第 2 章（二）

数据的输入和输出

I/O 流

- 在 C++中，将数据从一个对象到另一个对象的流动抽象为“流”。流在使用前要被建立，使用后要被删除。
- 数据的输入与输出是通过 I/O 流来实现的，cin 和 cout 是预定义的流类对象。cin 用来处理标准输入，即键盘输入。cout 用来处理标准输出，即屏幕输出。
- 从流中获取数据的操作称为提取操作，向流中添加数据的操作称为插入操作。

预定义的插入符和提取符

- “<<”是预定义的插入符，作用在流类对象 cout 上便可以实现项标准输出设备输出。
  - cout << 表达式 << 表达式...
- 标准输入是将提取符作用在流类对象 cin 上。
  - cin >> 表达式 >> 表达式...
- 提取符可以连续写多个，每个后面跟一个表达式，该表达式通常是用于存放输入值的变量。例如：
  - int a, b;
  - cin >> a >> b;

常用的 I/O 流类库操纵符

操纵符名	含　　义
dec	数值数据采用十进制表示
hex	数值数据采用十六进制表示
oct	数值数据采用八进制表示
ws	提取空白符
endl	插入换行符，并刷新流
ends	插入空字符
setsprecision(int)	设置浮点数的小数位数（包括小数点）
setw(int)	设置域宽

例：cout << setw(5) << setprecision(3) << 3.1415;

if 语句

If 语句的语法形式

if (表达式) 语句

例：if (x > y) cout << x;

if (表达式) 语句 1 else 语句 2

例：if (x > y) cout << x;
else cout << y;

if (表达式 1) 语句 1
else if (表达式 2) 语句 2
else if (表达式 3) 语句 3
...
else 语句 n

例 2-2 输入一个年份，判断是否闰年

```
#include <iostream>
using namespace std;
int main() {
    int year;
    bool isLeapYear;
    cout << "Enter the year: ";
    cin >> year;
    isLeapYear = ((year % 4 == 0 && year % 100 != 0) || (year % 400 == 0));
    if (isLeapYear)
        cout << year << " is a leap year" << endl;
    else
        cout << year << " is not a leap year" << endl;
    return 0;
}
```

嵌套的 if 结构

- 语法形式

```
if( )
        if( ) 语句 1
        else 语句 2
else
        if( ) 语句 3
        else 语句 4
```

- 注意
  - 语句 1、2、3、4 可以是复合语句；
  - 每层的 if 与 else 配对，或用 { } 来确定层次关系。

例 2-3：输入两个整数，比较两个数的大小

```
#include<iostream>
using namespace std;
int main() {
        int x, y;
        cout << "Enter x and y:";
        cin >> x >> y;
        if (x != y)
                if (x > y)
                        cout << "x > y" << endl;
                else
                        cout << "x < y" << endl;
        else
                cout << "x = y" << endl;
        return 0;
}
```

switch 语句

- 语法形式

  switch（表达式）

```
{ case   常量表达式 1：语句 1
  case   常量表达式 2：语句 2
         ┆
  case   常量表达式 n：语句 n
  default :          语句 n+1
}
```

- 执行顺序
  - 以 case 中的常量表达式值为入口标号，由此开始顺序执行。因此，每个 case 分支最后应该加 break 语句。
- 注意
  - case 分支可包含多个语句，且不用{ }。
  - 表达式、判断值都是 int 型或 char 型。
  - 如果若干分支执行内容相同可共用一组语句。

例 2-4：输入一个 0～6 的整数，转换成星期输出

```
#include <iostream>
using namespace std;
int main() {
    int day;
    cin >> day;
    switch (day) {
    case 0: cout << "Sunday" << endl; break;
    case 1: cout << "Monday" << endl; break;
    case 2: cout << "Tuesday" << endl; break;
    case 3: cout << "Wednesday" << endl; break;
    case 4: cout << "Thursday" << endl; break;
    case 5: cout << "Friday" << endl; break;
    case 6: cout << "Saturday" << endl; break;
    default:
      cout<<"Day out of range Sunday .. Saturday"<< endl;   break;
    }
    return 0;
}
```

while 语句

- 语法形式

  while（表达式）语句

  可以是复合语句，其中必须含有改变条件表达式值的语句。

- 执行顺序

  先判断表达式的值，若为 true 时，执行语句。

例 2-5 求自然数 1～10 之和

```
#include <iostream>
using namespace std;
int main() {
  int i = 1, sum = 0;
  while (i <= 10) {
      sum += i;  //相当于 sum = sum + i;
      i++;
  }
  cout << "sum = " << sum << endl;
        return 0;
}
```

do-while 语句

- do-while 语句的语法形式

  do 语句

  while (表达式)

  可以是复合语句，其中必须含有改变条件表达式值的语句。

- 执行顺序

  先执行循环体语句，后判断条件。

  表达式为 true 时，继续执行循环体。

例 2-6：输入一个数，将各位数字翻转后输出

```
#include <iostream>
using namespace std;
int main() {
        int n, right_digit, newnum = 0;
        cout << "Enter the number: ";
        cin >> n;
        cout << "The number in reverse order is  ";
        do {
                right_digit = n % 10;
                cout << right_digit;
                n /= 10;  /*相当于 n=n/10*/
        } while (n != 0);
    cout << endl;
        return 0;
}
```

例 2-7 用 do-while 语句编程，求自然数 1~10 之和

```
#include <iostream>
using namespace std;
int main() {
        int i = 1, sum = 0;
        do {
                sum += i;
                i++;
        } while (i <= 10);
        cout << "sum = " << sum << endl;
        return 0;
}
```

对比下面的程序

程序 1：

```
#include <iostream>
using namespace std;
int main() {
  int i, sum = 0;
  cin >> i;
  while (i <= 10) {
    sum += i;
    i++;
  }
  cout<< "sum= " << sum
      << endl;
  return 0;
}
```

程序 2：

```
#include <iostream>
using namespace std;
int main() {
  int i, sum = 0;
  cin >> i;
  do {
    sum += i;
    i++;
  } while (i <= 10) ;
  cout << "sum=" << sum
      << endl;
  return 0;
}
```

for 语句

- for 语句语法形式：

  for（初始语句；表达式1；表达式2）语句

  循环前先求解　为true时执行循环体　每次执行完循环体后求解

- for 语句的另一种形式：范围 for 语句：

  for（声明：表达式）

  　语句

例 2-8：输入一个整数，求出它的所有因子

```
#include <iostream>
using namespace std;
int main() {
        int n;
        cout << "Enter a positive integer: ";
        cin >> n;
        cout << "Number  " << n << "   Factors  ";
        for (int k = 1; k <= n; k++)
          if (n % k == 0)
            cout << k << "  ";
  cout << endl;
  return 0;
}
```

运行结果 1：

```
Enter a positive integer: 36
Number  36  Factors  1  2  3  4  6  9  12  18  36
```

运行结果 2：

```
Enter a positive integer: 7
Number  7   Factors  1  7
```

嵌套的控制结构、其他控制语句

例 2-10 输入一系列整数，统计出正整数个数 i 和负整数个数 j,读入 0 则结束。

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

int main() {
        int i = 0, j = 0, n;
```

```
    cout <<"Enter some integers please (enter 0 to quit):" << endl;
    cin >> n;
    while (n != 0) {
      if (n > 0) i += 1;
      if (n < 0) j += 1;
      cin >> n;
    }
    cout << "Count of positive integers: " << i << endl;
    cout << "Count of negative integers: " << j << endl;
    return 0;
}
```

其他控制语句

- break 语句

  使程序从循环体和 switch 语句内跳出，继续执行逻辑上的下一条语句。不宜用在别处。

- continue 语句

  结束本次循环，接着判断是否执行下一次循环。

- goto 语句

  使程序的执行流程跳转到语句标号所指定的语句。不提倡使用。

自定义类型

类型别名：为已有类型另外命名

- typedef 已有类型名 新类型名表
  - 例：

```
typedef double Area, Volume;
typedef int Natural;
Natural i1,i2;
Area a;
Volume v;
```

- using 新类型名 = 已有类型名;
  - 例：

```
using Area = double;
```

using Volume = double;

枚举类型

- 定义方式：
  将全部可取值一一列举出来。
- 语法形式：
  enum 枚举类型名 {变量值列表};
  例：enum Weekday {SUN, MON, TUE, WED, THU, FRI, SAT};
  默认情况下
  SUN=0，MON=1，TUE=2，……，SAT=6

C++包含两种枚举类型：

- 不限定作用域枚举类型：
  enum 枚举类型名 {变量值列表};
- 限定作用域的 enum 类将在第 4 章介绍。

不限定作用域枚举类型说明：

- 枚举元素是常量，不能对它们赋值
  例如有如下定义
  enum Weekday {SUN, MON, TUE, WED, THU, FRI, SAT};
  不能写赋值表达式：SUN = 0
- 枚举元素具有默认值，它们依次为： 0,1,2,……。
- 也可以在声明时另行指定枚举元素的值，如：
  enum Weekday{SUN=7,MON=1,TUE,WED, THU,FRI,SAT};
- 也可以在声明时另行指定枚举元素的值；
- 枚举值可以进行关系运算。
- 整数值不能直接赋给枚举变量，如需要将整数赋值给枚举变量，应进行强制类型转换。
- 枚举值可以赋给整型变量。

例 2-11

- 设某次体育比赛的结果有四种可能：胜（WIN）、负（LOSE）、平局（TIE）、比赛取消（CANCEL），编写程序顺序输出这四种情况。
- 分析：

比赛结果只有四种可能，可以声明一个枚举类型。

```
#include <iostream>
using namespace std;
enum GameResult {WIN, LOSE, TIE, CANCEL};
int main() {
    GameResult result;
    enum GameResult omit = CANCEL;
    for (int count = WIN; count <= CANCEL; count++) {
      result = GameResult(count);
      if (result == omit)
        cout << "The game was cancelled" << endl;
      else {
        cout << "The game was played ";
        if (result == WIN)      cout << "and we won!";
        if (result == LOSE)       cout << "and we lost.";
        cout << endl;
      }
    }
    return 0;
}
```

auto 类型与 decltype 类型

- auto：编译器通过初始值自动推断变量的类型
  - 例如：auto val = val1 + val2;
    如果 val1+val2 是 int 类型，则 val 是 int 类型；
    如果 val1+val2 是 double 类型，则 val 是 double 类型。
- decltype：定义一个变量与某一表达式的类型相同，但并不用该表达式初始化变量
  - 例如：decltype(i) j = 2;