

C++基础

Peking University

北京大学

ShuwenHe

何书文

1201220707@pku.edu.cn

2023 年 6 月 19 日

目录

第一章 linux 命令	4
第二章 vi 编辑器	6
第三章 基本语法	7
第四章 注释	8
第五章 数据类型	9
第六章 变量类型	10
第七章 变量作用域	11
第八章 常量	13
第九章 修饰符类型	14
第十章 运算符	15
第十一章 控制语句	17
第十二章 函数	18



Chapter

linux 命令

0.1 文件命令

mkdir cpp 创建文件夹

cd cpp 进入文件夹

vi richard.cpp

i 进入 insert 模式

o 进入下一行

int 整型

main() 主函数 function

// 单行注释

“ ” 输入字符串

endl

; 一行语句结束需要用分号结束

return 0 一个函数正确执行完成之后需要用 return 0 来结束

esc 推出 vi 编辑器

:wq 保存并推出 write quit

ls 查看当前文件夹有什么文件 ls - list directory contents

1 文本编辑器

2 C++编译器

命令行使用下面的命令来检查您的系统上是否安装了 gcc

g++ -v g++ 编译器 C++

使用 -o 选项指定可执行程序的文件名

g++ hello.cpp -o hello

./richard 执行编译器编译产生的二进制文件

指定使用 C++14 来编译

```
g++ -std=c++14 cpp.cpp -o cpp
```

2.1 g++常用命令选项

-o

file 生成指定的输出文件, 用在生成可执行文件时。

C++ 中的分号&语句块

编辑编译执行 C++程序



Chapter

vi 编辑器

yy->p 复制粘贴一行 dd 删除当前行

Chapter

基本语法

1 代码介绍

```
#include <iostream>
using namespace std;
int main() {
    cout<<"hello"<<endl;
    return 0;
}
```

C++头文件<iostream>

using namespace std; 告诉编译器使用 std 命名空间。

int main() 是主函数，程序从这里开始执行。

cout<<"Hello World"; 会在屏幕上显示消息 "Hello World"。

下一行 return 0; 终止 main() 函数，并向调用进程返回值 0。

在 C++ 中，分号是语句结束符。也就是说，每个语句必须以分号结束。它表明一个逻辑实体的结束。

2 标识符

3 关键字

Chapter

注释

1 单行多行注释

C++ 支持单行注释和多行注释。注释中的所有字符会被 C++ 编译器忽略。

// - 一般用于单行注释。

/* ... */ - 一般用于多行注释。

Chapter

数据类型

使用编程语言进行编程时，需要用到各种变量来存储各种信息。变量保留的是它所存储的值的内存位置。这意味着，当您创建一个变量时，就会在内存中保留一些空间。

您可能需要存储各种数据类型（比如字符型、宽字符型、整型、浮点型、双浮点型、布尔型等）的信息，操作系统会根据变量的数据类型，来分配内存和决定在保留内存中存储什么。

1 基本的内置类型

表 .5.1: 几种基本的 C++ 数据类型

类型	关键字
布尔型	bool
字符型	char
整型	int
浮点型	float
双浮点型	double
无类型	void
宽字符型	wchar_t

一些基本类型可以使用一个或多个类型修饰符进行修饰：

signed

unsigned

short

long

Chapter

变量类型

Chapter

变量作用域

一般来说有三个地方可以定义变量：

在函数或一个代码块内部声明的变量，称为局部变量。

在函数参数的定义中声明的变量，称为形式参数。

在所有函数外部声明的变量，称为全局变量。

作用域是程序的一个区域，变量的作用域可以分为以下几种：

局部作用域：在函数内部声明的变量具有局部作用域，它们只能在函数内部访问。局部变量在函数每次被调用时被创建，在函数执行完后被销毁。

全局作用域：在所有函数和代码块之外声明的变量具有全局作用域，它们可以被程序中的任何函数访问。全局变量在程序开始时被创建，在程序结束时被销毁。

块作用域：在代码块内部声明的变量具有块作用域，它们只能在代码块内部访问。块作用域变量在代码块每次被执行时被创建，在代码块执行完后被销毁。

类作用域：在类内部声明的变量具有类作用域，它们可以被类的所有成员函数访问。类作用域变量的生命周期与类的生命周期相同。

1 局部变量

在函数或一个代码块内部声明的变量，称为局部变量。它们只能被函数内部或者代码块内部的语句使用。下面的实例使用了局部变量：

```
int sum(){  
    int a,b,sum; // 局部变量声明  
    a = 1,b = 2; // 实际初始化  
    sum = a + b;  
    cout<<"sum= " <<sum<<endl;  
    return 0;  
}
```

2 全局变量

局部变量和全局变量的名称可以相同，但是在函数内，局部变量的值会覆盖全局变量的值。下面是一个实例：

```
// globalVariable 全局变量
int i = 3;
int globalVariable(){
    int i = 5;
    cout<<"i_=_"<<i<<endl;
    return 0;
}
```

3 块作用域

块作用域指的是在代码块内部声明的变量：

```
// blockScope块作用域
int blockScope(){
    int i = 1;
    {
        int i = 2; // 块作用域变量
        cout<<"i_=_"<<i<<endl;
    }
    cout<<"i_=_"<<i<<endl;
    return 0;
}
```



Chapter

常量

1 #define 预处理器

使用 `#define` 预处理器定义常量

```
// #define 预处理器定义常量
```

```
#define LENGTH 3
```

```
#define WIDTH 2
```

```
int areaDefine(){  
    int area;  
    area = LENGTH * WIDTH;  
    cout<<"area_□=□"<<area<<endl;  
    return 0;  
}
```

2 const 关键字

// 使用 `const` 前缀声明指定类型的常量

```
int constConstant(){  
    const int LENGTH_ = 3;  
    const int WIDTH_ = 2;  
    int area;  
    area = LENGTH_ * WIDTH_;  
    cout<<"area_□=□"<<area<<endl;  
    return 0;  
}
```

Chapter

修饰符类型

```
int modifier(){  
    short int i; // 有符号短整数  
    short unsigned int j;  
    j = 50000;  
    i = j;  
    cout<<"j_=_"<<j<<endl;  
    cout<<"i_=_"<<i<<endl;  
    return 0;  
}
```

Chapter

运算符

1 算术运算符

// 算术运算符

```
int arithmeticOperator(){
    int a = 5;
    int b = 3;
    int c;
    cout<<"a=_ " <<a<<endl;
    cout<<"b=_ " <<b<<endl;
    c = a + b;
    cout<<"c=_a+_b=_ " <<c<<endl;
    c = a - b;
    cout<<"c=_a-_b=_ " <<c<<endl;
    c = a * b;
    cout<<"c=_a*_b=_ " <<c<<endl;
    c = a / b;
    cout<<"c=_a/_b=_ " <<c<<endl;
    c = a % b;
    cout<<"c=_a%_b=_ " <<c<<endl;
    int d = 7;
    cout<<"d=_ " <<d<<endl;
    c = d++;
    cout<<"c=_d++=_ " <<c<<endl;
    c = d--;
    cout<<"c=_d--=_ " <<c<<endl;
    return 0;
}
```

2 关系运算符

```
// 关系运算符
int relationalOperator(){
    int a = 5;
    int b = 3;
    cout<<"a_="<<a<<endl;
    cout<<"b_="<<b<<endl;
    int c;
    if (a == b){
        cout<<"a_等于_b"<<endl;
    }else{
        cout<<"a_不等于_b"<<endl;
    }
    if(a < b){
        cout<<"a_小于_b"<<endl;
    }else{
        cout<<"a_不小于_b"<<endl;
    }
    if(a > b){
        cout<<"a_大于_b"<<endl;
    }else{
        cout<<"a_不大于_b"<<endl;
    }
    return 0;
}
```


Chapter

控制语句

1 for 循环

```
// 高斯求和公式求和 1+2+3+...+100

#include <iostream>
using namespace std;
int main() {
    int sum = 0;
    sum = (1+100)*100/2;
    cout << "gauss_sum=" << sum << endl;
    return 0;
}
```

2

for 循环求和 1+2+3+...+100

```
int forSum(){
    int sum = 0;
    for (int i = 1; i <= 100; i++) {
        sum += i;
    }
    cout << "for_sum=" << sum << endl;
    return 0;
}
```

Chapter

函数

1 CSP-J（普及组）2022 年 T1 乘方 (pow)

Chapter

数组

Chapter

STL