

Structures

1

Contents

- Definition
- Structures as user-defined data types
- Member access
- Member functions

2

Structure

- Think of this case:
 - You want to store some information about a person: his/her name, citizenship number and salary.
 - You can easily create different variables `name`, `citizenNo`, `salary` to store these information separately.
 - You would want to store information about multiple persons.
 - Create different variables for each information per person? i.e, `name1`, `citizenNo1`, `salary1`, `name2`, `citizenNo2`, `salary2`, ..., `nameN`, `citizenNoN`, `salaryN`.

Structure

- Think of this case:
 - You want to store some information about a person: his/her name, citizenship number and salary.
 - You can easily create different variables `name`, `citizenNo`, `salary` to store these information separately.
 - You would want to store information about multiple persons.
 - ~~Create different variables for each information per person? i.e, `name1`, `citizenNo1`, `salary1`, `name2`, `citizenNo2`, `salary2`, ..., `nameN`, `citizenNoN`, `salaryN`.~~

- **Solve with structure.**

Structure

- Structure is a **collection of variables** of different data types under a **single name**.
 - Similar (with some differences) to `class` (in object-oriented programming).

Structure Definition

- Syntax:

```
struct StructName
{
    DataType1 member1;
    DataType2 member2;
    ...
    DataTypeN memberN;
};
```

Examples

- Example 01:

```
struct Person{  
    char name[50];  
    char citizenNo[10];  
    float salary;  
};
```

Examples

- Example 02:

```
struct POINT{  
    float X;  
    float Y;  
};
```

Examples

- Example 03:

```
struct Line{  
    POINT start;  
    POINT end;  
};
```

Use of Structures

- Structures can use as user-defined **data types**.
- Examples:

```
Person person1, person2;  
POINT ptA, ptB;  
Line first_line, second_line;
```

Use of Structures

- Structure arrays: Arrays of same **structure data type**.
- Examples:

```
Person personArr[10]; //array of 10 Person elements.
```

```
POINT ptArr[30]; //array of 30 POINT elements.
```

```
Line lineArr[5]; //array of 5 Line elements.
```

Initialization

- A `struct` variable can be initialized when declaring like in the following examples.
- Examples:

```
POINT ptX = {9.3, 2.7};
```

```
Line line1 = {{2,3}, {7,2}};
```

```
Person person1 = {"Nguyen Van A", "7234", 9.5};
```

Member Access

- Members of a `struct` variable can be accessed using the dot (`.`) operator.

- Examples:

```
POINT ptX = {9.3, 2.7};  
POINT ptY;  
ptY.X = 4;  
ptY.Y = 7;  
std::cout << "Point X: " << ptX.X << " " << ptX.Y << std::endl;  
std::cout << "Point Y: " << ptY.X << " " << ptY.Y << std::endl;
```

Member Access

- Examples:

```
Line line1 = {{2,3}, {7,2}};  
std::cout << "(" << line1.start.X << "," << line1.start.Y << ")"  
    << "(" << line1.end.X << "," << line1.end.Y << ")" << std::endl;
```

Structure Size

- Use `sizeof` operator to get the size of the structure.

- Examples:

```
std::cout << "Size of Point: " << sizeof(Point) << std::endl;  
std::cout << "Size of Line: " << sizeof(Line) << std::endl;  
std::cout << "Size of Person: " << sizeof(Person) << std::endl;
```

Structure Size

- Notes:

- The size of a structure is usually a **multiple of 2, 4, ..** (depends on settings of compilers).
- `sizeof` can give different results when running programs.
- Use `#pragma pack(1)` to get correct size.

- Example:

```
struct STUDENT {  
    char studentID[7];  
    char studentName[41];  
    char DOB[11];  
    float score_01, score_02, score_03;  
};
```


Assignment Operator =

- The value of all members of a structure variable can be assigned to another structure using assignment operator =
if both structure variables are of same type.
- You don't need to manually assign each members.
- Examples:

```
POINT ptX = {10, 20}, ptY = {9, 4};  
Line line2, other_line;  
line2.start = ptX;  
line2.end = ptY;  
other_line = line2;
```

Passing Structures to Functions

- Structures can be passed to functions the same ways other data types do.
- Examples:

```
void Print(POINT p);  
void Input(PERSON &per);  
float Distance(POINT pt1, POINT pt2);  
float Distance(const Line& line);
```

Passing Structures to Functions

```
//Print the coordinate pair of point p
void Print(POINT p){
    std::cout << p.X << " " << p.Y << std::endl;
}

//Calculate the distance of line
float Distance(const Line& line){
    return sqrt(pow(line.start.X - line.end.X, 2)
               + pow(line.start.Y - line.end.Y, 2));
}
```

Member Functions

Special in C++: Member Functions

Member Functions

- In C++, functions can be put **inside** the structures and used as the structure members.
- Purposes:
 - Initialization
 - Methods

Member Functions

- Initialization (**constructors**):
 - Functions with the same Structure Name

```
struct POINT {  
    float X;  
    float Y;  
    POINT() { //constructor  
        X = 0;  
        Y = 0;  
    }  
};
```

Member Functions

- Methods:
 - The operations a structure (variable) can do.

- Examples:

```
struct POINT{  
    float X, Y;  
    POINT();  
    void Input(); //Input the member values of point  
    void Print(); //Output the point to the screen  
};
```

Member Functions

- Examples:

```
struct POINT {  
    void Input()  
    {  
        std::cout << "X: ";  
        std::cin >> X;  
        std::cout << "Y: ";  
        std::cin >> Y;  
    }  
};
```

Member Functions

- Examples:

```
struct POINT{  
    void Print() {  
        std::cout << X << " " << Y << std::endl;  
    }  
};
```

Member Functions

- Other ways of defining function members:

```
void POINT::Input() {  
    std::cout << "X: ";  
    std::cin >> X;  
    std::cout << "Y: ";  
    std::cin >> Y;  
}  
  
void POINT::Print() {  
    std::cout << X << " " << Y << std::endl;  
}
```

Member Functions

- The following example demonstrates how to use member functions of a structure variable.

```
int i;  
int n = 5;  
POINT ptArray[n]; //array of n POINT elements  
for (i = 0; i < n; i++){  
    std::cout << "*** Point " << i + 1 << endl;  
    ptArray[i].Input();  
}  
for (i = 0; i < n; i++)  
    ptArray[i].Print();
```

Questions and Answers