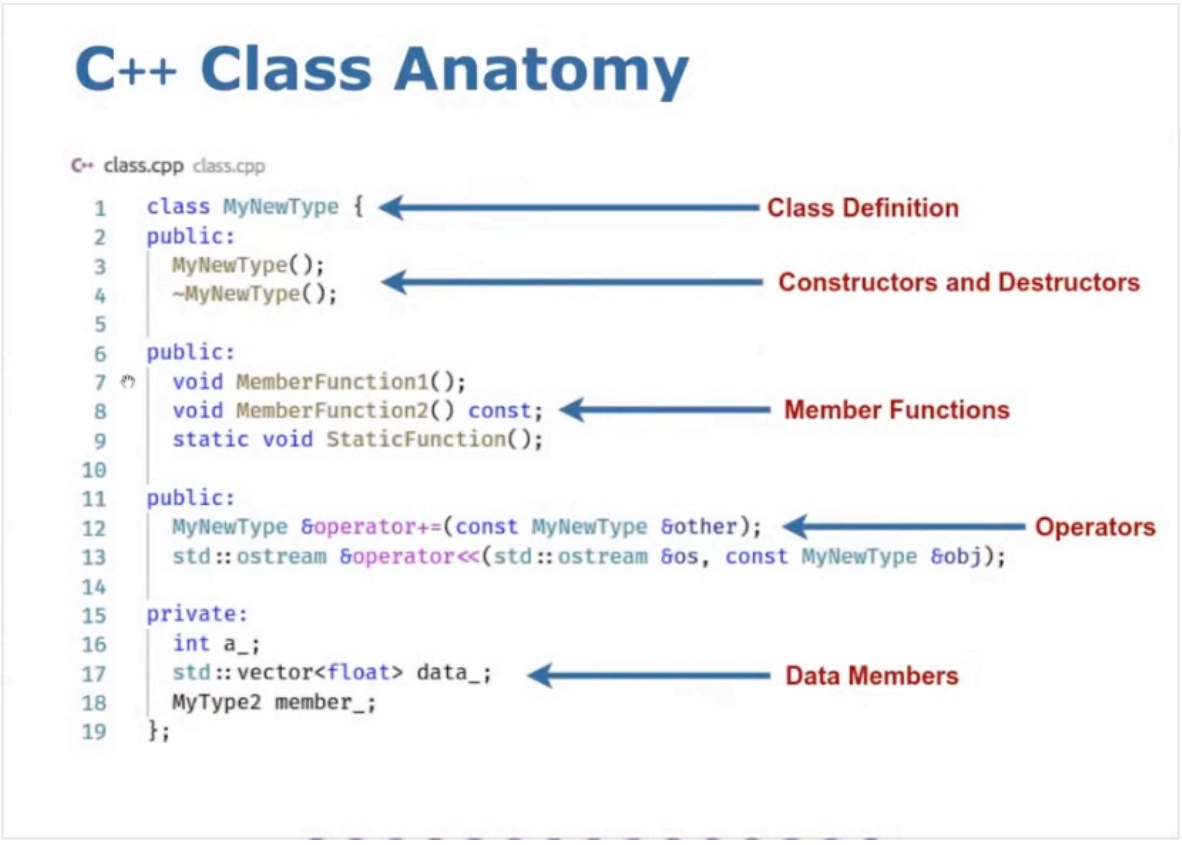
C++ Classes

- Classes are used to encapsulate data along with methods to process them.
- Every class or struct defines a new type.
- A variable of such type is an instance of class.
- Classes allow C++ to be used as an object Oriented Programming language.
- String, vector, etc. are all classes.



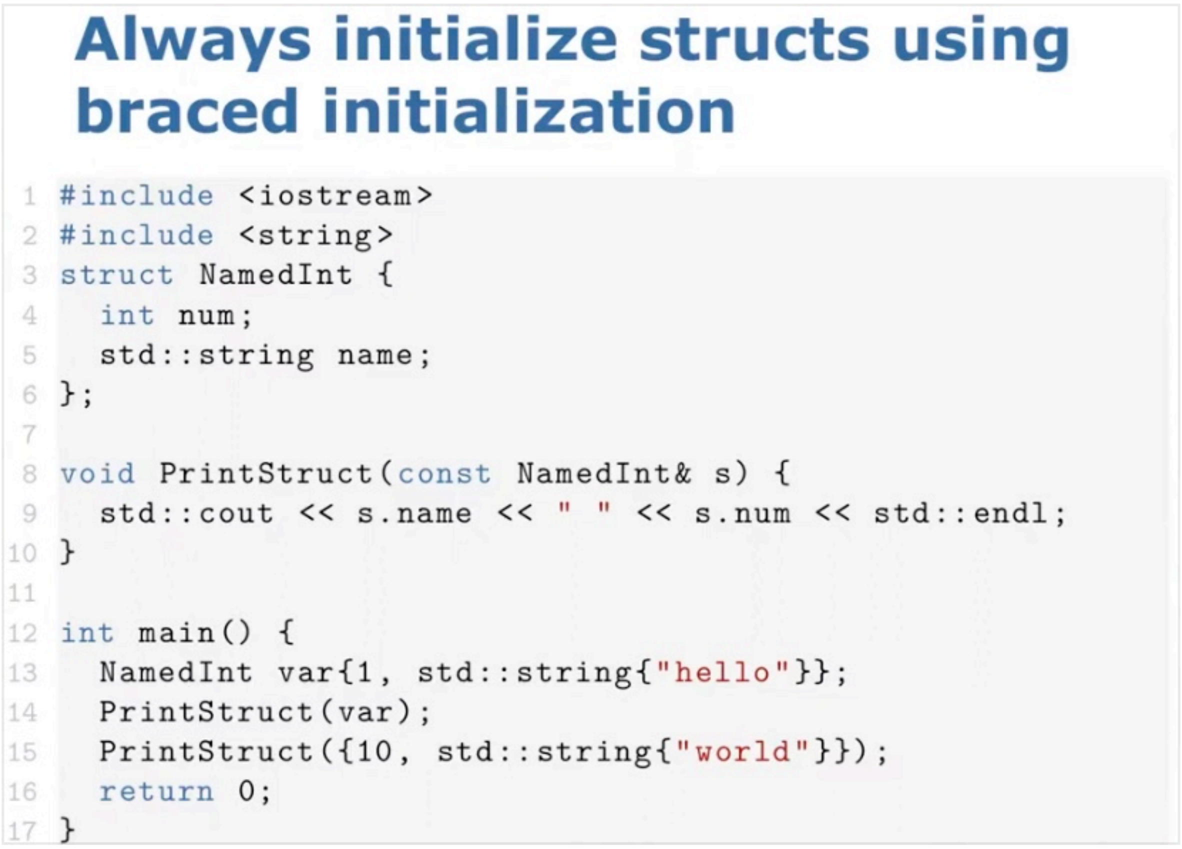
```cpp
class MyNewType {
public:
  MyNewType();
  ~MyNewType();

public:
  void MemberFunction1();
  void MemberFunction2() const;
  static void StaticFunction();

public:
  MyNewType &operator+=(const MyNewType &other);
  std::ostream &operator<<(std::ostream &os, const MyNewType &obj);

private:
  int a_;
  std::vector<float> data_;
  MyType2 member_;
};
```

Classes Syntax

- Definition starts with the keyword class.
- Classes have three access modifiers:
  - private
  - protected
  - public
- By default, everything is private.
- Access members with a "."
- Have two types of special functions:
  - Constructors: called upon creation of an instance of class
  - Destructor: called upon destruction of an instance of class

Structs

- Definition starts with the keyword struct
  struct ExampleStruct {
      Type value;
      Type value;
      Type value;
      // No functions!
  };
- Struct is a class where everything is public.
- Use struct as a simple data container, if it needs a function it should be a class instead.



Always initialize structs using braced initialization

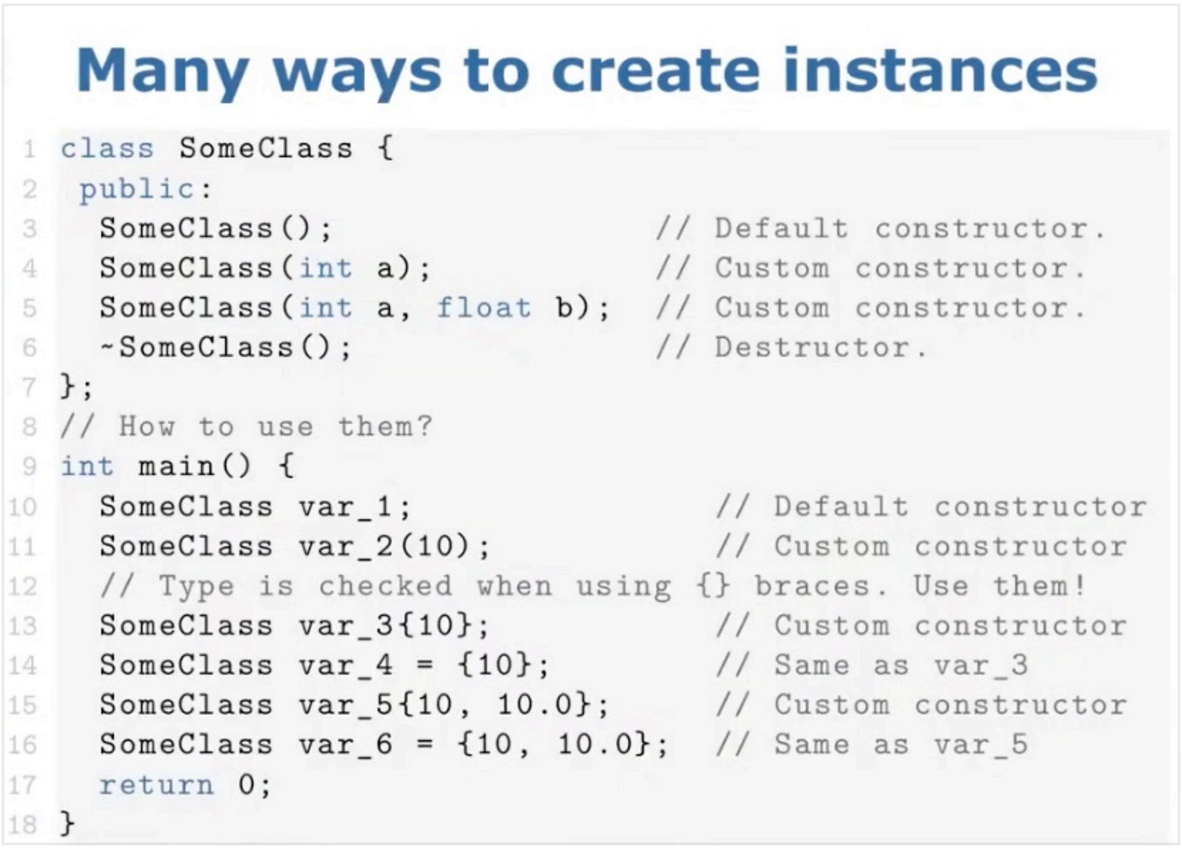
```cpp
#include <iostream>
#include <string>
struct NamedInt {
  int num;
  std::string name;
};

void PrintStruct(const NamedInt& s) {
  std::cout << s.name << " " << s.num << std::endl;
}

int main() {
  NamedInt var{1, std::string{"hello"}};
  PrintStruct(var);
  PrintStruct({10, std::string{"world"}});
  return 0;
}
```

Data stored in a class

- Classes can store data of any type.
- All data must be private.
- Use snake_case_ with a trailing “_" for private data members.
- Data should be set in Constructor.
- Cleanup data in the Destructor if needed.

Constructors and Destructors

- Classes always have at least one Constructor and exactly one Destructor.
- They don’t have any explicit return type.
- If there is no explicit constructor, an implicit default constructor will be generated.

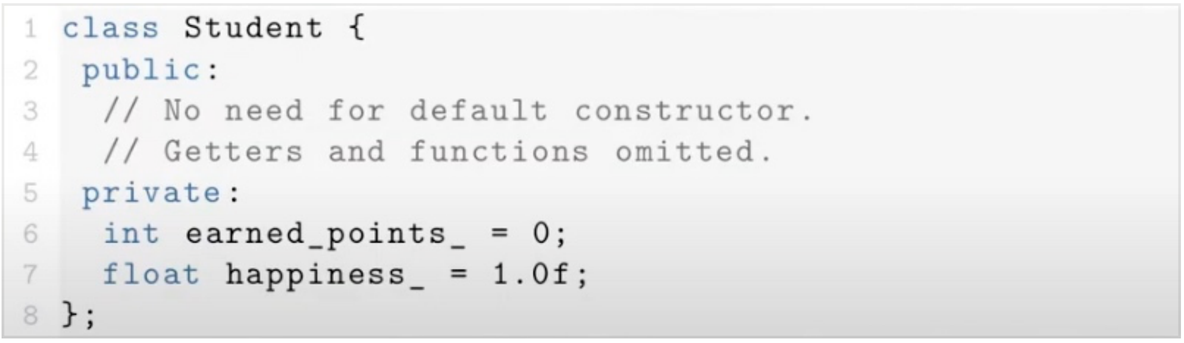


Many ways to create instances

```cpp
class SomeClass {
 public:
  SomeClass();                        // Default constructor.
  SomeClass(int a);                   // Custom constructor.
  SomeClass(int a, float b);          // Custom constructor.
  ~SomeClass();                       // Destructor.
};
// How to use them?
int main() {
  SomeClass var_1;                       // Default constructor
  SomeClass var_2(10);                   // Custom constructor
  // Type is checked when using {} braces. Use them!
  SomeClass var_3{10};                   // Custom constructor
  SomeClass var_4 = {10};                // Same as var_3
  SomeClass var_5{10, 10.0};             // Custom constructor
  SomeClass var_6 = {10, 10.0};          // Same as var_5
  return 0;
}
```

Always initialize members for classes

- C++11 allows to initialise variables in-place.
- Do not initialise them in the constructor.
- No need for explicit default constructor.



```cpp
class Student {
 public:
  // No need for default constructor.
  // Getters and functions omitted.
 private:
  int earned_points_ = 0;
  float happiness_ = 1.0f;
};
```

- Leave the members of structs uninitialised as defining them forbids using brace initialization.

Classes as Modules

- Prefer encapsulating information that belongs together into a class.
- Separate declaration and definition of the class into header and source files.