



**Computer Programming (b) - E1124**

**(Spring 2021-2022)**

**Lecture 7**

**Records (structs)**

**INSTRUCTOR**

**Dr / Ayman Soliman**



## ➤ Contents

- Objectives
- Records (structs)
- Accessing struct Members
- Assignment
- struct Variables and Functions
- Arrays versus structs
- structs within a struct



## ➤ Objectives

- Learn about records (structs)
- Examine various operations on a struct
- Explore ways to manipulate data using a struct
- Learn about the relationship between a struct and functions
- Discover how arrays are used in a struct
- Learn how to create an array of struct items

## ➤ Records (structs)

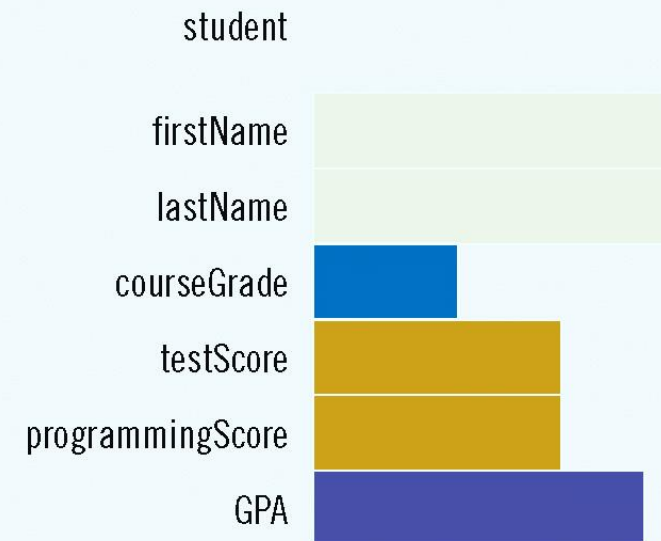
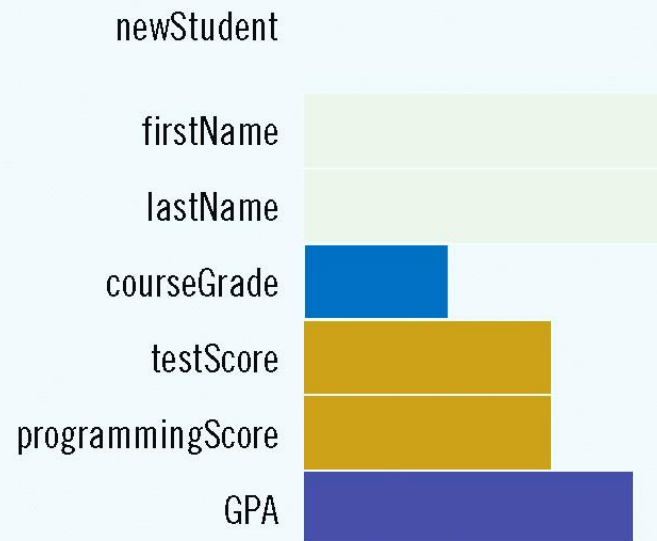
- **struct**: collection of a fixed number of components (members), accessed by name
- A struct is a **definition**, not a declaration
- Members may be of different types
- Syntax:

```
struct structName
{
    dataType1 identifier1;
    dataType2 identifier2;
    .
    .
    .
    dataTypen identifiern;
};
```

## ➤ Records (structs)

```
struct studentType
{
    string firstName;
    string lastName;
    char courseGrade;
    int testScore;
    int programmingScore;
    double GPA;
};

//variable declaration
studentType newStudent;
studentType student;
```



## ➤ Accessing struct Members

- The syntax for accessing a struct member is:

```
structVariableName.memberName
```

- The **dot (.)** is an operator, called the **member access operator**

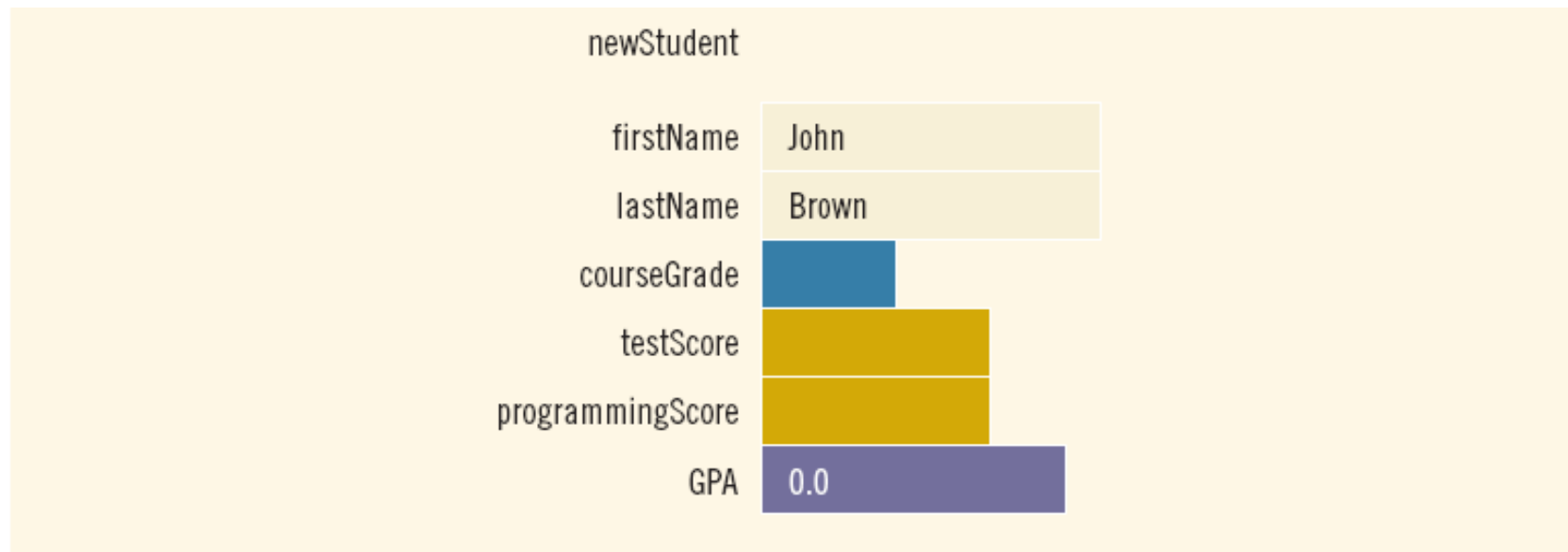
## ➤ Accessing struct Members (cont.)

- To initialize the members of newStudent:

```
newStudent.GPA = 0.0;
```

```
newStudent.firstName = "John";
```

```
newStudent.lastName = "Brown";
```



## ➤ Accessing struct Members (cont.)

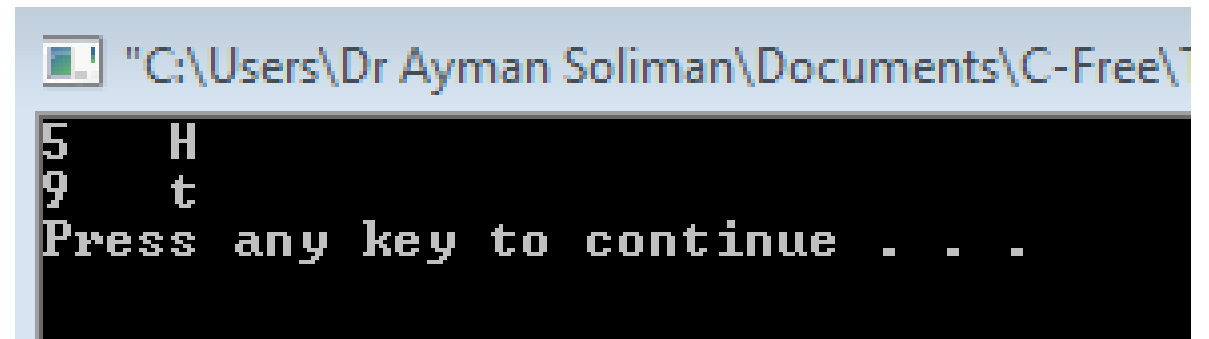
### ➤ More examples:

```
cin >> newStudent.firstName;
cin>>newStudent.testScore>>newStudent.programmingScore;
score = (newStudent.testScore + newStudent.programmingScore) / 2;
if (score >= 90)
    newStudent.courseGrade = 'A';
else if (score >= 80)
    newStudent.courseGrade = 'B';
else if (score >= 70)
    newStudent.courseGrade = 'C';
else if (score >= 60)
    newStudent.courseGrade = 'D';
else
    newStudent.courseGrade = 'F';
```



## ➤ Example 1

```
1 #include <iostream.h>
2 int main()
3 {
4     struct A
5     {
6         int x;
7         char c;
8     };
9     A var1, var2;
10    var1.x=5;
11    var2.x= var1.x+4;
12    var1.c='H';
13    var2.c='t';
14    cout<<var1.x<<"    "<<var1.c<<endl;
15    cout<<var2.x<<"    "<<var2.c<<endl;
16    return 0;
17 }
```



```
"C:\Users\Dr Ayman Soliman\Documents\C-Free\T
5 H
9 t
Press any key to continue . . .
```

## ➤ **Assignment**

➤ Value of one struct variable can be assigned to another struct variable of the same type using an assignment statement

➤ The statement:

`student = newStudent;`

➤ copies the contents of newStudent into student

## ➤ Assignment (cont.)

- The assignment statement:

**student = newStudent;**

- is equivalent to the following statements:

```
student.firstName = newStudent.firstName;
```

```
student.lastName = newStudent.lastName;
```

```
student.courseGrade = newStudent.courseGrade;
```

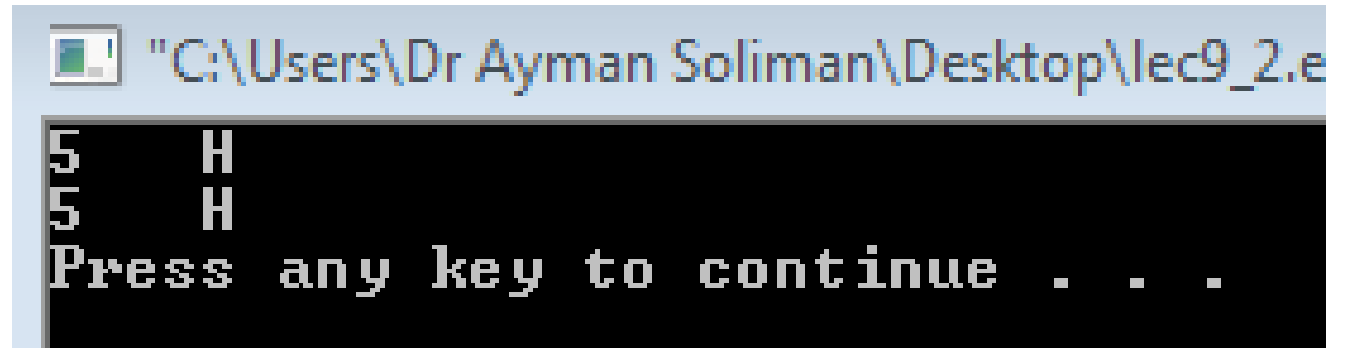
```
student.testScore = newStudent.testScore;
```

```
student.programmingScore = newStudent.programmingScore;
```

```
student.GPA = newStudent.GPA;
```

## ➤ Example 2

```
1 #include <iostream.h>
2 int main()
3 {
4     struct A
5     {
6         int x;
7         char c;
8     };
9     A var1, var2;
10    var1.x=5;
11    var1.c='H';
12    var2=var1;
13    cout<<var1.x<<"    "<<var1.c<<endl;
14    cout<<var2.x<<"    "<<var2.c<<endl;
15    return 0;
16 }
```



```
"C:\Users\Dr Ayman Soliman\Desktop\lec9_2.e
5    H
5    H
Press any key to continue . . .
```

## ➤ **Comparison (Relational Operators)**

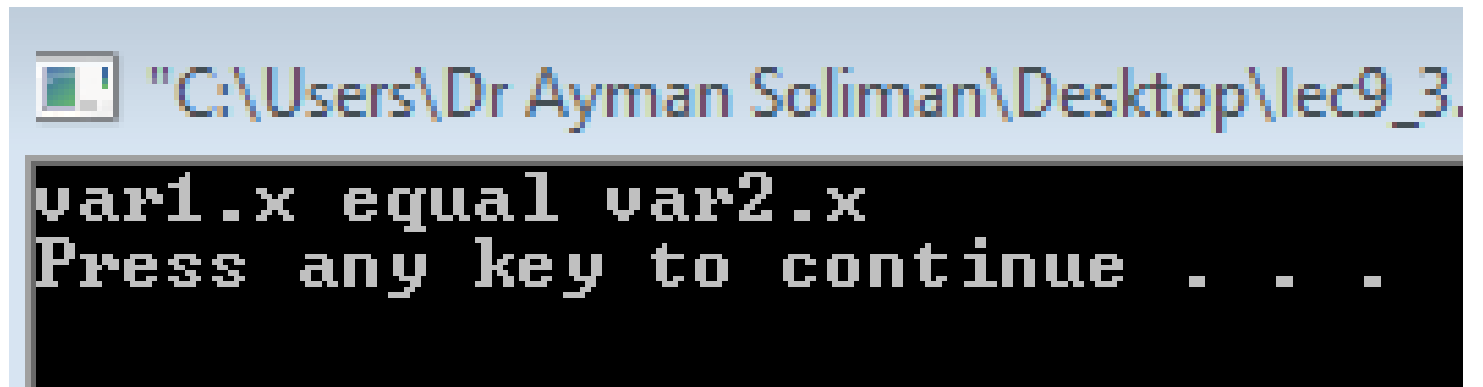
➤ Compare struct variables member-wise

➤ To compare the values of var1 and var2:

*if* ( $Var1.x == Var2.x$ )

## ➤ Example 3

```
1 #include <iostream.h>
2 int main()
3 {
4     struct A
5     {
6         int x;
7         char c;
8     };
9     A var1, var2;
10    var1.x=5;
11    var2.x= 5;
12    var1.c='H';
13    var2.c='t';
14    if (var1.x==var2.x)
15        cout<<"var1.x"<<" equal "<<"var2.x"<<endl;
16    else
17        cout<<"not equal"<<endl;
18    return 0;
19 }
```



```
"C:\Users\Dr Ayman Soliman\Desktop\lec9_3...
var1.x equal var2.x
Press any key to continue . . .
```

## ➤ **Input/output**

- No aggregate input/output operations on a struct variable
- Data in a struct variable must be read one member at a time
- The contents of a struct variable must be written one member at a time

```
cout << newStudent.firstName << " " << newStudent.lastName  
    << " " << newStudent.courseGrade  
    << " " << newStudent.testScore  
    << " " << newStudent.programmingScore  
    << " " << newStudent.GPA << endl;
```

## ➤ Struct Variables and Functions

- A struct variable can be passed as a parameter by value or by reference

```
void printStudent(studentType student)
{
    cout << student.firstName << " " << student.lastName
        << " " << student.courseGrade
        << " " << student.testScore
        << " " << student.programmingScore
        << " " << student.GPA << endl;
}
```

- A **function** can return a value of **type struct**



## ➤ Arrays versus structs

Aggregate Operation	Array	struct
Arithmetic	No	No
Assignment	No	Yes
Input/output	No (except strings)	No
Comparison	No	No
Parameter passing	By reference only	By value or by reference
Function returning a value	No	Yes

## ➤ Arrays in structs

- Two key items are associated with a list:

Values (elements)

Length of the list

- Define a struct containing both items:

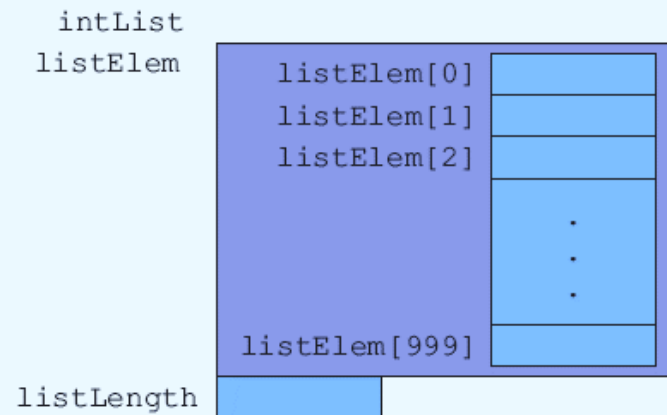
```
const int ARRAY_SIZE = 1000;

struct listType
{
    int listElem[ARRAY_SIZE];    //array containing the list
    int listLength;             //length of the list
};
```

## ➤ Arrays in structs

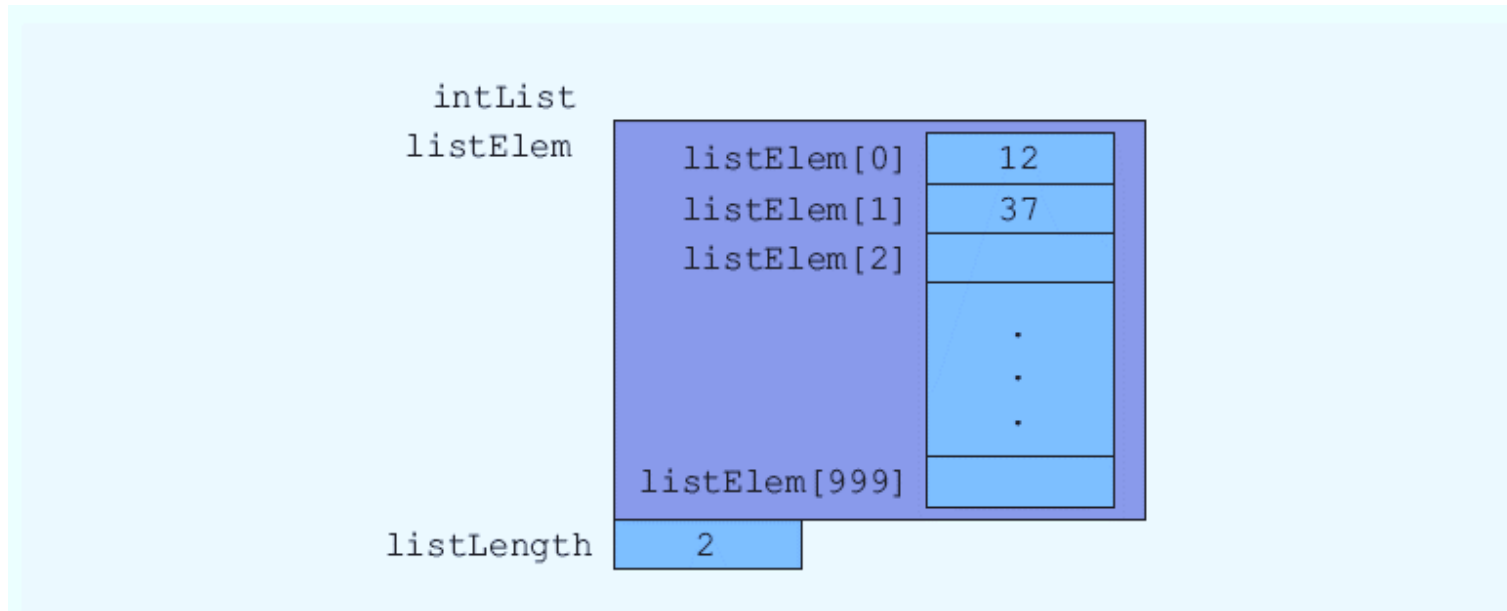
```
listType intList;
```

```
const int ARRAY_SIZE = 1000;  
  
struct listType  
{  
    int listElem[ARRAY_SIZE];    //array containing the list  
    int listLength;              //length of the list  
};
```



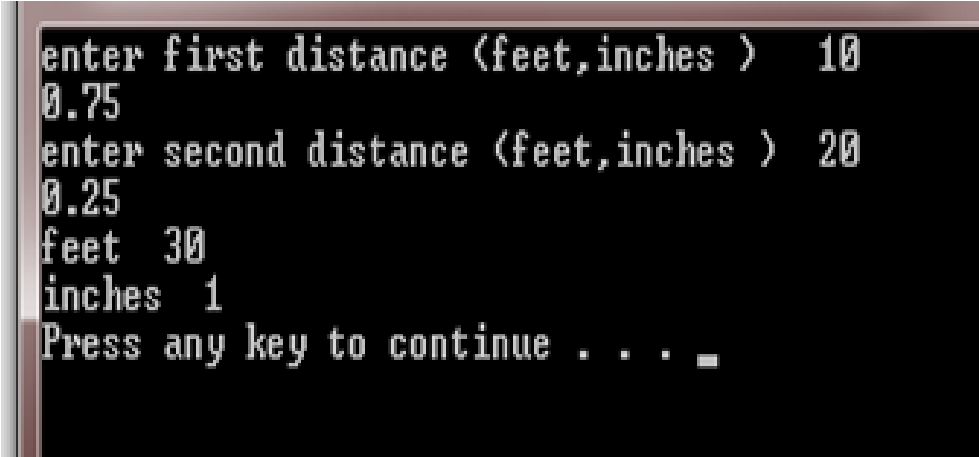
## ➤ Arrays in structs

```
intList.listLength = 0;           //Line 1
intList.listElem[0] = 12;        //Line 2
intList.listLength++;           //Line 3
intList.listElem[1] = 37;        //Line 4
intList.listLength++;           //Line 5
```



## ➤ Example 4

```
1 #include <iostream.h>
2
3 struct distance_type
4 {
5     int feet;
6     float inches;
7 };
8
9 distance_type add_distance(distance_type d1,distance_type d2);
10
11 int main ()
12 {
13     distance_type x,y ,z;
14     cout <<"enter first distance (feet,inches )  ";
15     cin >> x.feet>>x.inches;
16
17     cout <<"enter second distance (feet,inches )  ";
18     cin >> y.feet>>y.inches;
19
20     //cout << add_distance(x,y);
21     z= add_distance(x,y);
22     cout << "feet  "<<z.feet <<endl<<"inches  "<< z.inches<<endl;
23 }
24
25 distance_type add_distance(distance_type d1,distance_type d2)
26 {
27     distance_type result;
28     result.feet=d1.feet+d2.feet;
29     result.inches=d1.inches+d2.inches;
30     return result;
31 }
```



```
enter first distance (feet,inches )  10
0.75
enter second distance (feet,inches )  20
0.25
feet 30
inches 1
Press any key to continue . . . _
```

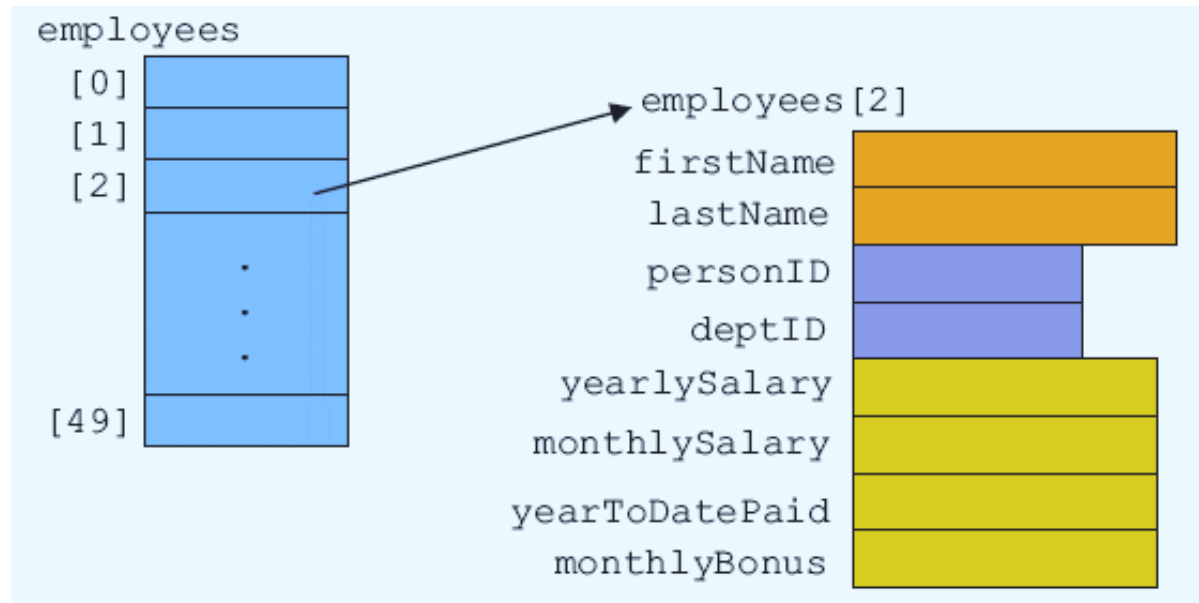
## ➤ Struct in Array

```
1 // use struct as data type of array//
2
3 #include <iostream.h>
4 #include <string>
5 using namespace std;
6
7 struct employee
8 {
9     int emp_no;
10    string fname;
11    string lname;
12    float salary;
13    float bonus;
14 };
15
16 int main ()
17 {
18     employee e[2];
19     for (int i=0;i<2;i++)
20     {
21         cout << "enter data of employee no "<<i+1<<endl;
22         cin >> e[i].emp_no>>e[i].fname>>e[i].lname>>e[i].salary>>e[i].bonus;
23     }
24
25     for (int i=0;i<2;i++)
26     {
27         cout << " employee no "<<i+1<< " ";
28         cout << e[i].emp_no<< " "<<e[i].fname<< " "<<e[i].lname<< " "<<e[i].salary<< " "<<e[i].bonus<<endl;
29     }
30
31 }
```

```
enter data of employee no 1
10
Ahmed
Ali
2000
500
enter data of employee no 2
11
Mona
Ahmed
1000
500
employee no 1 10 Ahmed Ali 2000 500
employee no 2 11 Mona Ahmed 1000 500
Press any key to continue . . .
```

## ➤ Example 5

```
employeeType employees[50];
```



```
struct employeeType  
{  
    string firstName;  
    string lastName;  
    int    personID;  
    string deptID;  
    double yearlySalary;  
    double monthlySalary;  
    double yearToDatePaid;  
    double monthlyBonus;  
};
```



## Example 5

```
ifstream infile; //input stream variable
                //assume that the file employee.dat has been opened
for (counter = 0; counter < 50; counter++)
{
    infile >> employees[counter].firstName
            >> employees[counter].lastName
            >> employees[counter].personID
            >> employees[counter].deptID
            >> employees[counter].yearlySalary;
    employees[counter].monthlySalary =
        employees[counter].yearlySalary / 12;
    employees[counter].yearToDatePaid = 0.0;
    employees[counter].monthlyBonus = 0.0;
}

double payCheck; //variable to calculate the paycheck

for (counter = 0; counter < 50; counter++)
{
    cout << employees[counter].firstName << " "
         << employees[counter].lastName << " ";

    payCheck = employees[counter].monthlySalary +
               employees[counter].monthlyBonus;

    employees[counter].yearToDatePaid =
        employees[counter].yearToDatePaid +
        payCheck;

    cout << setprecision(2) << payCheck << endl;
}
```



## ➤ structs within a struct

```
struct employeeType
{
    string firstname;
    string middlename;
    string lastname;
    string empID;
    string address1;
    string address2;
    string city;
    string state;
    string zip;
    int hiremonth;
    int hireday;
    int hireyear;
    int quitmonth;
    int quitday;
    int quityear;
    string phone;
    string cellphone;
    string fax;
    string pager;
    string email;
    string deptID;
    double salary;
};
```

versus

```
struct addressType
{
    string address1;
    string address2;
    string city;
    string state;
    string zip;
};

struct dateType
{
    int month;
    int day;
    int year;
};

struct contactType
{
    string phone;
    string cellphone;
    string fax;
    string pager;
    string email;
};

struct nameType
{
    string first;
    string middle;
    string last;
};

struct employeeType
{
    nameType name;
    string empID;
    addressType address;
    dateType hireDate;
    dateType quitDate;
    contactType contact;
    string deptID;
    double salary;
};
```

```

struct nameType
{
    string first;
    string middle;
    string last;
};

struct addressType
{
    string address1;
    string address2;
    string city;
    string state;
    string zip;
};

struct dateType
{
    int month;
    int day;
    int year;
};

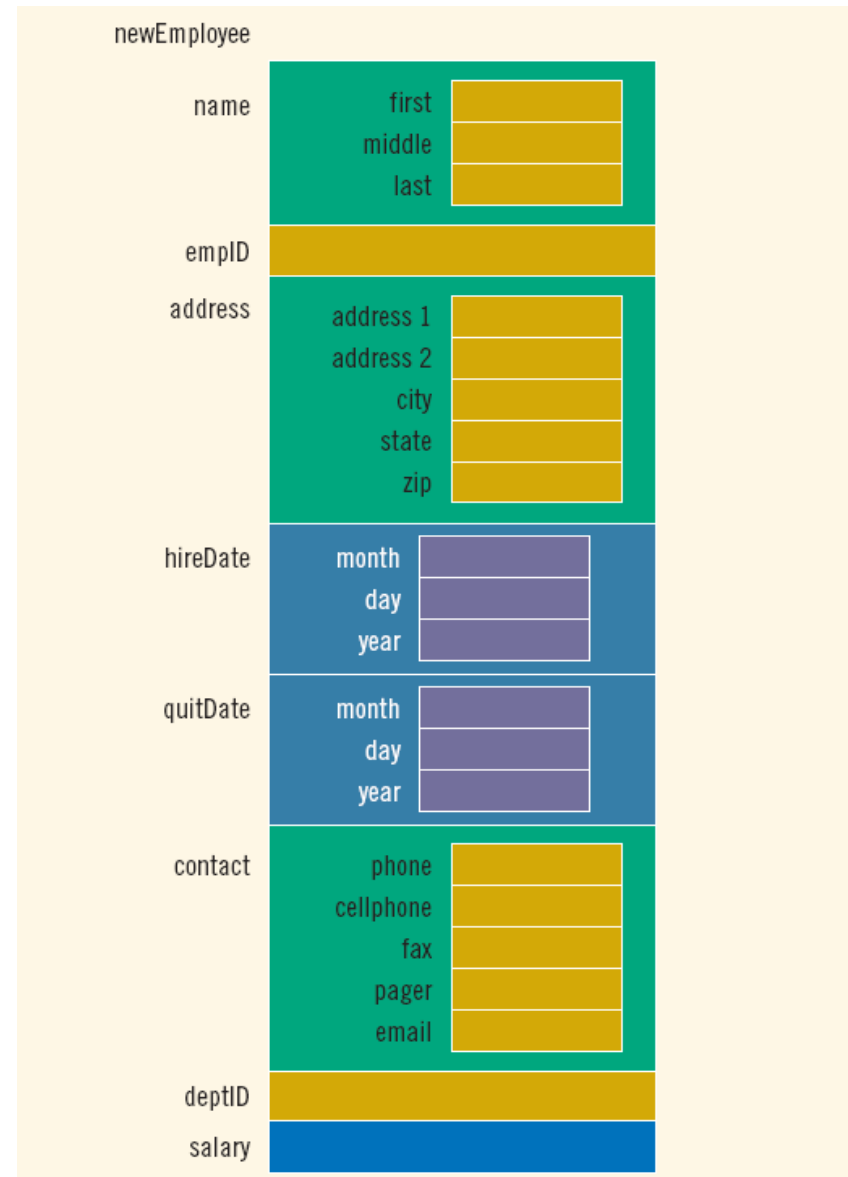
struct contactType
{
    string phone;
    string cellphone;
    string fax;
    string pager;
    string email;
};

struct employeeType
{
    nameType name;
    string empID;
    addressType address;
    dateType hireDate;
    dateType quitDate;
    contactType contact;
    string deptID;
    double salary;
};

```

## ➤ Memory locations

```
employeeType newEmployee;
```



Thank  
you

