实训 3　程序的流程控制

实训目的

- 了解结构化程序设计方法以及 3 种基本程序结构。
- 理解和掌握 3 种基本程序结构。
- 学会利用 3 种基本结构进行简单的程序设计。

实训内容

1.3.1　输入出租车类型和里程，计算打车的费用

```
//Example cp31.cpp
#include <iostream>
#include <cstring>
using namespace std;

int main()
{
    int taxiType;
    float s;
    float money;
    cout << "输入出租车类型（0 或 1）：";
    cin >> taxiType;
    cout << "输入里程：";
    cin >> s;
    if(s < 3 )
        money = 6;   //3 公里以内 6 元
    else
      if( taxiType == 0)
          money = 6 + (s-3)*1.5;     //车型 0 每公里 1.5 元
      else
          money = 6 + (s-3)*1.2;     //其他车型每公里 1.2 元

    cout <<"打车的费用为："<<money<<endl;

    return 0;
}
```

运行结果如图 3-1 所示。

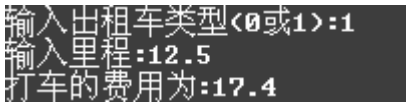

图 3-1　cp31.cpp 的运行结果

【调试与思考】

（1）如果考虑夜间 11:00 后起步价为 9 元，程序如何修改？

（2）如果不同车型的起步价也不同，如何处置？

1.3.2　输入一个成绩，判断成绩级别

```
//Example cp32.cpp
#include <iostream>
#include <cstring>
using namespace std;

int main()
{
    int score;
    cout << "请输入成绩：";
    cin >> score;

    switch(score/10)
    {
        case 0:
        case 1:
        case 2:
        case 3:
        case 4:
        case 5:cout<<"不合格";break;
        case 6:
        case 7:cout<<"合格";break;
        case 8:cout<<"良好";break;
        case 9:
        case 10:cout<<"优秀";break;
        default:cout<<"成绩输入可能有问题";break;
    }

    cout<<endl;
    return 0;
}
```

运行结果如图 3-2 所示。

图 3-2　cp32.cpp 的运行结果

score 为 80，则 switch 后面的整型表达式 score/10 等于 8，对应 case 8: …，所以输出“良好”。

如果输入 95，则 switch 后面的整型表达式 score/10 等于 9，对应 case 9: …，后面对应的模块为空，由于没有 break 语句，则转到 case 10 后面的模块，输出“优秀”后遇到 break 退出，如图 3-3 所示。

图 3-3　cp32.cpp 的运行结果

【调试与思考】

（1）如果考虑成绩是实数，如何修改程序？

（2）如果大于等于 85 分是优秀，如何处理？

1.3.3　计算 1+2+3+…+100

```
//Example cp33.cpp
#include <iostream>
using namespace std;
int main()
{
    int i = 1;              // 第一个人准备投 1 元
    int s = 0;
    while(i <=100)
    {
        s = s+ i;
        i = i +1;
    }
    cout <<"1+2+3+…+100="<<s<<endl;

    return 0;
}
```

运行结果如图 3-4 所示。

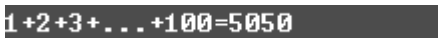

图 3-4　cp33.cpp 的运行结果

其实，上面程序中第 i 人正好投币 i 元，如果第 i 人投 2*i 元，则程序就是计算：

2 + 4 + 6 + … + 200

【调试与思考】

（1）读者可以考虑计算以下的和：

1）1+3+5+…+99

2）1+1/2+1/3+1/4+…+1/100

（2）请读者考虑用 do…while 语句和 for 语句如何实现上面的求和。

1.3.4 计算并输出 2～100 之间的素数之和

分析：所谓素数指的是除了 1 和自身以外，没有其他因子。根据这个定义，可以用除了 1 和自身以外的小于该数的数去除该数，如果所有的数都不能整除，则可以判断该数是素数。

例如，对于自然数 7，用 2、3、4、5、6 去除 7，发现都不能整除，则判断出 7 是素数；而对于自然数 15，用 2～14 之间的数（包括 2 和 14）去除 15，发现 3、5 可以整除 15，则 15 不是素数。

程序如下：

```
//Example cp34.cpp
#include <iostream>
using namespace std;

int main()
{
    int s=0;
    int i,j;
    for(i = 2 ; i <= 100 ; i++)          //产生 2～100 之间的数 i
    {
        for( j = 2 ; j <= i-1 ; j++)     //产生除了 1 和 i 以外的数
            if( i % j == 0 ) break;      //发现 i 的因子
        if( j ==i )                      //2～i-1 之间没有发现因子，则 i 是素数
            s = s + i;                   //累加素数
    }

    cout << "2 到 100 之间的素数之和为："<<s<<endl;

    return 0;
}
```

运行结果如图 3-5 所示。

```
2到100之间的素数之和为:1060
```

图 3-5 cp34.cpp 的运行结果

程序中判断素数的循环嵌在大循环之内，完成每个 i 是否是素数的判断。当所有的 j 都判断完成时，没有一个 j 是 i 的因子，则 i 必然是素数。这时候 j 必然不符合循环条件 j<=i-1，由于 j 是递增的，所以这时候 j 其实等于 i。

其实判断素数只要用 2～i/2 之间的数，甚至是 2～sqrt(i)之间的数也可以完成判断的任务。使用 sqrt 需要包含 cmath 头文件。

【调试与思考】

（1）如果考察的范围不是固定的，比如求 a 到 b 之间的素数之和，如何修改程序？

（2）如果求非素数之和，如何处理？

1.3.5　输出三角形图形

输出如下三角形图形：

```
    *
   ***
  *****
 *******
*********
```

程序如下：

```
//Example cp35.cpp
#include <iostream>
using namespace std;

void main()
{
	int i,j;
	for(i = 1 ; i <= 5 ; i++)
	{
		for( j = 1 ; j <= 5 - i ; j++)
			cout << " ";
		for( j = 1 ; j <= 2*i-1 ; j++)
			cout << "*";
		cout <<endl;
	}
}
```

分析：这是一类图形输出的例子。设计这样的程序需要抓住内外循环的关联。

其中需要找到以下规律：

- 星号前的空格个数与行数之间的关系。
- 星号个数与行数之间的关系。

对于本题，行号 i 从 1 开始循环到 5，规律如下：

- 星号前的空格个数为 5-i 个。
- 星号个数为 2*i-1 个。

【调试与思考】请读者考虑输出下面的图形：

```
    *********
   *******
  *****
 ***
*
```

```
   *
  ***
 *****
*******
 *****
  ***
   *
```

```
1
23
456
7890
```

1.3.6 打印日历

输出 2009 年 1 月的日历。2009 年 1 月 1 日是星期四，要求按照正常的日历排版格式，具体如图 3-6 所示。

图 3-6 cp36.cpp 的运行结果

输出日历格式需要考虑以下因素：

- 输出月标题、星期标题。
- 输出 1 月 1 日前面的空格。
- 换行输出的处理。

假设每天输出占位 4 个字符（前后各一个空格，天数占 2 个字符），1 月 1 日前面的空格个数应该是 4*4。

换行处理需要考虑 1 月 1 日星期几和是否是 7 的倍数，其判断式为：

(week + day) % 7 == 0

具体程序如下：

```
//Example cp36.cpp
#include <iostream>
#include <cstring>
#include <iomanip>

using namespace std;

int main()
{
    int year = 2009;
    int month = 1;
    int week = 4;
    int space;
    int day ;
        //每月的标志
        cout << setw(10) << year
             << setw(4) << "年"
             << setw(4) << month
             << setw(4) << "月" << endl <<endl;
        cout << setw(4)  << "日"
             << setw(4) << "一"
```

```
            << setw(4) << "二"
            << setw(4) << "三"
            << setw(4) << "四"
            << setw(4) << "五"
            << setw(4) << "六" <<endl;
        //输出空格
        for( space = 0 ; space < week ; space ++)
            cout<<"      ";          //每天对应 4 个空格

        for(day = 1 ; day <= 31; day ++)
        {
            cout << setw(4) << day ;
            if ((day +week) % 7 ==0 )
                cout <<endl;
        }
        cout <<endl;

        return 0;
}
```

【调试与思考】读者可以考虑如何输出一年的日历表，同时可以考虑如何将两个月或 3 个月的日历横排输出。

1.3.7　利用循环求不定方程的解

数学中的不定方程是指其中自变量的值不能直接解出，可能有多个解。例如，假设 x、y 都是非负整数，则方程 x+9y=20 的解有：

x=2，y=2

x=11，y=1

x=20，y=0

这 3 种情况。其判断求解的依据不仅是方程的等式，还有自变量是非负整数这个特征限制。

下列问题就是一个典型的不定方程问题：假设有 20 元，可以由 10 元、5 元和 1 元组成，问可以有哪些组成方式？

设 20 元中 10 元、5 元和 1 元各为 x、y、z 张，则有：

x*10+5*y+z*1 == 20

下面的程序列出了所有的解。

```
//Example cp37.cpp
#include <iostream>
#include <cstring>

using namespace std;

int main()
{
    int x,y,z;
```

```
    cout <<"10 元" <<'\t'
         <<"5 元" <<'\t'
          <<"1 元" <<'\t'<<endl;
    for(x = 0 ; x <= 2; x++)
        for(y = 0 ; y <= 4 ; y++)
            for( z = 0;z <= 20 ;z++)
                if(x*10+y*5+z == 20)
                    cout <<x <<'\t'
                         <<y <<'\t'
                          <<z <<'\t'<<endl;
    return 0;
}
```

运行结果如图 3-7 所示。

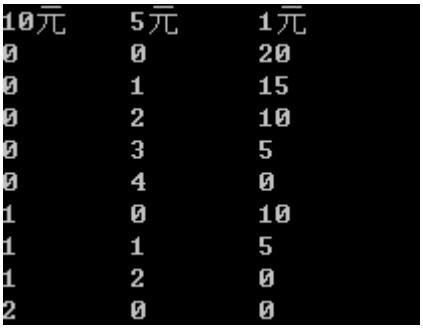

图 3-7 cp37.cpp 的运行结果

【调试与思考】

（1）考虑能否减少循环的嵌套？

（2）如果 10 元、5 元和 1 元都必须有，如何修改程序？