

Data Structure

ENG. LAMYA GABER

Structure

- Structures are used to collect related variables in one place.
- This means that all information items are treated as a single entity called a **structure**
- This will simplify any further information over the collected data.
- To define structure, you should use **struct** keyword inform that all data types between the next braces {...} are members of this new data type.
- You can define as many data types inside the structure.

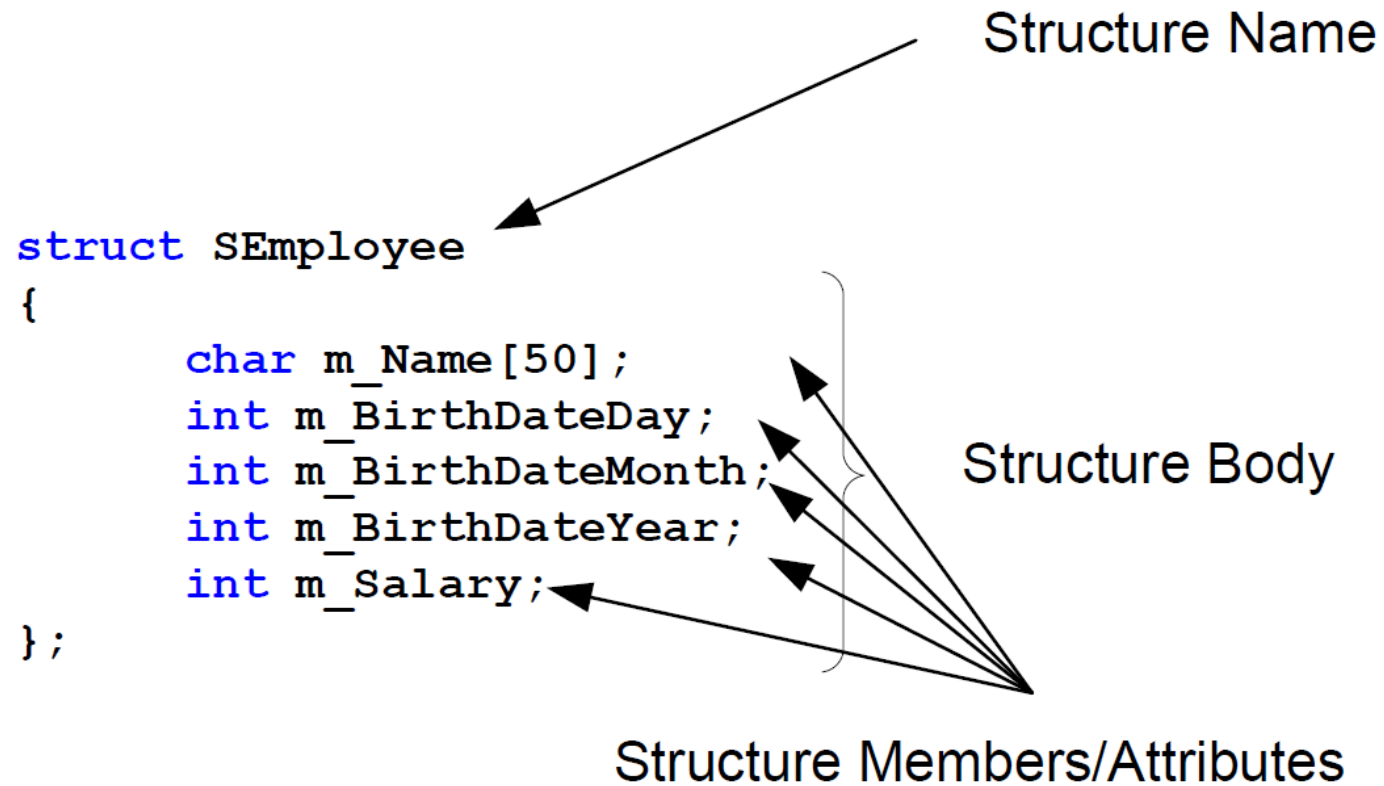
Define a structure

Structure Name

```
struct SEmployee
{
    char m_Name[50];
    int m_BirthDateDay;
    int m_BirthDateMonth;
    int m_BirthDateYear;
    int m_Salary;
};
```

Structure Body

Structure Members/Attributes



Define a structure

- **Structure** is a complex data type consisting of a set of members/attributes.
- Each attribute can have any data type. It can be int, float, short ...
- It can be single value or array.
- It can be const or variable.
- Also, structures may hold another structure (nested structure).

```
#include <iostream>
#include <string>
using namespace std;

struct SEmployee{
    string name;
    int birth_day;
    int birth_month;
    int birth_year;
};

int main(){
    SEmployee X;

    X.name = "Ahmed Mohammed";
    X.birth_day = 10;
    X.birth_month = 3;
    X.birth_year = 1996;

    cout << X.name << " is born in " << X.birth_day << "/" << X.birth_month << "/" << X.birth_year << endl;

    return 0;
}
```

Copying Structure Variable Contents

- In structures, if you need to copy structure to another structure. you don't have to copy each member/attribute in a structure.
- Alternatively, X variable is assigned to Z by direct copying using equal operator.
- By comparing it with arrays, this operation in structure will be easier and will reduce the code size.

```
#include <iostream>
#include <string>
using namespace std;

struct SEmployee{
    string name;
    int birth_day;
    int birth_month;
    int birth_year;
};

int main(){
    SEmployee X,Y;

    X.name = "Ahmed Mohammed";
    X.birth_day = 10;
    X.birth_month = 3;
    X.birth_year = 1996;

    Y = X;

    cout << Y.name << " is born in " << Y.birth_day << "/" << Y.birth_month << "/" << Y.birth_year << endl;

    return 0;
}
```

Initializing Structure Variables

```
int main(){
    SEmployee X = { "Ahmed Ali", 12, 5, 1994, 6000 };

    cout << "Name: " << X.name << endl;
    cout << "Birth date: " << X.birth_day << "/" << X.birth_month << "/" << X.birth_year << endl;
    cout << "Salary: " << X.salary << endl;

    return 0;
}
```

Nested Structure Definition

- You can define members with any data type even other structure type.

```
struct SEmployee{  
    string name;  
  
    int birth_day;  
    int birth_month;  
    int birth_year;  
  
    int graduation_day;  
    int graduation_month;  
    int graduation_year;  
  
    int salary;  
};
```

To increase readability and
reduce complexity



```
struct SDate{  
    int day;  
    int month;  
    int year;  
};  
  
struct SEmployee{  
    string name;  
  
    SDate birth_date;  
    SDate graduation_date;  
  
    int salary;  
};
```

```
#include <iostream>
#include <string>
using namespace std;

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

struct SEmployee{
    string name;

    SDate birth_date;
    SDate graduation_date;

    int salary;
};

int main(){
    SEmployee X = { "Ahmed Ali", { 12, 5, 1992 }, { 2, 4, 2014}, 6000 };

    cout << "Name: " << X.name << endl;
    cout << "Birth date: " << X.birth_date.day << "/" << X.birth_date.month << "/" << X.birth_date.year << endl;
    cout << "Graduation date: " << X.graduation_date.day << "/" << X.graduation_date.month << "/" << X.graduation_date.year << endl;

    cout << "Salary: " << X.salary << endl;

    return 0;
}
```

Using Structures with Functions

- Functions can use structure like any normal variable.
- it can be passed as a single value or array, and also it can be returned as a result of the function.
- Following example shows how to create functions to read and print employees to make our main function like this

```
int main(){
    SEmployee X;

    // call function of reading structure
    X = ReadEmployee();

    // call functiono of printing structure
    PrintEmployee(X);

    return 0;
}
```

```
#include <iostream>
#include <string>
using namespace std;

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

struct SEmployee{
    string name;
    SDate birth_date;
    SDate graduation_date;
    int salary;
};

struct SDate ReadDate(string dataname){
    struct SDate Date;
    cout << "Enter " << dataname << "as follows (12 3 1883): ";
    cin >> Date.day >> Date.month >> Date.year;
    return Date;
}
```

```

|struct SEmployee ReadEmployee(){
    struct SEmployee X;

    cout << "Enter the name of employee: ";
    cin >> ws;
    getline(cin, X.name);

    X.birth_date = ReadDate("Birth date ");
    X.graduation_date = ReadDate("Graduation date");

    cout << "Enter salary: ";
    cin >> X.salary;

    return X;
}

|void PrintEmployee(struct SEmployee X){
    cout << "Name: " << X.name << endl;
    cout << "Birth date: " << X.birth_date.day << "/" << X.birth_date.month << "/" << X.birth_date.year << endl;
    cout << "Graduation date: " << X.graduation_date.day << "/" << X.graduation_date.month << "/" << X.graduation_date.year << endl;

    cout << "Salary: " << X.salary << endl;
}

|int main(){
    SEmployee X;
    // call function of reading structure
    X = ReadEmployee();
    // call function of printing structure
    PrintEmployee(X);
    return 0;
}

```

Thank you

