面向对象程序语言C++ 标准库

周雅倩
zhouyaqian@fudan.edu.cn
2016/3/3

主要内容

Hello World程序
使用字符串
使用批量数据
组织程序和数据
使用序列式容器
使用库算法
使用关联式容器

C++的Hello World程序

注释
main函数
标准头文件

```
//a small C++ program
#include <iostream>
int main()
{
   std::cout << "Hello, world!" << std::endl;
   return 0; //返回0表示运行成功
}
```

标准输出流
输出操作符
结束当前输出行

操作符 <<

```
std::cout << "Hello, world!" << std::endl;
```

两个操作数
左结合

名字空间

在一个名字前加上std::表明这个名字属于名字空间std

名字空间是相关名字的集会；
标准库使用std来包含它定义的所有名字。
“：：”是生存空间操作符

使用字符串-输入

```
//ask for a person's name, and greet the person
#include <iostream>
#include <string>
int main()
{
   //ask for the person's name
   std::cout << "Please enter your first name:";
   // read the name;
   std::string name; //define name
   std::cin >> name; //read into name
   //write greeting
   std::cout << "Hello, " << name << "!" << std::endl;
   return 0;
}
```

输入操作符
变量

为名字装框输出

```
Please enter your first name:Zhou

**********
*        *
* Hello, Zhou! *
*        *
**********
```


为名字装框输出-code

```
#include <iostream>
#include <string>
int main()
{
    std::cout << "Please enter your first name:";
    std::string name; //define name
    std::cin >> name; //read into name
    //build the message that we intend to write
    const std::string greeting = "Hello, "+ name + "!";
    //build the second and fourth lines of the output
    const  std::string spaces(greeting.size(), ' ');
    const  std::string second = "* " + spaces + " *";
    //build the first and fifth lines of the output
    const  std::string first(second.size(), '*');
    //write it all
    std::cout << std::endl;
    std::cout << first << std::endl;
    std::cout << second << std::endl;
    std::cout << "* " << greeting << " *" << std::endl;
    std::cout << second << std::endl;
    std::cout << first << std::endl;
    return 0;
}
```


初始化、重载和常量

常量　　初始化

```
const std::string greeting = "Hello, "+
name + "!";
```

重载

构造函数和成员函数

```
const  std::string
spaces( greeting.size() , ' ') ;
```


string 类型

定义于标准头文件<string>中

一个string类型的对象可以包含零个或多个字符的序列。

string 类型操作

操作	说明
std::string s	定义一个变量，初始化为空
std::string t=s	定义一个变量t，初始化为s
std::string z(n, c)	定义一个变量z，用字符c的n份复制来初始化为z
os << s	把s中含有的所有字符，写入到os代表的输出流中
is >> s	从is表示的流中读取非空白符的字符，把从is成功读取的字符存入s中。
s + t	包含s中所有字符的复制并且在后面紧接着t中所有字符的复制
s.size()	s中含有的所有字符的个数

变量的定义

```
std::string hello="Hello";
std::string first(10, '*');
std::string name;
```


简化重复的std::

```
//a small C++ program
#include <iostream>
using std::cout; using std::endl;
int main()
{
  cout << "Hello, world!" << endl;
  return 0;
}
```


计算学生成绩

设想有一个课程，学生的期末考试成绩占最终成绩的40%，期中考试成绩占20%，作业的平均成绩占40%。设计程序计算学生的最终成绩。


```
#include <iostream>
#include <string>
#include <iomanip>
#include <ios>
using std::cin;                 using std::setprecision;
using std::cout;                using std::string;
using std::endl;                using std::streamsize;
int main()
{
    cout << "Please enter your first name:";
    string name; //define name
    cin >> name; //read into name
    cout << "Hello, " << name << "!" << endl;
    //ask for and read the midterm and final grades
    cout << "Please enter you midterm and final exam grades:";
    double midterm, final;
    cin >> midterm >> final;
    //ask for the homework grades
    cout << "Enter all your homework grades, followed by end-of-file:";
    //the number and sum of the grades read so far
    int count=0;
    double sum=0;
    double x;//a variable into which to read
    while(cin >> x){
            ++count;
            sum+=x;
    }
    //write the result
    streamsize prec=cout.precision();
    cout << "Your final grade is " << setprecision(3)
         << 0.2 * midterm + 0.4 * final + 0.4 * sum / count
         << setprecision(prec) << endl;
    return 0;
}
```


知识点

控制流，使得后来的输出以给定位数的有效数字显示

```
setprecision(3)
```

表示长度的类型

头文件<iomanip>中定义

```
streamsize prec=cout.precision();
```

头文件<ios>中定义

文件结束标志

检测输入的结束

```
while(cin >> x){/* */}
```

- 如果读取成功，x将会保存读取的值，while条件式就为真
- 如果读取失败（可能是输入结束或输入无效），while条件式就为假，也就不能使用x的值。

计算学生成绩

作业的平均成绩占40%

改为

作业的中值占40%

```
//ask for the homework grades
cout << "Enter all your homework grades, followed by end-of-file:";
//the number and sum of the grades read so far
int count=0;
double sum=0;
double x;//a variable into which to read
while(cin >> x){
        ++count;
        sum+=x;
}
```


使用中值取代平均值

在不知道有多少个数值的情况下，把这些数值都保存下来，这些数值每次只读入一个

读取全部数值后，对它们排序

有效地取出中间的一个值（或者两个）

用vector保存数据集

一个vector可以保存给定类型的一连串的值，

为了容纳新的值，vector可以根据需要来增长

而且它可以高效地取得每个单独的值

vector 实现

```
double x;//a variable into which to read
vector<double> homework;
while(cin >> x){
    homework.push_back(x);
}
```


vector详解

vector是一个容器;

一个单独的vector包含的所有值都有一个相同的类型;

不同的vector可以包含不同类型的对象;

当我们定义一个vector时,必须指定它将要包含的值的类型.

定义模板类对象

```
vector<double> homework;
vector<string> s;
```

生成输出

```
streamsize prec=cout.precision();
cout << "Your final grade is " << setprecision(3)
     << 0.2 * midterm + 0.4 * final + 0.4 * sum / count
     << setprecision(prec) << endl;
```


计算vector的大小

```
std::vector<double>::size_type size =
  homework.size();
```

```
typedef std::vector<double>::size_type
  vec_sz;
  vec_sz size = homework.size();
```


检测vector是否为空

```
if(size==0){
    cout << endl <<"You must enter your grades."
         << "Please try agian." <<endl;
    return 1;
}
```


排序：sort函数

```
sort(homework.begin(),homework.end());
```

表示vector的第一个元素

表示vector的最后一个元素的下一个位置

sort函数定义在头文件<algorithm>中；

它可以把一个容器中的值按照非降序来排列；

sort的参数指定了排序的元素范围

取中间元素

```
double median = size %2 == 0 ?
(homework[mid]+homework[mid-1])/2 :
homework[mid];
```


生成输出

```
streamsize prec=cout.precision();
cout << "Your final grade is " << setprecision(3)
     << 0.2 * midterm + 0.4 * final + 0.4 * median
     << setprecision(prec) << endl;
```