C++ 语法基础

cafébabe
南京航空航天大学
2024-09-26

代码框架 Code Template

如果你不想深究背后的原理，初学时可以直接背下来：

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

int main() {
    // your code here
    return 0;
}
```

程序的入口是 `main` 函数，`return 0;` 表示程序正常结束。

Hello, World!

```cpp
#include <iostream>

using namespace std;

int main() {
    cout << "Hello, World!\n";
    return 0;
}
```

输出：

```
Hello, World!
```

注释 Comments

```cpp
#include <iostream>

using namespace std;

int main() {
    // 这是一个单行注释
    cout << "Hello, World!\n";
    /* 这是一个注释块
    注释块可以跨越多行*/
    return 0;
}
```

注释对程序运行没有影响，可以用来解释程序的意思，还可以在让某段代码不执行（但是依然保留在源文件里）。

输入&输出 Input & Output

```cpp
#include <iostream>
#include <string> // 包含 string 类型

using namespace std;

int main() {
    string name;
    cout << "What's your name? ";
    cin >> name;
    cout << "Hello, " << name << "!\n";
    return 0;
}
```

输出：

```
What's your name? [AtomFirst]
Hello, AtomFirst!
```

`string name;` 是什么?

这是一个变量声明语句。

`string` 是 C++ 中的一种数据类型，表示字符串。

`name` 是一个变量名，用来存储用户输入的字符串。

`cin >> name;` 用来从标准输入读取一个字符串，存储到 `name` 中。

`cout << "Hello, " << name << "!\n";` 用来输出字符串。

数据类型 Data Types

这些是 C++ 中常见的数据类型：

- `int` 整数类型
- `double` 浮点数类型
- `char` 字符类型
- `string` 字符串类型
- `bool` 布尔类型
- `long long` 长整数类型

使用 `string` 类型需要包含头文件 `string`。

声明一个变量 Set a Variable

声明一个变量的语法是 `数据类型 变量名;`

赋值一个变量的语法是 `变量名 = 值;`

```
int main() {
    int a; // 声明一个整数变量 a
    a = 114514; // 给 a 赋值为 114514
    cout << a << "\n"; // 输出 a 的值和一个换行符
    return 0;
}
```

```
114514
```

在代码中，像 `114514` 这样的硬编码在代码中的值叫做 **字面量**。

整数类型 Integer

`int` 是 C++ 中的整数类型，通常占用 4 个字节。

`long long` 是 C++ 中的长整数类型，通常占用 8 个字节。

`int` 的取值范围是 -2^{31} 到 $2^{31}-1$

`long long` 的取值范围是 -2^{63} 到 $2^{63}-1$

浮点数类型 Floating Point

`double` 是 C++ 中的浮点数类型，通常占用 8 个字节。

根据 IEEE 754 标准，`double` 类型的精度约为 15 位有效数字。

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

int main() {
    double d = 3.1415; // 声明并初始化一个双精度浮点数变量
    cout << d << "\n";
    return 0;
}
```

字符类型 Character

`char` 是 C++ 中的字符类型，通常占用 1 个字节。

字符是一个字母、数字或者符号，字符字面量用**单引号**括起来。

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

int main() {
    char c = 'A'; // 声明并初始化一个字符变量
    cout << c << "\n";
    return 0;
}
```

输出：

```
A
```

字符串类型 String

`string` 是 C++ 中的字符串类型，需要包含头文件 `string`。

字符串字面量用**双引号**括起来。

```
#include <iostream>
#include <string>

using namespace std;

int main() {
    string s = "Hello, World!"; // 声明并初始化一个字符串变量
    cout << s << "\n";
    return 0;
}
```

布尔类型 Boolean

`bool` 是 C++ 中的布尔类型，只有两个值：`true` 和 `false`。`bool` 类型通常占用 1 个字节。

```cpp
#include <iostream>

using namespace std;

int main() {
    bool b = true; // 声明并初始化一个布尔变量
    cout << b << "\n";
    return 0;
}
```

语句 & 表达式 Statement & Expression

语句是 C++ 程序的基本构建块，每个语句以分号 `;` 结尾。表达式是由运算符和操作数组成的，表达式的值是一个对象。

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

int main() {
    int a = 1, b = 2; // 声明并初始化两个整数变量
    int c = a + b; // 表达式 a + b 的值赋给 c
    cout << c << "\n";
    return 0;
}
```

之前介绍的 `a = 114514;` 就是一个赋值语句。

`int a;` 是一个声明语句。

`cout << c << "\n";` 是一个输出语句。

= 和 ==

在 C++ 中，`=` 是赋值运算符，表示将右边的值赋给左边的变量。

C++ 中的等于运算符是 `==`，表示判断左右两边的值是否相等。

算术运算符 Arithmetic Operators

运算符	功能
`+` (单目)	正
`-` (单目)	负
`*` (双目)	乘法
`/`	除法
`%`	取模
`+` (双目)	加法
`-` (双目)	减法

单目运算符（又称一元运算符）指被操作对象只有一个的运算符，而双目运算符（又称二元运算符）的被操作对象有两个。

算术运算符优先级 Precedence

算术运算符中有两个单目运算符（正、负）以及五个双目运算符（乘法、除法、取模、加法、减法），其中单目运算符的优先级最高。

其中取模运算符 `%` 意为计算两个整数相除得到的余数，即求余数。

而 `-` 为双目运算符时做减法运算符，如 `2-1` ；为单目运算符时做负值运算符，如 `-1` 。

算术运算中的类型转换 Type Conversion

对于双目算术运算符，当参与运算的两个变量类型不同时，C++ 会将其中一个变量的类型转换为另一个变量的类型。一般来说，C++ 会将较低精度的类型转换为较高精度的类型。

例如，对于一个整型（`int`）变量 x 和另一个双精度浮点型（`double`）类型变量 y：

- `x/3` 的结果将会是整型；
- `x/3.0` 的结果将会是双精度浮点型；
- `x/y` 的结果将会是双精度浮点型；
- `x*1/3` 的结果将会是整型；
- `x*1.0/3` 的结果将会是双精度浮点型；

代码是按照顺序执行的

```cpp
#include <iostream>

using namespace std;

int main() {
    int a = 1, b = 2;
    cout << a + b << "\n";
    a = 3;
    cout << a + b << "\n";
    return 0;
}
```

输出：

```
3
5
```

分支结构 Branching

分支结构是程序中的一种控制结构，根据条件的真假执行不同的代码块。

```
int main() {
    int a = 1;
    if (a == 1) {
        cout << "a is 1\n";
    } else {
        cout << "a is not 1\n";
    }
    return 0;
}
```

输出：

```
a is 1
```

循环结构 Looping

循环结构是程序中的一种控制结构，根据条件重复执行代码块。

循环很方便，因为它们可以节省时间、减少错误，并且使代码更具可读性。

循环结构有三种常见的形式：

- `for` 循环
- `while` 循环
- `do...while` 循环

while 循环

`while` 循环是一种在条件为真时重复执行代码块的循环。

```
int main() {
    int i = 0;
    while (i < 5) {
        cout << i << " ";
        i++;
    }
    cout << "\n";
    return 0;
}
```

```
0 1 2 3 4
```

`for` 循环

当你知道循环次数时，使用 `for` 循环是最好的选择。

```cpp
int main() {
    for (int i = 0; i < 5; i++) { // for (初始化; 条件; 更新)
        cout << i << " ";
    }
    cout << "\n";
    return 0;
}
```

```
0 1 2 3 4
```

函数 Function

函数是一段封装好的代码块，可以重复调用。

```
void say_hello() {
    cout << "Hello, World!\n";
}

int main() {
    say_hello();
    return 0;
}
```

输出：

```
Hello, World!
```

函数的参数 Function Parameters

函数可以接受参数，参数是函数的输入。

```
void say_hello(string name) {
    cout << "Hello, " << name << "!\n";
}

int main() {
    say_hello("AtomFirst");
    return 0;
}
```

输出：

```
Hello, AtomFirst!
```

函数的返回值 Function Return Value

函数可以返回一个值，返回值是函数的输出。

```
int add(int a, int b) {
    return a + b;
}

int main() {
    cout << add(1, 2) << "\n";
    return 0;
}
```

输出：

```
3
```

递归 Recursion

递归是函数调用自身的过程。

```
int factorial(int n) {
    if (n == 0) {
        return 1;
    }
    return n * factorial(n - 1);
}

int main() {
    cout << factorial(5) << "\n";
    return 0;
}
```

```
120
```

数组 Array

数组是一种存储多个相同类型数据的数据结构。通过下标访问数组元素。在 C++ 中，数组的下标从 0 开始。

```cpp
#include <iostream>

using namespace std;

int main() {
    int a[5] = {1, 2, 3, 4, 5};
    cout << a[0] << "\n";
    return 0;
}
```

```
1
```

数组的遍历 Array Traversal

遍历数组是访问数组中的每个元素的过程。

```cpp
#include <iostream>

using namespace std;

int main() {
    int a[5] = {1, 2, 3, 4, 5};
    for (int i = 0; i < 5; i++) {
        cout << a[i] << " ";
    }
    cout << "\n";
    return 0;
}
```

```
1 2 3 4 5
```

数组的长度 Array Length

数组的长度是数组中元素的个数。

```cpp
#include <iostream>

using namespace std;

int main() {
    int a[5] = {1, 2, 3, 4, 5};
    cout << sizeof(a) / sizeof(a[0]) << "\n";
    return 0;
}
```

```
5
```

结构体 Struct

结构体是一种用户自定义的数据类型，可以存储多个不同类型的数据。

```
struct Point {
    int x;
    int y;
};

int main() {
    Point p = {1, 2};
    cout << p.x << " " << p.y << "\n";
    return 0;
}
```

```
1 2
```

总结 Summary

- C++ 程序的入口是 `main` 函数。
- C++ 中的数据类型有整数、浮点数、字符、字符串、布尔、长整数等。
- C++ 中的控制结构有分支结构、循环结构。
- 函数是一段封装好的代码块，可以重复调用。
- 数组是一种存储多个相同类型数据的数据结构。
- 结构体是一种用户自定义的数据类型，可以存储多个不同类型的数据。

资源 Resources

学习 C++ 语法基础，你可以参考以下资源：

- C++ Reference
- C++ Tutorial

刷题平台：

- Luogu
- LeetCode
- Codeforces