第3章 C++上机实验指导

3.1 上机实验题

3.1.1 实验1 上机环境和C++基础编程练习

1. 实验目的

了解 Visual C++或 C++ Builder 集成开发环境，或 Linux 下 C++开发环境。

熟悉集成开发环境的基本编辑命令及其功能键，练习使用常用功能菜单命令。

学习完整的 C++程序开发过程（编译、连接、调试、运行及查看结果）。

理解简单的 C++程序结构。

掌握 C++基本数据类型变量、常量的使用，理解其内存的概念。

学习 C++中各种运算式的使用。

2. 实验内容

编写程序实现求解二次方程的解。

基本要求如下：

仔细阅读上机实验环境指导，熟悉上机环境。

设置用户目录路径，编写程序，编译、调试并查看运行结果。

学习如何根据编译信息，定位并排除语法错误和语法警告。

学习并模仿主教材上的程序编写风格，养成良好编程习惯。

3.1.2 实验2 控制结构编程练习

1. 实验目的

理解 C++程序的控制结构。

熟练使用条件判断。

熟练使用各种循环结构。

进一步提高程序调试与修改编译错误的能力。

注意提高程序的可读性。

2. 实验内容

编写程序模拟实现以下游戏：

现有两人玩猜拳游戏，每人可分别用各自的拳头表示三种物体：石头（rock）、剪刀（scissors）和布（cloth）中的一种，两人同时出拳，游戏胜负规则如下。

石头对剪刀：石头赢；

剪刀对布：剪刀赢；

布对石头：布赢。

3.1.3 实验3 函数编程练习

1. 实验目的

掌握函数的声明、定义方法。

理解函数参数的传递。

掌握函数调用方法。

2. 实验内容

(1) 编写一个函数 Take(),该函数返回正整数 n 的第 k 位数字,例如,如果 n 为 543210,则调用函数 Take(n,0)返回数字 0,而调用函数 Take(n,3)返回数字 3。注意数字的位次顺序为从右到左,从 0 开始。

(2) 编写一个带默认参数的函数,计算 4 次多项式的值,测试该函数。

3.1.4 实验4 构造数据类型编程练习

1. 实验目的

理解数组的概念,掌握数组应用的一般方法。

理解指针的概念,掌握指针的使用。

深入理解指针与数组的区别与联系。

2. 实验内容

(1) 编写并测试一个函数,将一个二维数组顺时针旋转 90°。例如将数组

1	2	3		7	4	1
4	5	6	转换为：	8	5	2
7	8	9		9	6	3

(2) 定义一个函数 invert()将数组 a 中的 n 个整数按相反顺序存放。程序实现要求如下：

用数组作为函数形参实现该函数 invert(int A[], int n),函数调用时实参为数组。

用数组作为函数形参实现该函数 invert(int A[], int n),函数调用时实参为指针。

用指针作为函数形参实现该函数 invert(int *A, int n),函数调用时实参为数组。

用指针作为函数形参实现该函数 invert(int *A, int n),函数调用时实参为指针。

(3) 用枚举构造数据类型改写实验 2 中的猜拳游戏。

3.1.5 实验5 类与对象编程练习

1. 实验目的

掌握类的定义,根据具体需求设计类。

深入理解 C++中类的封装性。

会根据类创建各种对象。

掌握对象的各种成员的使用方法。

通过定义构造函数实现对象的初始化。

2. 实验内容

定义一个 FDAccount 类，用以描述一个定期存折(fixed deposit)，实现现金支取、余额合计、信息显示等。存折基本信息包括账号、账户名称、存款余额、存款期限(以月为单位)、存款利率(以百分点为单位)等。

3.1.6 实验 6 继承与派生编程练习

1. 实验目的

掌握派生与继承的概念与使用方法。

运用继承机制对现有的类进行重用。

掌握继承中的构造函数与析构函数的调用顺序。

为派生类设计合适的构造函数初始化派生类。

深入理解继承与组合的区别。

2. 实验内容

设计一个人员类 person 和一个日期类 date，由人员类派生出学生类 student 和教师类 professor，学生类和教师类的数据成员 birthday 为日期类。

3.1.7 实验 7 多态性编程练习

1. 实验目的

理解多态性的概念。

掌握如何用虚函数实现动态联编。

掌握如何利用虚基类。

2. 实验内容

设计一个飞机类 plane，由它派生出歼击机 fighter 类和轰炸机 bomber 类，歼击机 fighter 类和轰炸机 bomber 类又共同派生出歼轰机(多用途战斗机)。利用虚函数和虚基类描述飞机类及其派生的类族。

3.1.8 实验 8 类模板编程练习

1. 实验目的

理解类模板的概念。

掌握类模板的定义、实例化过程。

掌握类模板运用。

通过类模板，进一步理解 C++中代码重用思想。

2. 实验内容

定义一个通用队列模板类，队列中数据元素类型可以是字符型、双精度型和其他数据类型，队列的基本操作包括队列初始化、队列的析构、进队列、出队列、判断队列是否为空和判断队列是否是满队列，测试该队列。

3.1.9 实验9 输入/输出流与文件系统编程练习

1. 实验目的

理解C++的输入/输出流的概念。

熟悉I/O流的工作过程。

熟悉各种格式标志与各种格式控制方法。

分清文本文件与二进制文件的区别。

掌握二进制文件的输入/输出的步骤与操作。

会运用文件指针以及各种标志。

2. 实验内容

定义一个学生类,包含学生的学号、姓名和成绩等基本信息,将学生信息写入二进制文件student.dat中,实现对学生信息的显示、查询和删除等基本功能。

3.1.10 实验10 string类字符串处理编程练习

1. 实验目的

理解C++中的string类。

使用C++标准类库中的string定义字符串对象。

能使用string类成员函数、操作符对字符串对象进行各种操作。

2. 实验内容

编写程序,对输入文件中的内容进行分析,统计文件的行数、单词数和每个单词出现的次数。

3.2 上机实验题参考解答

3.2.1 实验1 上机环境和C++基础编程练习

编写程序实现求解二次方程的解。

【参考解答】

```
#include<iostream>
#include<cmath>
using namespace std;
void main()
{
  float a,b,c;                                    //方程系数
  cout<<"Enter the coefficients of a quadratic equation:"<<endl;
  cout<<"\ta:";
  cin>>a;
  cout<<"\tb:";
  cin>>b;
  cout<<"\tc:";
  cin>>c;
//显示方程
```

```
    cout<<"The equation is : "<<a<<" * x * x + "<<b<<" * x + "<<c<<" = 0"<<endl;
    float d = b * b - 4 * a * c;
    if(d <= 0)
      cout<<"The equation has not real solution."<<endl;
    float x1 = ( - b + sqrt(d))/(2 * a);            //解一
    float x2 = ( - b - sqrt(d))/(2 * a);            //解二
    cout<<"The solutions are:"<<endl;               //输出解
    cout<<"\tx1 = "<<x1<<endl;
    cout<<"\tx2 = "<<x2<<endl;
    cout<<"Check:"<<endl;                           //检查解
    cout<<"\ta * x1 * x1 + b * x1 + c = "<<a * x1 * x1 + b * x1 + c<<endl;
    cout<<"\ta * x2 * x2 + b * x2 + c = "<<a * x2 * x2 + b * x2 + c<<endl;
}
```

运行结果：

```
Enter the coefficients of a quadratic equation:
      a:1↙
      b:3↙
      c:2↙
The equation is : 1 * x * x + 3 * x + 2 = 0
The solutions are:
      x1 = - 1
      x2 = - 2
Check:
      a * x1 * x1 + b * x1 + c = 0
      a * x2 * x2 + b * x2 + c = 0
```

【思考问题】

尝试输入不同的方程系数，会出现什么情况呢？

程序总能够求得方程的解吗？该注意些什么？

如果程序出现溢出错误，怎么办？

3.2.2 实验 2 控制结构编程练习

(1) 现有两人玩猜拳游戏，每人可分别用各自的拳头表示三种物体：石头(rock)、剪刀(scissors)和布(cloth)中的一种，两人同时出拳，游戏胜负规则如下。

石头对剪刀：石头赢；

剪刀对布：剪刀赢；

布对石头：布赢。

【参考解答】

```
#include<iostream>
using namespace std;
#define Player1 0;
#define Player2 1;
#define Tie 2;
void main()
{
  int choice1,choice2,winner;
```

```
    cout<<"Choose rock(0),cloth(1),or scissors(2):"<<endl;
    cout<<"Player #1:";              //1 号选手选择
    cin>>choice1;
    cout<<"Player #2:";              //2 号选手选择
    cin>>choice2;
    switch(choice2-choice1)          //判断胜负结果
    {
       case 0:
            winner=Tie;
            cout<<"\tYou tied."<<endl;
            break;
       case -1:
       case  2:
            winner=Player1;
            cout<<"\tPlayer #1 wins."<<endl;
            break;
       case -2:
       case 1:
            winner=Player2;
            cout<<"\tPlayer #2 wins."<<endl;
    }
}
```

运行结果:

```
Choose rock(0),cloth(1),or scissors(2):
Player #1:2↙
Player #2:0↙
      Player #2 wins.
```

【思考问题】

程序可以用 if-else 嵌套实现吗? 如果可以,请加以改写。

如果游戏规定:比赛连续进行三次,赢两次或两次以上者将最终获胜,如何改写程序?

(2) 一个生性非常贪婪的百万富翁遇到一个陌生人,陌生人对富翁说:“我每天给你十万元,而你第一天只需要给我一分钱,第二天我仍然给你十万元,你给我二分钱,第三天我仍然给你十万元,你给我四分钱……直到满一个月(30 天)为止,好吗? 如果你同意,我们就定一个契约。”贪心的富翁很高兴,欣然接受了这个契约。请编写一个程序,计算双方的付出分别是多少?

【参考解答】

```
#include<iostream>
using namespace std;
void main()
{
  int i;
  long int stranger=1,millionaire=100000,balance;
  for(i=2;i<=30;i++)
  {
    stranger=2*stranger;
    millionaire=millionaire+100000;
  }
```

```
  stranger = stranger/100;
  cout <<"stranger:"<< stranger <<"￥"<< endl;              //陌生人付出的钱
  cout <<"millionaire:"<< millionaire <<"￥"<< endl;        //富翁付出的钱
  //陌生人获得的差额
  cout <<" \tbalance = stranger - millionaire = "<< stranger - millionaire <<
    "￥"<< endl;
}
```

运行结果：

```
stranger:5368709￥
millionaire:3000000￥
       balance = stranger - millionaire = 2368709￥
```

【思考问题】

用 while 循环改写该程序。

用 do…while 循环改写该程序。

3.2.3 实验 3 函数编程练习

(1) 编写一个函数 Take()，该函数返回正整数 n 的第 k 位数字，例如，如果 n 为 543 210，则调用函数 Take(n,0)返回数字 0，而调用函数 Take(n,3)返回数字 3。注意数字的位次顺序为从右到左，从 0 开始。

【参考解答】

```
#include<iostream>
using namespace std;
int Take(long ,int);
void main()
{
  int n,k;
  cout <<"Enter a integer:";
  cin >> n;
  do  {
      cout <<"location:";
      cin >> k;
      cout <<"Digit number "<< k <<" of "<< n <<" is "<< Take(n,k)<< endl;
  }while(k > 0);
}
int Take(long n,int k)
{
    for(int i = 0;i < k;i++)
      n/ = 10;     //去掉最右边的数字
    return n % 10;
}
```

运行结果：

```
Enter a integer:2467389 ↙
location:3 ↙
Digit number 3 of 2467389 is 7
```

```
location:5↙
Digit number 5 of 2467389 is 4
location:0↙
Digit number 0 of 2467389 is 9
```

(2) 编写一个带默认参数的函数,计算 4 次多项式的值,测试该函数。

【参考解答】

```
#include<iostream>
using namespace std;
double polynomial(double,double,double = 0,double = 0,double = 0,double = 0);
                                                                    //函数原型
void main()
{
  double x;
  cout<<"Please enter x: ";
  cin>>x;
  cout<<"polynomial(x,1) = "<<polynomial(x,1)<<endl;
  cout<<"polynomial(x,1,2) = "<<polynomial(x,1,2)<<endl;
  cout<<"polynomial(x,1,2,3) = "<<polynomial(x,1,2,3)<<endl;
  cout<<"polynomial(x,1,2,3,4) = "<<polynomial(x,1,2,3,4)<<endl;
  cout<<"polynomial(x,1,2,3,4,5) = "<<polynomial(x,1,2,3,4,5)<<endl;
}
double polynomial(double x,double a0,double a1,double a2,double
  a3,double a4)
{
  return a0 + (a1 + (a2 + (a3 + a4 * x) * x) * x) * x;
}
```

运行结果:

```
Please enter x: 2.36↙
polynomial(x,1) = 1
polynomial(x,1,2) = 5.72
polynomial(x,1,2,3) = 22.4288
polynomial(x,1,2,3,4) = 75.0058
polynomial(x,1,2,3,4,5) = 230.108
```

【思考问题】

在 C++中,使用带有默认形参值的函数有什么好处?

C++中使用带有默认形参值的函数时,应该注意什么问题?

使用带有默认形参值的函数时,如果省去了某个实参,应该注意什么?

能够将这个程序改为用函数模板实现吗?

3.2.4 实验 4 构造数据类型编程练习

(1) 编写并测试一个函数,将一个二维数组顺时针旋转 90°。例如将数组

$$\begin{matrix}1&2&3\\4&5&6\\7&8&9\end{matrix}$$ 转换为: $$\begin{matrix}7&4&1\\8&5&2\\9&6&3\end{matrix}$$

【参考解答】

```
#include<iostream>
using namespace std;
const int SIZE=3;
typedef int Matrix[SIZE][SIZE];
void print(Matrix);
void rotate(Matrix);
void main()
{
   Matrix m={{1,2,3},{4,5,6},{7,8,9}};
   cout<<"Before matrix is rotated:"<<endl;
   print(m);              //输出旋转前的二维数组
   cout<<"After matrix is rotated:"<<endl;
   rotate(m);             //旋转二维数组
   print(m);              //输出旋转后的二维数组
}
void print(Matrix A)      //输出数组元素
{
   int i,j;
   for(i=0;i<SIZE;i++)
   {
     for(j=0;j<SIZE;j++)
        cout<<A[i][j]<<"\t";
     cout<<endl;
   }
}
void rotate(Matrix A)     //旋转二维数组
{
   int i,j;
   Matrix temp;
   for(i=0;i<SIZE;i++)
    {
      for(j=0;j<SIZE;j++)
          temp[i][j]=A[SIZE-j-1][i];
    }
    for(i=0;i<SIZE;i++)
    {
      for(j=0;j<SIZE;j++)
          A[i][j]=temp[i][j];
    }
}
```

运行结果：

```
Before matrix is rotated:
1       2       3
4       5       6
7       8       9
After matrix is rotated:
7       4       1
8       5       2
9       6       3
```

【思考问题】

数组会溢出吗？为什么C++中数组可能溢出,其实质是什么？

改写上面的程序,使得数组溢出,然后思考C++是如何处理数组溢出的？

(2) 定义一个函数invert()将数组a中的n个整数按相反顺序存放。程序实现要求如下：

用数组作为函数形参实现该函数invert(int A[], int n),函数调用时实参为数组。

用数组作为函数形参实现该函数invert(int A[], int n),函数调用时实参为指针。

用指针作为函数形参实现该函数invert(int *A, int n),函数调用时实参为数组。

用指针作为函数形参实现该函数invert(int *A, int n),函数调用时实参为指针。

要求1：用数组作为函数形参实现该函数invert(int A[], int n),函数调用时实参为数组。

【参考解答】

```
#include<iostream>
using namespace std;
void invert(int A[], int n)  //将数组元素反序存放
{
   int i,j,temp;
   int m=(n-1)/2;
   for(i=0;i<=m;i++)
   {
       j=n-1-i;
       temp=A[i];
       A[i]=A[j];
       A[j]=temp;
   }
}
void main()
{
   int i;
   int a[10]={0,1,2,3,4,5,6,7,8,9};
   for(i=0;i<10;i++)
     cout<<a[i]<<"\t";
   invert(a,10);
   printf("The array has been reverted:\n");
   for(i=0;i<10;i++)
     cout<<a[i]<<"\t";
   cout<<endl;
}
```

要求2：用数组作为函数形参实现该函数invert(int A[], int n),函数调用时实参为指针。

【参考解答】

```
#include<iostream>
using namespace std;
void invert(int A[], int n)  //将数组元素反序存放
{
```

```
    int i,j,temp;
    int m = (n - 1)/2;
    for(i = 0;i <= m;i++)
    {
        j = n - 1 - i;
        temp = A[i];
        A[i] = A[j];
        A[j] = temp;
    }
}
void main()
{
    int i;
    int a[10] = {0,1,2,3,4,5,6,7,8,9};
    int * p;
    for(i = 0;i < 10;i++)
      cout << a[i]<<"\t";
    p = a;
    invert(p,10);
    printf("The array has been reverted:\n");
    for(p = a;p < a + 10;p++)
      cout << * p <<"\t";
    cout << endl;
}
```

要求 3：用指针作为函数形参实现该函数 invert(int * A, int n)，函数调用时实参为数组。

【参考解答】

```
#include < iostream >
using namespace std;
void invert(int * A, int n)  //将数组元素反序存放
{
    int * i, * j,temp, * p;
    int m = (n - 1)/2;
    i = A;
    j = A + n - 1;
    p = A + m;
    for(;i < p;i++,j--)
    {
        temp = * i;
        * i = * j;
        * j = temp;
    }
}
void main()
{
    int i;
    int a[10] = {0,1,2,3,4,5,6,7,8,9};
    for(i = 0;i < 10;i++)
      cout << a[i]<<"\t";
```