C++ 急速入门

author:Rainboy

2021

Contents

i 前言

c++ 因为是兼容 c 语言的语言。

c++ 本身有cin,cout两个对象用于处理输入输出，但也可以使用scanf,printf这两个 c 风格的函数来处理输入输出。

所以这篇文章里写了两种风格的 IO。c 风格的 IO 函数可以不用学会，但需要理解。

其中标记*的文章可以只用看懂，而不用会使用。

第 1 章 第一个程序 c++ 风格

i c++ 风格 helloWorld

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

- iostream是头文件，是input output stream的缩写。
- 包含iostream头文件后，就可以使用cout，cin这两个对象了。
- main是主要的意思，程序从main的第一句代码开始执行。
- ;号表示一句代码的结束，;前面可以什么也不写，表示空语句

ii 如何编译并运行程序

0.1 方法 1

1. 用codeblocks写入上面的代码，保存运行

0.1 方法 2

1. 使用vim写入上面的代码，保存退出 (<esc>:wq)

2. 编译

```
g++ -g -o 1 helloworld.cpp
```

3. 运行

```
./1
```

iii c 风格 helloWorld

```
#include <cstdio>
int main(){
    printf("Hello World!");
    return 0;
}
```

- cstdio是头文件，是c stand input output的缩写。
- 包含cstdio头文件后，就可以使用printf，scanf这两个函数了。
- main是主要的意思，程序从main的第一句代码开始执行。
- ;号表示一句代码的结束，;前面可以什么也不写，表示空语句

如果编译并运行程序

1. 用vim写入上面的代码，保存退出 (<esc>:wq)
2. 编译

```
g++ -g -o 1 helloworld.cpp
```

3. 运行

```
./1
```

第 2 章 注释

注释是给人看的，代码在执行的时候会忽略。

- // 单行注释，从这个符号开始到行尾都算是注释
- /* ... */ 多行注释，被包涵的内容都算是注释

```cpp
#include <cstdio>
int main(){
    // 这是单行注释
    std::cout << "Hello world!" << std::endl; // 这也是单行注释
    /* 这是注释 */
    /* 这也是
    注释 */
    return 0;
}
```

第 3 章　数据类型

$C++$ 有以下 5 种基本数据类型

类型	关键字	占空间大小	printf 标志	标志对应单词
布尔型	bool	1 Byte	%d	decimal
字符型	char	1 Byte	%c	char
整型	int	4 Byte	%d	decimal
浮点型	float	4 Byte	%f	float
双浮点型	double	8 Byte	%lf	long float

第 4 章　输入输出

c++ 是 c 语言的进化，所以可以使用 c 语言的输入输出函数`scanf`，`printf`，当然 c++ 也有它特有的输入输出方法`cin`，`cout`。

下面的我们来具体看一下如何输入输出数据

c 风格读取代码：

- 输入输出时要指定标记（`%d %f %lf`）
- 速度快
- 读取数字的时候会略过不可见字符（空格，换行符）

c++ 风格读取代码：

- 开启 **IO** 不与 **c** 同步优化后，快！

- 读取的时候简单，不用记指定的标记
- 格式化输出的时候要写的字多，记的东西多 。
- 在对一些格式化的输出的时间不太好用，例如输出：`number is %8d \n\n,hello world`，但在真实比赛中很少用到

关闭同步的方式：

```
std::ios::sync_with_stdio(false);
cin.tie(0); //来解除std::cin和std::cout之间的绑定，提高效率。
```

c++11 以上的语言可以使用如下的匿名函数 (Lambda 捕获)

```
static auto _=[]()
{
    ios::sync_with_stdio(false);
    cin.tie(0);
    return 0;
}();
```

读取字符

无论哪种方法读取数字时都会略过不可见字符（空格，换行符），注意

- 如果使用 cin 读取字符的时候会略过不可见字符
- 如果使用 scanf 读取字符的时候不会略过不可见字符

```
abc 123 8
newline
hello world
```

c++ 风格

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

int main(){
    while ( 1 ) {        //无限循环
        char c;
        cin >> c;
        cout << "read char is " << c << endl;
    }
    return 0;
}
```

c 风格

```
#include <cstdio>

int main(){
    while ( 1 ) {        //无限循环
        char c;
        scanf("%c",&c);
        printf("read char is %c\n",c);
    }
    return 0;
}
```

ii 读取数字

无论哪种方法 (cin,scanf) 读取数字时都会略过不可见字符（空格，换行符）

```
1 1.1 2.0
3
4
5
```

```
#include <iostream>
using namespace std;
int main(){
    int a; //定义一个整型变量a
    double b,c;
    int x,y,z;
    cin >> a >> b >>c; //都写一行太长了，所以换一行写
    cin >> x >> y >> z;
    //每个数占一行
    cout << a << endl;
    cout << b << endl;
    cout << c << endl;
    cout << x << endl;
    cout << y << endl;
    cout << z << endl;
    return 0;
}
```

c 风格数字读取，需要指明要读取的数字的类型（%d,%c,%f 等）

```cpp
#include <cstdio>

int main(){
    int a; //定义一个整型变量a
    double b,c;
    int x,y,z;
    scanf("%d",&a);
    scanf("%lf%lf",&b,&c);// %lf表示要读取一个double类型的数
    scanf("%d%d%d",&x,&y,&z); // %d表示读取整数
    printf("%d %lf %lf\n",a,b,c); // 输出：1 1.100000 2.000000
    printf("%d %d %d",x,y,z); //输出：3 4 5
    return 0;
}
```

iii cout 格式化输出

- 进制相关
  - std::hex 16 进制显示
  - std::oct 8 进制显示
  - std::dec 10 进制显示
- 精度相关
  - std::defaultfloat 默认显示，有几位显示几个小数
  - std::fixed 表示浮点输出应该以固定点或小数点表示法显示，而不是以科学记数字法

  fixed 操作符与 setprecision 操作符一起使用时，setprecision 指定浮点数字的小数点后要显示的位数，而不是要显示的总有效数位数。

  - std::setprecision(n)
- 宽度相关
  - std::setw() 指定宽度
  - std::fill(char) 指定填充的字符

```cpp
/* author: Rainboy email: rainboylvx@qq.com  time: 2021年 04月 22日 星期四
    20:52:19 CST */
#include <bits/stdc++.h>
using namespace std;
typedef long long ll;
const int maxn = 1e6+5,maxe = 1e6+5; //点与边的数量

int n,m;
/* 定义全局变量 */

int main(){
    int a = 7 ,b = 12,c = 16;
    cout << a <<" "<< b <<" "<< c << endl;
    cout << hex << a <<" "<< b <<" "<< c << endl;
    cout << oct << a <<" "<< b <<" "<< c << endl;
    cout << dec << a <<" "<< b <<" "<< c << endl;

    cout << setw(5) << a << endl;
    cout << setw(10) << b << endl;
    cout << setw(10) << setfill('!')<< b ;

    double d = 0.1234567;
    cout << d << endl;
    cout << fixed << setprecision(4) << d << endl; //四舍五入
    //cout << setprecision(4) << d << endl; //不用fixed
    cout << d << endl;
    cout << setprecision(3) ;
    cout << d << endl;

    cout << setprecision(10) << d << endl; //10位

    cout << defaultfloat << d << endl;

    return 0;
}
```

iv printf 格式化输出

- 进制相关
  - %x 16 进制显示
  - %o 8 进制显示
  - %d 10 进制显示
- 精度相关
  - %.8lf double 类型浮点数，保留 8 位小数，四舍五入
  - %.8f float 类型浮点数，保留 8 位小数，四舍五入

- 宽度相关
  - printf("%4d",12),4 个宽度输出,右对齐
  - printf("%04d",12),4 个宽度输出,右对齐，用 0 补全
  - printf("%-4d",12),4 个宽度输出,左对齐

```
#include <bits/stdc++.h>

int main(){
    int a = 7 ,b = 12,c = 16;
    printf("%d %d %d\n",a,b,c); //输出 "7 12 16" 然后换行
    printf("%x %x %x\n",a,b,c); //16进制输出
    printf("%o %o %o\n",a,b,c); //8进制输出

    printf("%5d\n",5); //5个位置，用来输出5
    printf("%10d\n",5);//10个位置，用来输出5，右对齐
    printf("%-10d\n",5);//10个位置，用来输出5，左对齐
    printf("%010d\n",5);

    return 0;
}
```

第 5 章　变量

i 变量的定义

变量就是可以变化的量,它有一个名字，你可以这样理解变量

- 在内存上有一个“箱子”,这个“箱子”有名字，和一个编号（地址）
- 你可以改变“箱子”里的内容（数字）

ii 内存模型

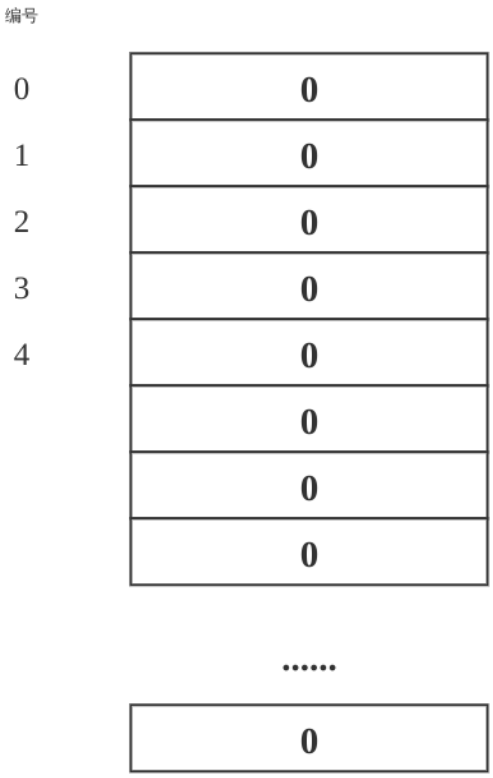


Figure 0.1: 内存模型

- 内存可以认为一个很长的纸条
- 纸条由一个一个小格子组成
- 每个格子都有一个编号，从 0 开始
- 每个格子的大小是 8 bit，也就是 1 byte

iii 变量占用的内存大小

类型	格子数	大小
bool	1	1 byte
char	1	1 byte

类型	格子数	大小
int	4	4 byte
float	4	4 byte
double	8	8 byte

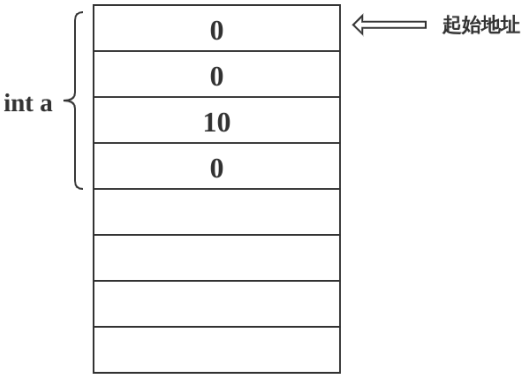


Figure 0.2: int 内存占用

iv 内存大小转换

英文	中文	大小关系
bit	位	$1bit$ 也就是一个 0 或 1 占用的大小
byte	字节	$1byte = 8bit$
kb	千字节	$1kb = 1024byte$
mb	兆字节	$1mb = 1024kb$
gb	千兆字节	$1gb = 1024mb$

第 6 章 运算符

i 定义

运算符就是可以运算的符号，常见的运算符有以下几种：

- 算术运算符
- 关系运算符
- 逻辑运算符
- 位运算符
- 赋值运算符
- 杂项运算符

ii 算术运算符

运算符	描述	实例
+	把两个操作数相加	A + B 将得到 30
-	从第一个操作数中减去第二个操作数	A - B 将得到 -10
*	把两个操作数相乘	A * B 将得到 200
/	分子除以分母	B / A 将得到 2
%	取模运算符，整除后的余数	B % A 将得到 0
++	自增运算符，整数值增加 1	A++ 将得到 11
–	自减运算符，整数值减少 1	A–将得到 9

实例：

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

int main(){
    int a = 21;
    int b = 10;
    int c;
    cout << "c: " << c << endl;
    c = a-b;
    cout << "c: " << c << endl;
    c = a*b;
    cout << "c: " << c << endl;
    c = a/b;
    cout << "c: " << c << endl;
    c = a%b;
    cout << "c: " << c << endl;
    int d = 10; // 测试自增加，自减
    c=d++;
    cout << "c: " << c << endl;

    d=10; //重新赋值
    c=d--;
    cout << "c: " << c << endl;

    return 0;
}
```

结果如下：

```
c: 0
c: 11
c: 210
c: 2
c: 1
c: 10
c: 10
```

iii 关系运算符

运算符	描述	实例
==	检查两个操作数的值是否相等，如果相等则条件为真。	(A == B) 不为真。
!=	检查两个操作数的值是否相等，如果不相等则条件为真。	(A != B) 为真。
>	检查左操作数的值是否大于右操作数的值，如果是则条件为真。	(A > B) 不为真。

运算符	描述	实例
<	检查左操作数的值是否小于右操作数的值，如果是则条件为真。	(A < B) 为真。
>=	检查左操作数的值是否大于或等于右操作数的值，如果是则条件为真。	(A >= B) 不为真。
<=	检查左操作数的值是否小于或等于右操作数的值，如果是则条件为真。	(A <= B) 为真。

实例：

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

int main()
{
   int a = 21;
   int b = 10;
   int c ;

   if( a == b )
   {
      cout << "Line 1 : a 等于 b" << endl ;
   }
   else
   {
      cout << "Line 1 : a 不等于 b" << endl ;
   }
   if ( a < b )
   {
      cout << "Line 2 : a 小于 b" << endl ;
   }
   else
   {
      cout << "Line 2 : a 不小于 b" << endl ;
   }
   if ( a > b )
   {
      cout << "Line 3 : a 大于 b" << endl ;
   }
   else
   {
      cout << "Line 3 : a 不大于 b" << endl ;
   }
   /* 改变 a 和 b 的值 */
   a = 5;
   b = 20;
   if ( a <= b )
   {
      cout << "Line 4 : a 小于或等于 b" << endl ;
   }
   if ( b >= a )
   {
      cout << "Line 5 : b 大于或等于 a" << endl ;
   }
   return 0;
}
```

结果如下：

```
Line 1 : a 不等于 b
Line 2 : a 不小于 b
Line 3 : a 大于 b
Line 4 : a 小于或等于 b
Line 5 : b 大于或等于 a
```

iv 逻辑运算符

运算符	描述	实例
&&	称为逻辑与运算符。如果两个操作数都 true，则条件为 true。	(A && B) 为 false。
|| 称 如	为逻辑或运算符。果两个操作数中有任意一个 true，则条件为 true。	(A || B) 为 true。
!	称为逻辑非运算符。用来逆转操作数的逻辑状态，如果条件为 true 则逻辑非运算符将使其为 false。	!(A && B) 为 true。

实例：

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

int main()
{
   int a = 5;
   int b = 20;
   int c ;

   if ( a && b )
   {
      cout << "Line 1 : 条件为真"<< endl ;
   }
   if ( a || b )
   {
      cout << "Line 2 : 条件为真"<< endl ;
   }
   /* 改变 a 和 b 的值 */
   a = 0;
   b = 10;
   if ( a && b )
   {
      cout << "Line 3 : 条件为真"<< endl ;
   }
   else
   {
      cout << "Line 4 : 条件不为真"<< endl ;
   }
   if ( !(a && b) )
   {
      cout << "Line 5 : 条件为真"<< endl ;
   }
   return 0;
}
```

结果如下：

```
Line 1 : 条件为真
Line 2 : 条件为真
Line 4 : 条件不为真
Line 5 : 条件为真
```

v 位运算符

TODO

vi 赋值运算符

下表列出了 C++ 支持的赋值运算符：

运算符	描述	实例
=	简单的赋值运算符，把右边操作数的值赋给左边操作数	C = A + B 将把 A + B 的值赋给 C
+=	加且赋值运算符，把右边操作数加上左边操作数的结果赋值给左边操作数	C += A 相当于 C = C + A
-=	减且赋值运算符，把左边操作数减去右边操作数的结果赋值给左边操作数	C -= A 相当于 C = C - A
*=	乘且赋值运算符，把右边操作数乘以左边操作数的结果赋值给左边操作数	C = *A* 相当于 *C* = *C* A
/=	除且赋值运算符，把左边操作数除以右边操作数的结果赋值给左边操作数	C /= A 相当于 C = C / A
%=	求模且赋值运算符，求两个操作数的模赋值给左边操作数	C %= A 相当于 C = C % A
<<=	左移且赋值运算符	C «= 2 等同于 C = C « 2
>>=	右移且赋值运算符	C »= 2 等同于 C = C » 2
&=	按位与且赋值运算符	C &= 2 等同于 C = C & 2
^=	按位异或且赋值运算符	C ^= 2 等同于 C = C ^ 2
|=	按位或且赋值运算符	C |= 2 等同于 C = C | 2

vii 杂项运算符

下表列出了 C++ 支持的其他一些重要的运算符。

运算符	描述
sizeof	sizeof 运算符返回变量的大小。例如，sizeof(a) 将返回 4，其中 a 是整数。
Condition ? X : Y	条件运算符。如果 Condition 为真? 则值为 X : 否则值为 Y。
,	逗号运算符会顺序执行一系列运算。整个逗号表达式的值是以逗号分隔的列表中的最后一个表达式的值。
&	指针运算符 & 返回变量的地址。例如 &a; 将给出变量的实际地址。
*	指针运算符 * 指向一个变量。例如，*var; 将指向变量 var。

viii C++ 中的运算符优先级

TODO

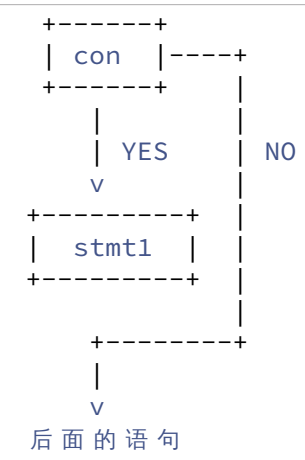
第 7 章 控制结构

if 语句的形式有两种：

1. 没有 else

```
if ( 条件 )
    语句1;
```

- 条件 -> con
- 语句 1 -> stmt1
- 语句 2 -> stmt2



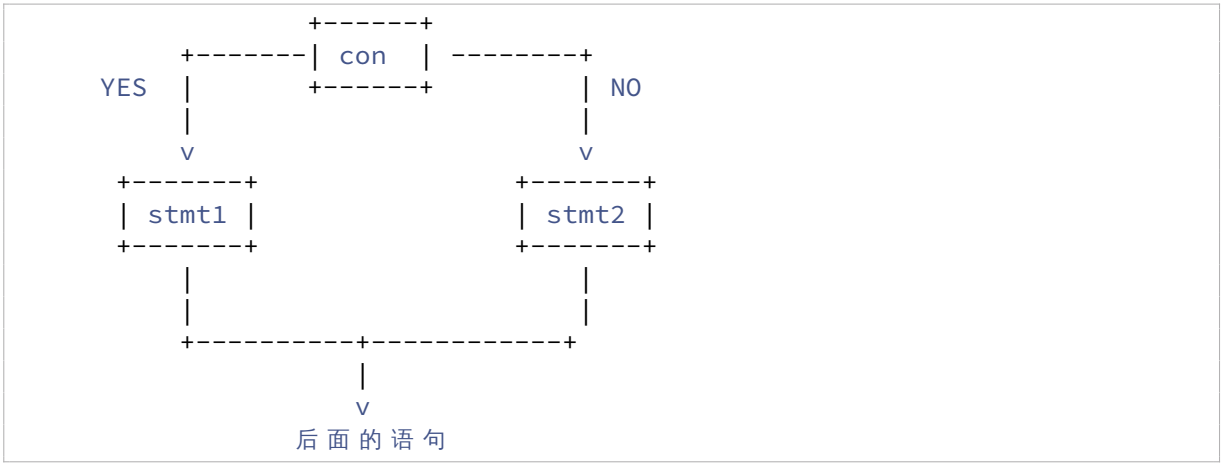
样例：输入一个分数，判断是否及格，及格输出 YES，否则什么也不做。

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

int main(){
    int a;
    cin >> a;
    if( a>=60 ) //是否>=60
        cout <<"yes\n";
    return 0;
}
```

2. 有else

```
if ( 条件 )
    语句1;
else
    语句2;
```



样例：输入一个分数，判断是否及格

```
#include <bits/stdc++.h>    //万能头文件
using namespace std;
typedef long long ll;

int main(){
    int a; //定义一个变量
    cin >> a; //输入一个值
    if( a >= 60)    //if只能控制后面的一句话
        cout << "YES\n";
    else
        cout << "NO\n";
    return 0;
}
```

注意

- **if**可以单独出现
- 如果有**else**，必须要有**if**
- **if**和**else**都只能控制后面紧跟着的一名话
  - 如果想控制多句话，用{}括起来，形成一个语句块
- **if**和**else**合起来算一句话，如果没有**else**，**if**算一句话

i if语句之间的嵌套

思想下面的几个代码的运行结果

```
#include <bits/stdc++.h>     //万能头文件
using namespace std;
typedef long long ll;

int main(){
    int a; //定义一个变量
    cin >> a;
    if( a >= 60)     //if只能控制后面的一句话
        //下面的if else 形成一句话 ，被上面的if(a>=60) 控制
        if ( a>= 70)
            cout << ">= 70\n";
        else
            cout << ">=60 ,< 70\n";
    else
        if( a > 30)
            cout << "a > 30,a<60\n";
    return 0;
}
```

任务：编写这个代码，回答下面的问题

- 分别输入20，30，40，60 70 80查看输出结果是什么，分析原因

```
#include <bits/stdc++.h>     //万能头文件
using namespace std;
typedef long long ll;

int main(){
    int a; //定义一个变量
    cin >> a; //输入一个值
    if( a >= 60)     //if只能控制后面的一句话
        //下面的if else 形成一句话 ，被上面的if(a>=60) 控制
        if ( a>= 70)
            cout << ">= 70\n";
        else
            cout << ">=60 ,< 70\n";
    else
        if( a > 30)
            cout << "a > 30,a<60\n";
    return 0;
}
```

- 分别输入60，70，80，90查看输出结果是什么，分析原因

学习到：

if和**else**都只能控制后面紧跟着的一名话,如果想控制多句话，用{}括起来，形成一个语句块

ii if ... else if ...else if...

编写运行下面的代码，分别输入50 60 75 85 95，查看输出结果是什么，分析原因。

```
#include <bits/stdc++.h>     //万能头文件
using namespace std;
typedef long long ll;

int main(){
    int a; //定义一个变量
    cin >> a;//输入一个值
    if( a >= 90)  //60
        cout << "you xiu\n";
    else if ( a >= 80 )
        cout << "liang hao\n";
    else if( a >=60)
        cout << "ji ge\n";
    else
        cout << "bu ji ge\n";
    return 0;
}
```

与下面的写

```
#include <bits/stdc++.h>     //万能头文件
using namespace std;
typedef long long ll;

int main(){
    int a; //定义一个变量
    cin >> a;//输入一个值
    if( a >= 90)  //60
        cout << "you xiu\n";
    else {
        if ( a >= 80 )
            cout << "liang hao\n";
        else {
            if( a >=60)
                cout << "ji ge\n";
            else
                cout << "bu ji ge\n";
        }
    }
    return 0;
}
```

- **else**和它上面的最近的同级别的**if**配对

第 8 章　循环结构

循环有三种基本的循环：

- for 循环
- while 循环
- do while 循环

这三种循环本质上一样的，可以互相转换。

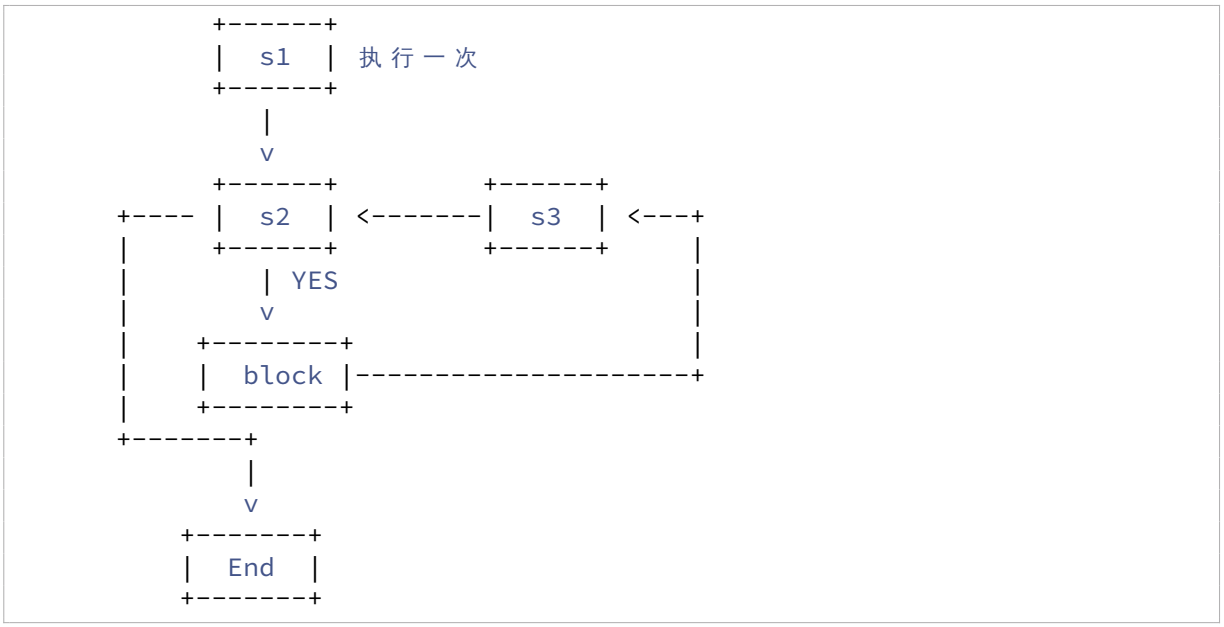
i　for 循环

语法如下

```
for( s1 : s2 : s3) {
    block;
}
```

- s1,s2,s3 分别表示 语句 1，语句 2，语句 3
- block 表示多条语句

它的执行过程如下



例子 1 ：输出 1 到 10，然后输出 10 到 1

```
#include <bits/stdc++.h> //万能头文件
using namespace std;

int main(){
    for(int i=1;i<=10;++i){ //输出1 到 10
        cout << i << " ";
    }
    cout << endl;
    for(int i=10;i>=1;--i) //这一行没有带括号{}，
        cout << i << " "; //for与if 一样默认控制下面的一句话
    cout << endl;

    return 0;
}
```

例子 2 ：计算 1 到 100 的和

```
#include <bits/stdc++.h> //万能头文件
using namespace std;

int main(){
    int sum = 0; // sum 是英文[求和]的意思
    for(int i=1;i<=100;++i){
        sum+= i;
    }
    cout << sum << endl;
    return 0;
}
```

特别说明： for 小括号内的三条语句都可以写成空语句，看下面的代码

代码 3

```
#include <bits/stdc++.h> //万能头文件
using namespace std;

int main(){
    int i=1;
    for( ; i < 10 ; ){
        cout << i << " ";
        i = i+2;
    } // 想一想这个for的输出结果是什么
    // 这个for的 s1 是空语句 ; 执行这名话相当于什么也什么做
    // s2 是一个判断条件
    // s3 是空语句 ; 执行这名话相当于什么也什么做
    return 0;
}
```

输出结果是：1 3 5 7 9

代码 4

```
#include <bits/stdc++.h> //万能头文件
using namespace std;

int main(){
    for( ; ; ){
        cout << "hello" << endl;
    } // 想一想这个for的输出结果是什么
    // 这个for的 s1 是空语句 ; 执行这名话相当于什么也什么做
    // s2 是一个空语句 ; c++ 认为空语句的条件是真
    // s3 是空语句 ; 执行这名话相当于什么也什么做
    return 0;
}
```

输出结果是：无限输出hello, 可以按ctrl + c 强制终止程序

ii continue 语句

> continue ：继续，遇到它，跳转到 s3 去执行

```
#include <bits/stdc++.h> //万能头文件
using namespace std;

int main(){
    for(int i=1;i<=10;++i){
        if( i == 5)
            continue;
        cout << i << " ";
    }
    return 0;
}
```

输出结果是：1 2 3 4 6 7 8 9 10 没有 5

iii break 语句

> break : 打断，遇到它，终止控制 break 的那个 for 语句

```
#include <bits/stdc++.h> //万能头文件
using namespace std;

int main(){
    for(int i=1;i<=10;++i){
        if( i == 5)
            break;
        cout << i << " ";
    }
    return 0;
}
```

输出结果是：1 2 3 4 没有 5 和后面的数字

iv 嵌套循环

```
#include <bits/stdc++.h> //万能头文件
using namespace std;

int main(){
    for(int i=1;i<=3;++i){
        cout << i << ": ";
        for(int j=1;j<=i;++j){
            cout << j << " ";
        }
        cout << endl;
    }
    return 0;
}
```

输出结果如下：

```
1: 1
2: 1 2
3: 1 2 3
```

例子：输出 9x9 乘法表

```
#include <bits/stdc++.h> //万能头文件
using namespace std;

int main(){
    for(int i=1;i<=9;++i){
        for(int j=1;j<=i;++j){
            cout << i <<"x" <<j << "=" << i*j;
        }
        cout << endl;
    }
    return 0;
}
```

```
1x1=1
2x1=2 2x2=4
3x1=3 3x2=6 3x3=9
4x1=4 4x2=8 4x3=12 4x4=16
5x1=5 5x2=10 5x3=15 5x4=20 5x5=25
6x1=6 6x2=12 6x3=18 6x4=24 6x5=30 6x6=36
7x1=7 7x2=14 7x3=21 7x4=28 7x5=35 7x6=42 7x7=49
8x1=8 8x2=16 8x3=24 8x4=32 8x5=40 8x6=48 8x7=56 8x8=64
9x1=9 9x2=18 9x3=27 9x4=36 9x5=45 9x6=54 9x7=63 9x8=72 9x9=81
```

第 9 章　数组

i　引言

如果你需要写一个程序记录 3 个人的成绩

```
int a,b,c;
scanf("%d%d%d",&a,&b,&c);
```

那如果你需要记录 1000 个人的成绩呢？就不能用定义一个一个变量的方式了，要使用数组

```
int a[1000];
for(int i = 0; i<= 999 ;i++)
    scanf("%d",&a[i]);
```

0.9　1. 什么是数组？

数组：一组数字 ### 2. 如何定义数组数组类型数组名 [数组大小]

```
// 1. 直接定义一个数组，名字叫a，有100个元素
int a[100];

// 2.定义时直接初始化，有3个元素，3个元素分别为 1 2 3
int abc[3] = {1,2,3};

// 3.部分初始化 其中第一个元素为10 其它元素为0
int d[3] = {10};

// 3. 不直接指定数组的大小，数组的大小由初始化列表的元素个数来决定
// foo大小为5
int foo[]  = {1,2,3,4,5};

// 4 定义了一个char数组
char bar[] = {'a','1','2'};
```

0.9 3. 数组定义后如何使用

- 数组的下标从 0 开始，如
  - `int a[3]`有三个元素, 分别是`a[0],a[1],a[2]`
- 使用下标访问数组里的一个元素，就像普通的变量一样

```
int a[3];         //定义
scanf("%d",&a[1]);      //输入
a[0] = 1;         //赋值
a[0] = a[1] + a[2]; //运算
```

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

int main(){
    int a[5];
    for(int i=0;i<=4;++i){ //由键盘输入5个数字
        scanf("%d",&a[i]); //  存到数组里
    }

    printf("array is : ");
    for(int i=0;i<=4;++i){
        printf("%d ",a[i]); //输出每个元素并空一格
    }

    return 0;
}
```

0.9 4. 一个小题目: 猴子选大王

TODO

0.9 5. 总结与注意事项

- 定义数组时，必须知道数组的确定大小，也就是说要用常量表达式，`int a[常量表达式]`
  - `int n;int a[n];` 错，因为n是变量
- 定义数组里可以直接初始化`int a[3] = {1,2,3}`
- 定义数组里可以通过初始化来指明数组大小`int a[] = {1,2,3}`
- 数组的下标从 0 开始，`int a[3]`,共 3 个元素，第一个`a[0]`,最后一个`a[2]`
- 通过数组的下标来访问元素

第 10 章 字符串

TODO

第 11 章 函数

什么是函数: 带有名字的代码块 出现的原因:

1. 复用
2. 递归

使用函数的好处:

1. 模块化
2. 易读
3. 易修改 (只用修改对应的函数)

如何定义

```
[返回值类型] [函数名字](参数列表) {
    函数体
}
```

返回值类型: void int double char bool

返回值语句: return

return 应该如何使用?

- 如果函数是 void 类型 那么应该这样写`return;`
- 如果函数是 int 类型 `return` 数值/表达
- 其它类似

注意: 函数遇到 return, 函数就结束了

0.11 问题 1 求最大值

- 第一行, 一个整数 n, 表示要求接下来 n 行一对整数的最大值
- 第 n+1 行, 一个整数 m, 表示要求接下来 m 行三个整数的最大值

```
3
1 2
3 1
2 3
3
1 2 3
3 2 1
1 4 7
```

不使用函数

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

int main(){
    int n,m;
    std::cin >> n;
    for(int i=1;i<=n;++i){
        int a,b;
        std::cin >> a >> b;
        if( a > b )
            std::cout << a << std::endl;
        else
            std::cout << b << std::endl;
    }
    std::cin >> m;
    for(int i=1;i<=m;++i){
        int a,b,c;
        std::cin >> a >> b >> c;
        if( a < b)  a = b; //a是 a b 之间的最大值
        if( a < c ) a = c;//a是 a c 之间的最大值
        std::cout << a << std::endl;
    }
    return 0;
}
```

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

int mymax(int a,int b){
    if( a > b) return a;
    return b;
}

int main(){
    int n,m;
    std::cin >> n;
    for(int i=1;i<=n;++i){
        int a,b;
        std::cin >> a >> b;
        std::cout << mymax(a, b) <<endl;
    }
    for(int i=1;i<=m;++i){
        int a,b,c;
        std::cin >> a >> b >> c;
        std::cout << mymax(mymax(a, b),c) << std::endl;
    }
    return 0;
}
```

发现使用了函数后, 代码量变少了, 可读性变高了.

i 参数传递

- 值传递 (本质上来讲, 所有的参数传递都是值传递)
- 引用传递 参数是类似这种 (int &a,int &b)

怎么去记忆?

如果一个函数是`void change(int a)`, 那么调用`int b= 100;change(b)` 相当于

```
void change(int a){
    //int a = b; 相当于隐式的调用了这名话
    change(a) // int a = a;
}
```

引用传递

```
void change(int& a)
    // int& a = b;
}
```

ii 指针

第 12 章 递归

TODO

第 13 章 总结

1. `ctrl+alt+t`打开 terminal
2. terminal 下编译代码`g++ -g -o 1 1.cpp`
3. 基础数据类型与大小 (byte)
    - bool 1
    - char 1
    - int 4

- float 4
- double 8
- long long 8

4. 变量 variable

- 定义 `int a,b,c;`
- 全局变量，未初化时值为 0
- 局部变量，未初化时值随机

6. 运算符

- 算术 `+ - * / % ++ --`
- 关系 `> >= < <= == !=`
- 逻辑 `! && ||`
- 三元 `? :`
- sizeof

7. 控制结构

- if
- if … else …
- else if

8. 循环

- for(int i=1;i<=10;i++) { … }
- for(; 1 ;) { … }
- for(; ;) { … }
- `while( expr ){ } <=> for(; expr ;){ ... }`

9. 数组

- 一维数组
- 定义 `int a[10];`
- 定义并初始化
- `int a[3] = {1,2,3};`
- `int a[3] = {1};`
- `int a[3] = {0};`
- 二维数组

10. 函数

- 定义

- 返回值
- 参数列表
- void
- return 执行就结束这个函数
- 参数的拷贝传值
- mymax 函数
- 指针，指针作为参数
- 引用，引用作为参数
- myswap 函数

11. 递归

- 本质 自己调用自己
- 每个人作事的过程都一样，但面对的事情的规模不一样
- 问题分解！
- 叠方块
- 前进
- 回溯
- 递归树，递归的模型是在树上行走