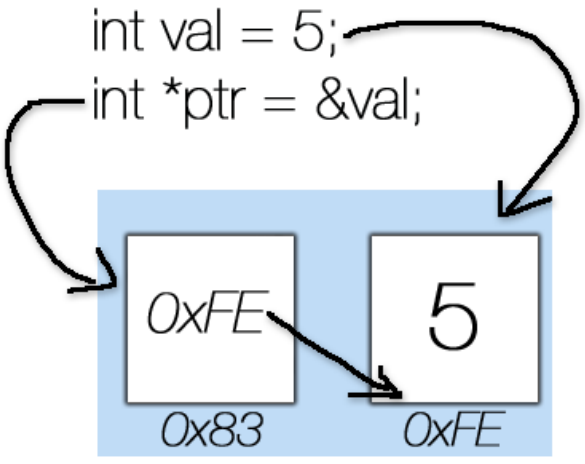
To assign the address of variable to pointer we use ampersand symbol (&).

```
int  *ptr,val;

val=5;

ptr=&val;
```

```
cout<<"Address of val: "<<&val<<endl;
cout<<"Address of val: "<<ptr<<endl;
cout<<"Address of ptr: "<<&ptr<<endl;
cout<<"Value of val: "<<*ptr;
```



2. STRUCTURE

Structure is a group of data elements grouped together under one name. These data elements, known as members, can have different types and different lengths. Data structures can be declared in C++ using the following syntax:

```
struct type_name
{
member_type1 member_name1;
member_type2 member_name2;
member_type3 member_name3;
.
.
}object_names;
```

Where type_name is a name for the structure type, object_name can be a set of valid identifiers for objects that have the type of this structure. Within braces {}, there is a list with the data members, each one is specified with a type and a valid identifier as its name.

For example:

```
struct product
{
  int weight;
  double price;
} ;

product apple;
product banana, melon;
```

or

```
struct product {
  int weight;
  double price;
} apple, banana, melon;
```

Once the three objects of a determined structure type are declared (`apple`, `banana`, and `melon`) its members can be accessed directly. The syntax for that is simply to insert a dot (`.`) between the object name and the member name. For example, we could operate with any of these elements as if they were standard variables of their respective types:

```
apple.weight
apple.price
banana.weight
banana.price
melon.weight
melon.price
```

```
main()

{
        banana.price=0.04;
        banana.weight=1;
        cout<<banana.price;
}
```

Ex2:

```
struct Person
{
   char name[50];
   int age;
   float salary;
};
 main()
{
   Person p1;

   cout << "Enter Full name: ";
   cin.get(p1.name, 50);
   cout << "Enter age: ";
   cin >> p1.age;
   cout << "Enter salary: ";
   cin >> p1.salary;
   cout << "\nDisplaying Information." << endl;
   cout << "Name: " << p1.name << endl;
   cout <<"Age: " << p1.age << endl;
   cout << "Salary: " << p1.salary;
}
```

3.POINTERS TO STRUCTURE

It's possible to create a pointer that points to a structure. It is similar to how pointers pointing to native data types like int, float, double, etc. are created. Note that a pointer in C++ will store a memory location.

```
#include <iostream>
using namespace std;
struct Length
{
        int meters;
        float centimeters;
};

main()
{
        Length *ptr, l;

        ptr = &l;

        cout << "Enter meters: ";
        cin >> (*ptr).meters;
        cout << "Enter centimeters: ";
        cin >> (*ptr).centimeters;
        cout << "Length = " << (*ptr).meters << " meters " <<
(*ptr).centimeters << " centimeters";
}
```

4.STRUCT AS FUNCTION ARGUMENT

You can pass a struct to a function as an argument. This is done in the same way as passing a normal argument. The struct variables can also be passed to a function. A good example is when you need to display the values of struct members. Let's demonstrates this:

```
struct Person
{
        int citizenship;
        int age;
};
```

```
void func(struct Person p)
{
        cout << " Person citizenship: " << p.citizenship<<endl;
        cout << " Person age: " << p.age;
}
main()
{
        struct Person p;

        p.citizenship = 1;
        p.age = 27;

        func(p);
}
```