《现代C++编程实现》电子版：https://leanpub.com/c01

控制语句

YouTube频道： hwdong

博客： hwdong-net.github.io

简单语句

- 最简单的语句是只有一个分号的**空语句**。

```
;
```

- 变量定义语句

```
int ival = 3, jval;
auto radius = 2.15;
```

- 表达式后面跟一个分号；构成了**表达式语句**。

```
std::cout<<ival;
```

复合语句

- 花括号{和}括起来的一系列语句构成一个**复合语句**（也称为**程序块**或**语句块**）。

```
#include <iostream>
int main()
{
    auto a=1,b=0;
    {
        a=3;
        auto b = 5;
    }
    std::cout<<a<<'\t'<<b<<std::end;
}
```

复合语句

- 内部块中的b是独立的局部变量，生命周期仅在块内，结束后被销毁。因此，输出的b是外部的b，值为0。

```cpp
#include <iostream>
int main() {
    auto a = 1, b = 0;
    {
        a = 3;
        auto b = 5;       // 内部块中的局部变量b
    }
    std::cout << a << '\t' << b << std::endl; // 输出: 3 0
}
```

控制语句

- 条件语句

  满足某个条件，执行相应程序块

- 循环语句

  满足某个条件，重复执行相应程序块

- 跳转语句

  改变程序执行流程、从当前位置跳转到其他位置的控制语句。

条件语句

if 语句

switch语句

if语句

如果 (休息日)
    在家睡懒觉

如果 (休息日)
    在家睡懒觉
否则
    去上班

如果 (休息日)
    在家睡懒觉
否则，如果（雨雪天）
    坐车去上班
否则
    骑车去上班

if语句

```
if(条件表达式)
    程序块
```

```
if(条件表达式)
    程序块
else
    程序块
```

```
if(条件表达式1)
    程序块
else if(条件表达式2)
    程序块
...
else
    程序块
```

程序块的缩进提高了代码的可读性

```
double  score;
std::cin>>score;
if(score<60)
        std::cout<<"不及格！ "<<std::endl;
```

程序块的缩进提高了代码的可读性

```
double  score;
std::cin>>score;
if(score<60)
        std::cout<<"不及格！ "<<std::endl;
else
        std::cout<<"及格了！ "<<std::endl;
```

程序块的缩进提高了代码的可读性

```
double  score;
std::cin>>score;
if(score<60)
        std::cout<<"不及格！ "<<std::endl;
else  if(score<70)
        std::cout<<"及格！ "<<std::endl;
else  if(score<80)
        std::cout<<"中等！ "<<std::endl;
else  if(score<90)
        std::cout<<"良好！ "<<std::endl;
else
        std::cout<<"优秀！ "<<std::endl;
```

多个互斥的条件

- 使用if嵌套语句，需要注意if-else的匹配是从内到外的

```
double  score;
std::cin>>score;
if(score>=60)
    if(score>90)
        std::cout<<"优秀！ "<<std::endl;
else
    std::cout<<"不及格！ "<<std::endl;
```

- 使用if嵌套语句，需要注意if-else的匹配是从内到外的

```
double  score;
std::cin>>score;
if(score>=60)
    if(score>90)
        std::cout<<"优秀！ "<<std::endl;
    else
      std::cout<<"不及格！ "<<std::endl;
```

- 为了表示正确的程序设计意图，可以借助于{}来控制if和else的匹配。

```
double  score;
std::cin>>score;
if(score>=60){
        if(score>90)
            std::cout<<"优秀！ "<<std::endl;
}
else
        std::cout<<"不及格！ "<<std::endl;
```

- 当一个if或else块里有多条语句时，也要用花括号{}括起来，不然其含义就不对了

```
int a = 100;
if(a<0)
    std::cout<<"a 的绝对值是";
    std::cout<<-a;
else
    std::cout<<"a 的绝对值是";
    std::cout<<a;
```

- 当一个if或else块里有多条语句时，也要用花括号{}括起来，不然其含义就不对了

```
int a = 100;
if(a<0){
    std::cout<<"a 的绝对值是";
    std::cout<<-a;
}
else{
    std::cout<<"a 的绝对值是";
    std::cout<<a;
}
```

switch语句

- 格式：

```
switch(可转化为整型的表达式){
    case (整型或枚举型)常量表达式 1:
        程序块 1
    case (整型或枚举型)常量表达式 2:
        程序块 2
    //...
    default:
        默认程序块
}
```

```
int x;
std::cin>>x;
switch(x){
    case 0:
    case 1:
        std::cout<<"x 是 0 或 1\n";
        break;    //break 关键字用于跳出 switch
    case 2:
        std::cout<<"x 是 2\n";
        break;    //break 关键字用于跳出 switch
    default:
        std::cout<<"x 不是 0，1， 2\n";
        break;    //break 关键字用于跳出 switch
}
```

```
unsigned vowelCnt = 0, nonVowelCnt=0;
char ch;
while(std::cin>>ch){
    switch(ch){
        case 'a':
        case 'e':
        case 'i':
        case 'o':
        case 'u':
            vowelCnt++;
            break;                  //跳出 switch 循环
        default:
            nonVowelCnt++;
            break;                  //跳出 switch 循环
    }
}
```

关于 switch 有几个语法点：

- case 标签必须是整型常量表达式
- case 后面的常量表达式值不能相同
- 某个 case 里定义变量，必须加花括号，否则假如执行其他 case 会出现“作用域里的该变量定义未初始化”
- default 标签可以省略
- break 关键字定义的 break 语句用于跳出整个 switch 语句。

```
//下面的switch 省略了default 标签
int  x,y;
switch(x){
      case  3.14:                    //错：case  标签必须是整型常量表达式
            //...  do  something
            break;
      case  y:                       //错：case  标签必须是整型常量表达式
            //...  do  something
            break;
}
```

```
int  x;
//...
//下面的有2 个case 的标签值都是1，不能相同！
switch(x){
        case  1:
                /...
                break;
        case  1:
                //...
                break;
        case  2:
                //...
                break;
}
```

```
switch (v) {
case 1: int x = 0; //  初始化
        std::cout << x << '\n';
        break;
default:                        // 编译错误：因为default标签，可能导致 'x' 未初始化
        std::cout << "default\n";
        break;
}
```

```
switch (v) {
case 1: {    int x = 0;
        std::cout << x << '\n';
        break;
          }          //   'x' 的作用域在此结束
default: std::cout << "default\n"; //  无错误
        break;
}
```

if/switch语句中的初始化语句

if(初始化语句;表达式)

switch(初始化语句;表达式)

```
{
    auto var = doSomething();
    if(condition(var))
        //if块
    else
        //else块
}
```

在 C++17 中可以写成：

```
    if(auto var = doSomething;condition(var))
        //if块
    else
        //else块
```

一个是代码更加简洁，另外，var只属于if语句，从而不会污染周围环境

```
const string s = "Hello,my weibo name is hw-dong";
const auto it = s.find("Hello");
if (it != std::string::npos)
        std::cout << it << " Hello\n";

const auto it2 = s.find("hw-dong");
if (it2 != std::string::npos)
        std::cout << it2 << " hw-dong\n";
```

在2个if语句中都使用了同样的表示位置的变量it，避免了代码中过多的变量名，每个it只属于它所在的if语句。

```
const string s = "Hello,my weibo name is hw-dong";
if (const auto it = s.find("Hello"); it != std::string::npos)
        std::cout << it << " Hello\n";

if (const auto it = s.find("hw-dong"); it != std::string::npos)
        std::cout << it << " hw-dong\n";
```

循环语句

while语句

for语句

while循环语句

```
while(表达式)
    程序块
```

```
do
        程序块
while(表达式);
```

计算 n的阶乘 *n! = 1*2*3...*n*

```
#include <iostream>
using namespace std;
int main() {
    int n, i{1}, factorial{1};
    cout << "请输入一个正整数: ";
    cin >> n;

    while ( i <= n) {       //只要i小于等于n，就一直执行while循环体
        factorial *= i;       //factorial = factorial * i;
        ++i;
    }

    cout<< n <<"的阶乘是：= "<< factorial;
    return 0;
}
```

计算键盘输入成绩的平均分

```
auto score{0}, average{ 0 };
    auto num{ 0 };
    while (std::cin >> score) {
            average += score;
            num++;
    }
std::cout << "平均成绩是： " << average / num
          << std::endl;
```

break：跳出循环

```
double score, average{ 0 };
auto num{ 0 };
while(std::cin >> score){
        if (score < 0)
                break;          //跳出while循环
         average += score;
        num++;
}
std::cout << "平均成绩是： " << average / num << std::endl;
```

do-while

- while语句的另外一个变种是所谓的**do-while**语句。

  do

  　程序块

  while(表达式);

dh1

do-while

```
double score, average{0};
    auto num{0};
    std::cin >> score;          //注意：先输入得到一个分数
    do {
            if (score < 0)
                    break;          //跳出while循环
            average += score;
            num++;
    } while (std::cin >> score);
    std::cout << "平均成绩是：" << average / num << std::endl;
```

不要忘记分号

幻灯片 34

dh1 dong hongwei, 2019/12/18

continue

- 关键字**continue**用于直接中断循环体里的后续语句执行，回到循环开头重新执行循环。

```
int main() {
    double score, average = {0};
    auto num{0};
    while (std::cin >> score) {
        if (score < 0)
            continue;   //回到循环开头环
        average += score;
        num++;
    }
    std::cout << "平均成绩是：" << average / num << std::endl;
}
```

for循环语句

for (初始表达式；条件表达式；后处理表达式)

程序块

//计算1到100整数之和的代码：

```
auto s{0} ;
for(auto i{1} ; i<=100; i++)
   s += i;
std::cout<<"1到100之间的整数之和是："<<s<<std::endl;
```

for循环语句

for (初始表达式；条件表达式；后处理表达式)

程序块

计算 1 到 100 整数之和

```
auto s{0}; // 初始化累加器

for (auto i{1}; i <= 100; i++) // 循环100次
  s += i;                      // 累加

std::cout << "1 到 100 之间的整数之和是: " << s << std::endl;
```

for (初始表达式; 条件表达式; 后处理表达式)

程序块

```
auto s{0}; // 初始化累加器

for (auto i{1}; i <= 100; i++) // 循环100次
  s += i;                      // 累加

std::cout << "1 到 100 之间的整数之和是: " << s << std::endl;
```

- auto i{1}： 初始化表达式，将循环控制变量 i 初始化为 1。
- i <= 100： 条件表达式，只要 i 的值小于等于 100，循环就会继续执行。
- i++： 后处理表达式，每次循环结束后，将 i 的值自增 1。
- s += i： 循环体，将 i 的值累加到变量 s 中。

for 循环与 while 循环的等价性

for (初始表达式；条件表达式；后处理表达式)

程序块

```
初始化表达式;
while( 条件表达式){
    程序块
    后处理表达式;
}
```

- 计算1到100的整数和可用`while`循环计算：

```
auto s{0};
auto i{1}; // 初始化表达式

while (i <= 100) { // 条件表达式
  s += i;
  i++; // 后处理表达式
}

std::cout << "1 到 100 之间的整数之和是: " << s << std::endl;
```

for 循环的灵活性

- 它的 "初始化表达式" 和 "后处理表达式" 都可以省略。

```
double score, average{ 0 };
auto num{ 0 };
for(; std::cin >> score; ){
    if (score < 0)
        break;  //跳出while循环
    average += score;
    num++;
}
```

for 循环的灵活性

- 也可以将部分逻辑移到条件表达式：

```
double score, average{0.0};
auto num{0};
for (; std::cin >> score && score >= 0;) {
  average += score;
  num++;
}
```

break 和 continue 语句

在 for 循环中的作用与在 while 循环中相同：

- break： 立即终止循环。
- continue： 跳过当前迭代的剩余部分，直接进入下一次迭代。

break 和 continue 语句

```
#include <iostream>

int main() {
  for (auto i{1}; i <= 100; i++) {
    if (i % 3 != 0)
      continue; // 停止后续语句执行，回到循环的条件表达式 "i <= 100"
    std::cout << i << std::endl;
  }
  return 0;
}
```

范围 for 循环 (Range-based for loop)

```
for (类型 变量 : 容器)
    程序块
```

工作原理:

1. **范围:** 范围 可以是数组、std::vector、std::list 等容器，也可以是任何具有 begin() 和 end() 函数的对象（提供迭代器）。
2. **变量声明:** 用于声明一个变量，该变量将依次存储 范围 中的每个元素。通常使用 auto 关键字来让编译器自动推断变量类型。
3. **程序块:** 循环体中的程序块代码会对当前声明的变量进行操作。

Range for

优点:

- **简洁易读:** 语法简单，更容易理解。
- **避免越界:** 不再需要手动管理索引，避免数组越界等错误。
- **通用性:** 适用于各种容器和序列。

```
#include <iostream>

int main() {
  int arr[] = {10, 20, 30, 40, 50};

  for (int element : arr) {
    std::cout << element << " ";
  }
  std::cout << std::endl; // 输出: 10 20 30 40 50
  return 0;
}
```

跳转语句

- break和continue
- return
- goto

goto语句

用goto关键字定义的可以跳转到**标签**位置的**goto 语句**，其格式是：

```
goto 标签名;
//...
标签名:
```

```
std::cout<<"平均成绩是："<< average /num<<std::endl;
double score, average{ 0 };
        auto num{ 0 };
        for(; std::cin >> score; ){
                if (score < 0)
                        goto label;         //跳到标签label处执行
                average += score;
                num++;
        }
        label:
std::cout << "平均成绩是：" << average / num << std::endl;
```

总结

本章介绍了C++中的语句，特别是控制语句：

- **简单语句**：如变量定义、表达式语句。
- **复合语句**：用{}括起的语句块。
- 控制语句：
  - 条件语句：if、switch。
  - 循环语句：while、do-while、for、范围-based for。
  - 跳转语句：break、continue、return、goto。

控制台游戏-Pong游戏

- 参见书

关注我

博客： hwdong-net.github.io

Youtube频道

hwdong

@hwdong · 5.01K subscribers · 558 videos

博客：https://hwdong-net.github.io >

youtube.com/c/4kRealSound and 4 more links