Structures


Canberra Institute
of Technology


aie
SPECIALIST EDUCATORS IN
GAMES, ANIMATION & FILM VFX

Contents

- What are structs?
- Struct syntax
- Accessing variables
- Motivation for using structs

Our own data structures

- Up until now, we have covered primitive C++ data types.
  - int, char, float, bool, etc
- It is very common that a group of variables are used together regularly.
- In C++, we can group data types into a structure
- Structures are built from existing basic data types

Declaration

- Declaring a struct lets you define your own variable type.

```
struct StructName
{
    int someVariable;
    char anotherVariable;
};
```

Declaration

- To define a struct you type the struct keyword.

```
struct StructName
{
    int someVariable;
    char anotherVariable;
};
```

Declaration

- After the keyword, you must give your struct declaration a unique name.

```
struct StructName
{
    int someVariable;
    char anotherVariable;
};
```

Declaration

- Then, between braces, you give a list of variable names.
  - These are the variables that will make up the struct.

```
struct StructName
{
    int someVariable;
    char anotherVariable;
};
```

Declaration

- A struct declaration must end in a semicolon.

```
struct StructName
{
    int someVariable;
    char anotherVariable;
};
```

What does declaring a struct do?

- Declaring a struct creates a new type of variable.
- You can then create variables of that type, just like you would make ints, floats, etc
- Structs group variables together. When the variable bob is declared, it actually creates two variables
  - bob.health
  - bob.mana

```
//Declares the Player variable type
struct Player
{
    int health;
    int mana;
};

void main()
{
    //Creates a variable of type Player called bob
    Player bob;

    //Creates a variable of type int called my_number
    int my_number;

    system("pause");
}
```

The Dot Operator

- We call the parts of a struct its member variables
- Member variables are accessed with the ‘.’ operator.
- Member variables can be treated as normal variables in all respects.

The Dot Operator

- You can use the member variables just like any other variable.

- Just like normal variables, if you don't give them a value, they have random data inside them.

```
struct Player
{
    int health;
    int mana;
};

void main()
{
    Player bob;
    bob.health = 100;
    bob.mana = 100;

    if(bob.health < 50)
    {
        cout << "Bob's health is low" << endl;
    }

    system("pause");
}
```

Initialization

- Initializing a struct to ‘{}’ sets all of its member variables to 0.

- Structures can be assigned to with another struct of the same type. This copies all of the variables across to the assigned variable.

```
struct Point2D
{
    float x;
    float y;
};

void main()
{
    Point2D origin = {};
    Point2D playerPosition = origin;
}
```

The Initializer List

- The initializer list works by putting values inside the ‘{}’
- You list the values you want to set in the same order the variables appear in the struct definition.
- If you don’t fill out all the variables, any that weren’t set are set to 0

```
struct Point2D
{
    float x;
    float y;
};

void main()
{
    Point2D origin = {};
    Point2D playerPosition = {5.2f, 6.94f};
}
```

Nested Structs

- Structures can be members of other structures.
- The dot operator can be chained to access nested structs

```
struct Point2D
{
    float x;
    float y;
};

struct PlayerShip
{
    Point2D position;
    Point2D velocity;
};
```

```
void main()
{
    PlayerShip ship = {{0,0}, {0,0}};
    ship.position.x = 4.5f;

}
```

Functions

- Structs can be used as function arguments and return types – again, just like regular variables.

```
struct Point2D
{
    float x;
    float y;
};

Point2D AddPoints(Point2D point_one, Point2D point_two)
{
    Point2D result = {};
    result.x = point_one.x + point_two.x;
    result.y = point_one.y + point_two.y;
    return result;
}
```

Summary

- Structures allow us to group variables together and make our code more cleaner and logical.
- You access the variables inside a struct variable using the ‘.’ operator.
- You can initialize structs with the initializer list ‘{}’
- They function just like any other primitive variable.

References


Canberra Institute
of Technology


aie
SPECIALIST EDUCATORS IN
GAMES, ANIMATION & FILM VFX