Structs and Unions

Let's learn about structs and unions in this lesson.

WE'LL COVER THE FOLLOWING

Structs

Introduction

A Structure in C++ is a group of data elements grouped together under one name. These data elements, known as **members**, can be of different types and sizes. It is a user-defined data type that allows us to combine data items of different kinds.

The scope of `Struct`

Structs are almost identical to classes. The default access specifier for a struct is `public` instead of `private`.

> The default inheritance specifier is `public` instead of `private`.

Example

Let’s consider an example of a **Person** struct which contains `age`, `size`, `weight`, and `name` as members. A struct always ends with a `;`.

```
struct Person{
int age;
int size;
int weight;
std::string name;
};
```

Structs should be used instead of classes if the data type is a simple data holder.

Unions

Introduction

A union is a special data type where all members start at the same address. A union can only hold one type at a time, therefore, we can save memory. A tagged union is a union that keeps track of its types. By using union, we are actually pointing to the same memory for the different data types used.

Rules

Unions are special class types.

- Only one member can exist at any one point in time.
- They only need as much space as the biggest member requires, which saves memory.
- The access specifier is `public` by default.
- They cannot have `virtual` methods like with Inheritance.
- They cannot have references.
- They cannot be inherited nor inherited from.

Example

Let’s take a look at an example of the union:

```
#include <iostream>

union Value {
    int i;
    double d;
};

int main(){
  Value v = { 123 };      // now v holds an int
  std::cout << v.i << '\n';    // write 123
  v.d = 987.654;          // now v holds a double
  std::cout << v.d << '\n';    // write 987.654
}
```

In this chapter, we have learned about classes, objects, and related topics. In the next chapter, we'll study inheritance in detail. Without any further ado, let's start!