Computer Programming

An-Najah N. University
Computer Engineering Department
Luai Malhis, Ph.D,

Structures

Definition

- Structure is a user defined data type, that is build of basic data types (char, int, double,….).
- Structure allow many variables of different types grouped together under the same name.
- To define a structure we use the following format:

```
struct name
{
 type member1;
 type member2;
…
…
};
```

Structure Definition Example

- We can Define a structure called person which is made up of a string for the name and an integer for the age and a double for salary.
- struct person

```
{
  char name[20];
  int age;
  double salary;
};
```

Defining Variable of A structure

- Previously we defined a data type called person.
- However, we must create a variable of that type to be able to use it.
- Following is sample variable declarations:

```
#include<iostream.h>
struct person{ char name[20];  int age;  double salary;}
void main() {
  person p1;          // variable of type person
  person *ptr;       // pointer of type person
  person PA[100]; // Array of persons }
```

Accessing member variables

- After defining a variable of type structure, we can access structure member variables using the '.' (call it dot) operator.
- In the previous example to access member fields of the structure we place a dot between the structure variable name (p1) and the name of a member variable (name, age or salary).

  p1.age , p1.salary, p1.name
- If the variable is a structure we use and arrow ->

  to separate pointer name and field name:

  ptr->age, ptr->salary, ptr->name;

Structure Initialization

- Like other data type we can initialize structure when we declare it. As far initialization goes structures obey the same set of rules as arrays. We initialize the fields of a structure following structure declaration with a list containing values for each filed.
- Assume we have the following structure definition:

  struct Employee{ int emp_id; char name[25]; char department[10]; float salary; };

Structure Initialization 2

We can initialize a variable of type employee when we declare it as follows

- Employee emp1 ={125,"basil","marketing",500.00};
- This initializes the emp_id field to 125, the name field to "basil", the department field to "marketing" and the salary filed to 500.0.
- We can use assignment statement to initialize one structure to another. Example:

  Employee emp2 = emp1;

Structure Initialization 3

- We can initialize a variable of type employee one field at a time using the assignment operator.

  Employee emp3.

  emp3.salary = 2470.28;

  strcpy(emp3.name, "Ahmad");

  strcpy(emp3.department,"sales");

  emp3.emp_id = 27;
- We can also use cin statement to read fields of a structure: cin >> emp3.name; cin >> emp3.emp_id;

  cin >> emp3.salary >> emp3.department;

Pointer to structures

- When declaring a pointer to structure, before we use the pointer to access member variables the pointer must be made to point to an existing structure or new structure. Example:

```
Employee emp4 ={133,"nael","sales",1234.5};
Employee *pt1, *pt2, *p3;
pt1 = &emp4;  // pt points to existing structure;
pt2 = new Employee;
pt3 = pt1;
```

Pointer to structures 2

- Then to access member variable use the "->" to separate pointer name and field name. Example

```
strcpy(p1->name,"Nader");
p1->emp_id = 1234;
strcpy(p1->department,"Human Resources");
p1->salary = 1765.5;
```

// Note: Employee *p5; p5->emp_id = 1234.6;

is invalid because p5 does not point to a an variable of type structure or to new structure.

Array of structures

- It is possible to define an array of structures. for example if we are maintaining information of all the students in some university. We need to use an array of structures to maintain information about all students.

```
struct info
{
    int id_no;              char name[20];
    char address[20];       int age;
};
```

Then, we can define an array of structure information as follows: info student[100];

Array of Structures 2

- Then can access member fields as follows:

```
student[0].id_no =  212;
strcpy(student[4].name,"walid");
int x; cin >> x; cout << student[x].age;
```

- We can also declare and array of pointers to structure as follows: info *stptr[200]; Then to we can access member variables as follows:

```
stptrp[10] = new structure;
stptr[10]->age=19; strcpy(stptr[10]->name,"ab");
```

Example:

```
#include<iostream.h >
struct info
{   int id_no;   char name[20];  char address[20];    int age;  }
void main() {
     info std[100];   int I,n;
     cout << "Enter the number of students";      cin>> n;
     cout<<  "Enter Id_no, name, address,  and age");
      for(I=0;I < n;I++) {
      cin>> std[I].id_no >> std[I].name;
       cin>> std[I].address >> std[I].age;  }
      cout <<  "Student information";
     for (I=0;I< n;I++) {
         cout << std[I].id_no << "   "  << std[I].name << "  ";
          cout <<  std[I].address;  << "  " << std[I].age <<    endl;
}
```

Nested Structure

- A structure may be defined as a member of another structure. In such structures the declaration of the embedded structure must appear before the declarations of other structures. Example

```
struct date {   int day;     int month;     int year; };
struct info
{
   int id_no;                  char name[20];
   char address[20];       int age;
   date dob;
};
```

the structure student constrains another structure date as its one of its members.

Accessing member variable of Nested structure

- Given: info st1; info *ptr; Then

```
strcpy(st1.name,"Adel");
st1.id_no = 223344; ……..
st1.dob.day = 12;  st1.dob.month = 9;
st1.year = 1998;
ptr = &st1;   // make sure ptr point to structure
cout << ptr->name;   cout << pt->id_no;
cout << ptr->dob.day  << ptr->dob.month;
```

Pointer inside a structure

- A structure may contain a pointer to a variable to another structure. Programmers must be carful to allocate memory to these pointers before accessing them. In this case we must allocate memory to each structure then use the -> to access each member. Example:

```
struct data { int age; char *name; };  data ex; Then:
ex.age = 21;       ex.name = new char[30];
strcpy(ex.name,"abc");
data * ptr;  ptr = new data; ptr->name = new char[20];
ptr->age = 22;  strcpy(ptr->name,"abc');
```

Examples

- Given the following definition:
- struct info

```
{
      int id_no;
      char name[20];
      char address[20];
      int age;
};
```

Example 1

- Write code to declare a variable of type info call it s1 and give it the following values 123 for id , “wael” for name , “jenin” for address and 21 for age.
- Solution1: info s1;

```
s1.id_no = 123;     strcpy(s1.name,”wael”);
s1.age = 21; strcpy(s1.address,”jenin”);
```

- Solution 2:

```
info s1 = {123,”wael”,”jenin”,21};
```

Example 2

- Define another variable of type info call it s2 and read information of s2 from the keyboard.
- Solution: info s2;

```
cin >> s2.id_no;   cin >> s2.name;
cin >> s2.address; cin >> s2.age;
```

- Print the content of s2 to screen.

```
cout << s2.id_no;    cout << s2.name;
cout << s2.address;  cout << s2.age;
```

Example 3

- Define a pointer to structure and make it point to s1. Then use the pointer to print s2.

```
info *ptr = &s2;
cout << ptr->id_no;    cout << ptr->name;
cout << ptr->address;  cout << ptr->age;
```

- Define a pointer to info and call it ptr; allocate new memoy to ptr and read info from the K.B.

```
info *ptr = new info;
 cin >> ptr->.id_no;   cin >> ptr->.name;
cin >> ptr->address; cin >> ptr->.age;
```

Example 4

- Declares an array called A of info of size 1000 then read the structure content from the K.B.

```
info A[1000];  int n;  cin >> n;
for (int i = 0; I < n; i++) {
cin >> A[i].id_no;   cin >> A[i].name;
cin >> A[i].address; cin >> A[i].age; }
```

- Suppose we have info *ptr = A; then

```
 for (int i = 0; I < n; i++, ptr++) {
cout << ptr->id_no;    cout << ptr->name;
cout << ptr->address;  cout << ptr->age;}
```

Passing structure to functions

- We can pass structures as arguments to a functions, and retrun structures from functions.
- A structure may be passed into a function as pass by value, reference and by address.
- You can also return a structure from a function or apointer to dynamically allocated structure in the function.
- A program example is to display the contents of a structure passing the individual elements to a function is shown next.

- # include < iostream.h >
- struct Employee{   int emp_id;   char name[25]; char department[10];   float salary;   };

-
  ```
  Employee readEmp() { Employee emp1;
     cin >> emp1.name;  cin >> emp1. emp_id;
      cin >> emp1.department; cin >> emp1.salary;
  return(emp1);   }
  ```

-
  ```
  void printEmp( Employee e )
  {     cout << e.name;  cout << e.emp_id;
         cout << e.department;  cout << e.salary;  }
  ```
- void main() {Employee em1;   em1 = reademp(); printEmp(); }

Example

- Write a function that takes and array of **info** as parameter and array size; in your function return a stucture of the oldest students.

```
info largSt(info A[], int size)
{
       info st;  st = A[0];
       for (int i =1; i <  size; i++)
               if (st.age < A[i].age) st =  A[i];
        return(st);
 }
```

Other Examples

- Given an array of structure info call it A, size 100
  Write code to do the following
  (1) Print the the count of students that start with the letter ‘A’ or ‘a’, Start and end with same letter.
  the count of names larger that 10 characters.
  print the name of students that are from Jenin.
  (2) sort the array A in according the name in ascending order. Then compute total salaries.
  (3) Sort the array A in descending order according to salary. Print employee name of the largest salary.

p1

```
int counta =0, countsl =0, countg10 =0, jen =0;
for (int i = 0; i < 100; i++)
{
   if (A[i].name[0] == ‘A’ && A[i].name[0] == ‘a’ )
      counta++;
  if(A[i].name[0] == name[strlen(A[i].name)-1]))
     countsl++;
  if(strlen(A[i].name) > 10)  countg10++;
  if (strcmp(A[i].name,”Jenin”) jen++;
}
```

p2

```
Info temp;
for (int i =0; i < 99; i++)
      for (int j = i+1; j < 100; j++)
          if(strcmp(A[i].name,A[j].name) > 0){
               temp = A[i];
              A[i] = A[j];
              A[j] = temp;
          }
// compute total salaries.
int sum =0;   for (int I =0; I <100;i++) sum+= A[i].salary;
```

p3

```
info temp;
for (int i =0; i < 99; i++)
      for (int j = i+1; j < 100; j++)
          if(A[i].salary < A[j].salary)
          {
               temp = A[i];
              A[i] = A[j];
              A[j] = temp;
          }
cout << “the highest salary is “ << A[i].name <<
“with salary” <<  A[i].salary.
```