# 实验1 简单程序开发

## 实验目的

1. 初步掌握一种常用C++语言继承环境(IDE)的使用，了解程序的组成与运行一个程序的过程。
2. 掌握C++的基本语法及程序的基本结构，了解C++中的数据类型，常量与变量
3. 了解C++的输入与输出方法
4. 初步掌握变量的引用。
5. 掌握调试一般程序的方法。

## 实验工具与准备工作

复习相关课程内容。一台安装有Visual C++、Dev-C++等集成环境软件。

## 实验内容

阅读下列程序，写出执行结果，然后输入程序，调试程序，比较结果的正确性。

文本

描述已自动生成

文本

描述已自动生成

1. 掌握函数调用、静态变量以及变量的作用范围，例如for循环中的i的生命周期，回答框图问题。

文本

描述已自动生成

1. 程序填空。下面程序实现两个变量值交换的操作，请根据题意完成填空。

知识点：引用、函数参数传递，函数先声明后使用。

文本

描述已自动生成

1. 改正下面程序中的错误，使其能够正确输出数组第6各元素的值。回答框中的问题。

文本

描述已自动生成

1. 输入两个整数，由函数实现将它们按由小到大的顺序进行排列，要求使用函数引用。
2. \*编一个程序，用一个函数实现对n个整数类型数据进行从小到大的排序，函数原型如下：

void Sort(int a[],int n)

1. \*编一个程序，用一个函数实现求n个整数数据的最大值，函数原型如下：

int Max(int a[],int n)

## 实验提示

1. 属于基本输出cout方面的题目，应注意换行
2. 所有指针的长度为4
3. 静态变量n在程序运行期间由系统分配固定的存储空间的方式并被赋予初值16，函数每调用一次，n的值将+1.
4. 应记住通过循环赋值交换两个变量的方法，参考答案：b tem
5. 数组a的第1个元素为a[0],…,第6个元素为a[5]
6. 注意出现逆序时交换a与b
7. 可以通过n-1趟（第0趟，第1趟，第n-2趟）排序来完成，注意第i趟将在a[i],a[i+1],…a[n-1]中的最小值交换到a[i]处。
8. 可设max=a[0],对a[1]….a[n-1]中的每个元素，只要大于max，就将其复制给max。

题7和题8中使用动态数组来从键盘输入任意多个数据进行排序。

int \* a;

int n;

cout<<"intput the number of integers"<<endl;

cin>>n;

cout<<"input"<<n<<"个整数,用空格间隔，输入完毕后按回车键"<<endl;

a=new int[n];

for(int i=0;i<=n-1;i++)

cin>>a[i];

## 实验步骤

1. 创建工程exp1-1。
2. 建立源程序文件main.cpp
3. 编译及运行程序。

## 测试与结论

将实验运行后的屏幕显示写在此处

# 实验2 类与对象

## 实验目的

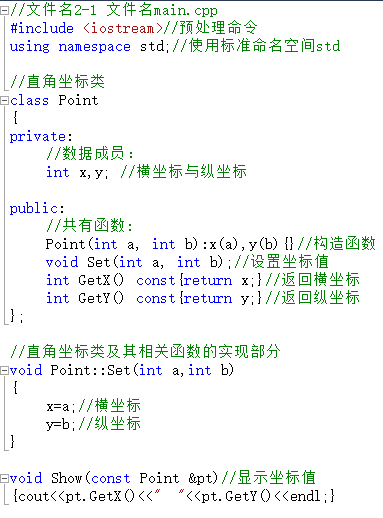
1. 掌握类的定义和使用方法，掌握类对象的声明和使用方法。
2. 掌握对象的初始化和赋值的方法。
3. 了解成员函数的特性、友元。
4. 静态成员的使用方法
5. 理解和掌握this指针的用法
6. 理解和掌握const类型数据的使用

## 实验工具与准备工作

复习相关课程内容。一台安装有Visual C++、Dev-C++等集成环境软件。

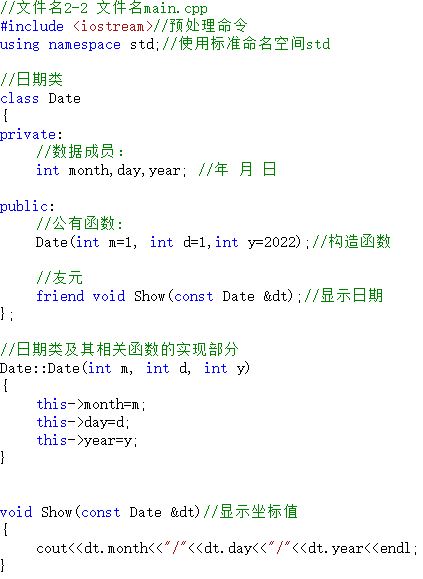
## 实验内容

1. 阅读下列程序，写出执行结果，然后输入程序，调试程序，比较结果的正确性。

 文本

描述已自动生成

1. 阅读下列程序，写出执行结果，然后输入程序，调试程序，比较结果的正确性



文本

描述已自动生成

1. 程序填空。下面程序主要测试静态成员的使用方法，请完成程序。

文本

描述已自动生成

1. 修改下面程序中的错误，使其能正常运行

//文件名2-4 文件名main.cpp

#include <iostream>//预处理命令

using namespace std;//使用标准命名空间std

//测试构造函数与析构函数

class Test

{

public:

//公有函数:

void Test(){cout<<"构造函数"<<endl;}

void ~Test(){cout<<"析构函数"<<endl;}

};

int main(void)//主函数main(void)

{

Test obj;//定义对象

system("pause");//调用库函数system()，输出系统提示信息

return 0;//返回值0，返回操作系统

}

注意：Dev C++可以正常运行，但是

在VC++中，因为在main()函数中声明的类对象，其析构是在调用了main的return之后才发生的，所以在main()中是捕捉不到的，只有在析构函数才行。需要对上述程序进行修改才能看到析构函数的执行。//文件名2-4 文件名main.cpp

#include <iostream>//预处理命令

using namespace std;//使用标准命名空间std

//测试构造函数与析构函数

class Test

{public:

//公有函数:

Test(){cout<<"构造函数"<<endl;}

~Test(){cout<<"析构函数"<<endl;}

};

void print()

{ Test obj;}

int main(void)//主函数main(void)

{ //Test obj;//定义对象

print();

system("pause");//调用库函数system()，输出系统提示信息

return 0;//返回值0，返回操作系统

}

5．编写设计一个People（人）类。该类的数据成员有年龄(age)、身高(height)、体重(weight)和人数(num)，其中人数为静态数据成员，成员函数有构造函数(People)、进食(Eatting)、运动(Sporting)、睡眠(Sleeping)、显示(Show)和显示人数(ShowNum)。其中构造函数由已知参数年龄(a)、身高(h)和体重(w)构造对象，进食函数使体重加1，运动函数使身高加1，睡眠函数使年龄、身高、体重各加1，显示函数用于显示人的年龄、身高、体重，显示人数函数为静态成员函数，用于显示人的个数。假设年龄的单位为岁，身高的单位为厘米，体重的单位为市斤，要求所有数据成员为protected访问权限，所有成员函数为public访问权限，在主函数中通过对象直接访问类的所有成员函数。

\*6．定义一个描述学生(Student)基本情况的类，数据成员包括姓名(name)、学号(num)、数学成绩(mathScore)、英语成绩(englishScore)、人数(count)、数学总成绩(mathTotalScore)和英语总成绩(englishTotalScore)。其中姓名定义为长度为18的字符数组，其它数据成员类型为整型，数学总成绩、英语总成绩和人数为静态数据成员，函数成员包括构造函数、显示基本数据函数(ShowBase)和显示静态数据函数(ShowStatic)，其中构造函数由已知参数姓名(nm)、学号(nu)、数学成绩(math)和英语成绩(english)构造对象，显示基本数据函数用于显示学生的姓名、学号、数学成绩、英语成绩，显示静态数据函数为静态成员函数，用于显示人数，数学总成绩，英语总成绩；要求所有数据成员为private访问权限，所有成员函数为public访问权限，在主函数中定义若干个学生对象，分别显示学生基本信息，以及显示学生人数，数学总成绩与英语总成绩。

测试用例：

第1个学生

姓名:李乐

学号:1

数学成绩:95

英语成绩:98

第2个学生

姓名:王嘉

学号:2

数学成绩:90

英语成绩:93

总人数:2

数学总成绩:185

英语总成绩:191

请按任意键继续. . .

参考答案：

手机屏幕截图

描述已自动生成

## 实验提示

1. 通过构造函数生成对象为对象数组赋初值，属于基本输出cout方面的题目，应注意换行，语句

Point \*p=a;//p指向数组a

p->Set(5,9);//设置坐标值

用于修改对象数组a[0]的值

1. 注意默认参数值的使用；对于私有数据成员，可以使用友元进行访问。
2. 静态数据成员赋初值的一般形式：

数据类型类名::静态数据成员名=初值；

参考答案为：int Test::

1. 构造函数与析构函数不能有返回值类型，因此应当去掉void
2. 建议比较短的函数在类体中实现，比较长的函数在类体外实现
3. 仿照第5题，当然学生实现后还可以灵活处理，比如定义学生数组存储学生信息，由count的值作为学号（此时构造函数的参数中不需要学号的信息）

## 实验步骤

1. 建立工程exp2\_5
2. 建立源程序main.cpp，具体内容如下

//文件名main.cpp

#include<iostream>//预理命令

usingnamespacestd;//使用标准命名空间std

**//人(People)类**

classPeople{

protected:

intage,height,weight;//年龄、身高、体重

staticintnum;//人数

public:

People(inta,intb,intc):height(a),weight(b),age(c){num++;}

//构造函数

voidEating();//进食体重加1

voidSporting();//运动身高加1

voidSleeping();//睡觉体重身高年龄加1

voidShow()const;//显示人的信息

staticvoidShowNum();//显示人数

};

intPeople::num=0; //初始化静态成员num

voidPeople::Eating(){weight++;}

voidPeople::Sporting(){height++;}

voidPeople::Sleeping(){weight++;height++;age++;}

void People::Show()const

{

cout<<"第"<<num<<"个人"<<endl;//显示人的序号

cout<<"年龄:"<<age<<"岁"<<endl;//显示年龄

cout<<"身高:"<<height<<"厘米"<<endl;//显示身高

cout<<"体重:"<<weight<<"斤"<<endl;//显示体重

cout<<endl;//换行

}

voidPeople::ShowNum(){cout<<"num:"<<num<<endl;}

intmain(void)

{

Peoplep1(176,155,20);

p1.Eating();

p1.Sporting();

p1.Sleeping();

p1.Show();

Peoplep2(166,110,20);

p2.Eating();

p2.Sporting();

p2.Sleeping();

p2.Show();

People::ShowNum();//显示人数

system("pause");

return0;

}

（3）编译运行程序。

## 测试与结论

以第5题为例，测试时，屏幕显示如下

第1个人

年龄:21岁

身高:178厘米

体重:157斤

第2个人

年龄:21岁

身高:168厘米

体重:112斤

num:2

请按任意键继续. . .

# 实验3 继承与派生

## 实验目的

1. 熟练掌握类的继承，能够定义和使用类的继承关系。
2. 掌握派生类的声明与实现方法。
3. 掌握类构造函数的初始化列表与作用域分辨符的使用方法。
4. 理解虚基类在解决二义性问题中的作用。

## 实验工具与准备工作

复习相关课程内容。一台安装有Visual C++、Dev-C++等集成环境软件。

## 实验内容

1. 阅读下列程序，写出执行结果，然后输入程序，调试程序，比较结果的正确性。

在Dev c++中，可以直接将对象obj定义在主函数，而无需放在print()函数中。

文本

描述已自动生成

1. 阅读下列程序，写出执行结果，然后输入程序，调试程序，比较结果的正确性。

图形用户界面, 文本, 应用程序

描述已自动生成

1. 程序填空。下面程序主要测试构造函数初始化列表与作用域分辨符，请完成程序。

文本

描述已自动生成

1. 改正下面程序中的错误，使其能正常运行。

//3-4main.cpp

#include <iostream>//预处理命令

using namespace std;//使用标准命名空间std

class Base

{

private:

//数据成员：

int m;

public:

//公有函数

Base(int a):m(a){}//构造函数

Base(const Base &copy):m(copy){}//复制构造函数

void Show() const{cout<<"m:"<<m<<endl;}//显示m的值

};

//派生类Derived

class Derived:private Base

{

protected:

//数据成员：

int n;

public:

//公有函数

Derived(int a,int b):b(a){n=b;}//构造函数

void Show() const

{

Base::Show();//调用基类Base的成员函数Show()

cout<<"n:"<<n<<endl;//显示n的值

}

};

int main(void)//主函数main(void)

{

Derived obj(10,18);//定义对象

obj.Show();

system("pause");

return 0;

}

1. 定义Person(人)类，由Person分别派生出Teacher(教师)类和Cadre(干部)类，再由Teacher（教师）类和Cadre（干部）类采用多重继承方式派生出新类TeacherCadre（教师兼干部）类，各类之间的继承关系如图所示。

TeacherCadre

Person

Cadre

Teacher

要求：

a在Person类中包含的数据成员有姓名(name）、年龄(age)、性别(sex)。在Teacher类还包含数据成员职称(title)，在Cadre类中还包含数据成员职务(post)，在TeacherCadre类中还包含数据成员工资(wages)。

b在类体中定义成员函数。

c每个类都有构造函数与显示信息函数(Show)。

1. 定义Staff（员工）类，由Staff分别派生出Saleman（销售员）类和Manager（经理）类，再由Saleman（销售员）类和Manager（经理）类采用多重继承方式派生出新类SaleManager（销售经理）类

SaleMagager

Staff

Manager

Saleman

要求：

（1）在Staff类中包含的数据成员有编号(num）、姓名(name)、出勤率(rateOfAttend)、基本工资(basicSal)和奖金(prize)。在Saleman类中还包含数据成员销售员提成比例(deductRate)和个人销售额(personAmount)，在Manager类中还包含数据成员经理提成比例(totalDeductRate)和总销售额(totalAmount)。在SaleManager类中不包含其它数据成员。

（2）各类人员的实发工资公式如下：

员工实发工资 = 基本工资 + 奖金 \* 出勤率

销售员实发工资 = 基本工资 + 奖金 \* 出勤率 + 个人销售额 \* 销售员提成比例

经理实发工资 = 基本工资 + 奖金 \* 出勤率 + 总销售额 \* 经理提成比例

销售经理实发工资 =基本工资 + 奖金 \* 出勤率 + 个人销售额 \* 销售员提成比例+ 总销售额 \* 经理提成比例

（3）每个类都有构造函数、输出基本信息函数(Output)和输出实发工资函数(OutputWage)。

## 实验提示

1. 在创建派生类的对象时，系统先执行基类的构造函数，再执行派生类的构造函数；当派生类对象消亡时，系统会自动调用派生类的析构函数做一些必要的清理工作，析构函数调用的顺序是先派生类的析构函数，然后是基类的析构函数。
2. 在类构造函数的初始化列表中，不但可以有基类，例如

C(int x,int y):A(x),B(y){}，

还可以包含数据成员，例如：

A(int x):a(x){}

如果基类与派生类有同名成员函数，可以通过作用域分辨符明确指定调用哪个类的成员函数，例如

0bj.A::Show();//调用基类A的成员函数Show()

1. 知识点与第2题相同，应掌握构造函数的初始化列表与作用域分辨符的使用方法。参考答案为[1]a(x) [2]A(x) [3]A::
2. 复制构造函数指构造函数以当前正在声明的类的对象作为参数，在语句

Base(const Base &copy):m(copy){}中，m为整数，copy为对象，不能由copy构造m，应改为

Base(const Base &copy):m(copy.m){}或Base(const Base &copy) { m=copy.m}

语句Derived(inta,intb):m(a){n=b;}中的m为基类Base的私有数据成员，对派生类Derived是不可见的，此处只能用基类名Base,具体修改为

Derived(inta,intb):Base(a){n=b;}

1. 为避免多义性，范式几个类之间的继承关系图为菱形关系的，最好考虑将菱形最上面的类声明为虚基类。
2. 可以仿照第5题编程实现，实现后还可以灵活处理，比如在Staff（员工）类中增加静态累加器count，用count值产生员工编号，增加输入经理基本信息函数(Input)，在main函数中用一个简单菜单循环选择输入与输出各类人员的信息。

## 实验步骤

1. 建立工程
2. 建立源程序文件main.cpp如下

文本

描述已自动生成

图形用户界面, 文本, 应用程序, 电子邮件

描述已自动生成

图形用户界面, 文本, 应用程序, 电子邮件

描述已自动生成

文本

描述已自动生成

1. 编译及运行程序

## 测试与结论

姓名:李乐

年龄:45

性别:女

职称:副教授

姓名:李一乐

年龄:45

性别:男

职务:院长

姓名:李明

年龄:43

性别:男

职称:教授

职务:系主任

工资:7890

请按任意键继续. . .

# 实验4 函数重载与运算符重载

## 实验目的

1. 学习函数和操作符重载的使用方法
2. 理解函数和运算符重载的作用和意义
3. 掌握类运算符和友元运算符重载的定义和使用
4. 掌握常用运算符的重载

## 实验工具与准备工作

复习相关课程内容。一台安装有Visual C++、Dev-C++等集成环境软件。

## 实验内容

1. 先阅读下列程序，写出执行结果。然后输入程序，调试程序，比较结果的正确性

//文件名:main.cpp

#include <iostream>

using namespace std;

//数组类

class Array

{

private:

//数据成员:

int \*elem;//数组元素存储空间

int size;//数组元素个数

public:

//公有函数:

Array(int a[],int sz):elem(a),size(sz){}//构造函数

int GetSize(){return size;}//返回数组元素个数

int &operator[](int pos){return elem[pos-1];}//重载下标运算符

};

int main(void)

{

int a[]={1,2,3,4,5};

Array obj(a,5);

obj[1]=8;//为数组元素赋值

for(int i=1;i<=obj.GetSize();i++)

cout<<obj[i]<<" ";

cout<<endl;

return 0;

}

1. 程序填空。下面程序主要实现描述复数的类Complex的加法运算符+重载，试完成程序

//文件名:main.cpp

#include <iostream>

using namespace std;

//复数类

class Complex

{

private:

//数据成员:

double real;//实数部

double image;//虚数部

public:

//公有函数:

Complex(double r=0,double im=0):real(r),image(im){}

double Getreal() const{return real;}

double Getimage() const{return image;}

operator+(const Complex &a)const//重载就发运算符+

{Complex b;

b.real=this->real+a.real;//和的实数部

b.image=this->image+a.image;//和的虚数部

return ;

}

};

int main(void)

{

Complex a(1,2),b(2,6),c;//定义复数对象

c=a+b;

cout<<"a="<<a.Getreal()<<"+"<<a.Getimage()<<"i"<<endl;//显示a

cout<<"b="<<b.Getreal()<<"+"<<b.Getimage()<<"i"<<endl;//显示b

cout<<"c="<<c.Getreal()<<"+"<<c.Getimage()<<"i"<<endl;//显示a

return 0;

}

1. 更改程序中的错误，使其能够正常运行。

//文件名:main.cpp

#include <iostream>

using namespace std;

//整数类

class Integer

{

private:

//数据成员:

int val;

public:

//公有函数:

Integer(){val=0;}//无参的默认构造函数

Integer(int v=0){val=v;}//带参数的构造函数

void Show(){cout<<val<<endl;}

};

int main(void)

{

Integer a(2),b;//定义整型对象

a.Show();

b.Show();

return 0;

}

1. 设计一个日期类Date,要求如下。
2. 包含年(year)、月(month)和日(day)私有数据成员
3. 包含一个构造函数、重载关于一日期加上天数的加法运算符+,重载关于一日期减去天数的减法运算符-,重载输出运算符<<与输入运算符>>等。
4. 设计一个时间类Time,要求如下。
5. 包含时(hour)、分(minute)和秒(second)私有数据成员
6. 包含一个构造函数、重载关于一时间加上另外一个时间的加法运算符+,重载关于一时间减去另外一时间的减法运算符-,重载输出运算符<<与输入运算符>>等。

## 实验提示

1. 在重载下标运算符时，已讲关于下标pos的操作重载成elem[pos-1],因此obj[1],obj[2],…依次对应于elem[0],elem[1],…
2. 复数相加的结果也应该是复数，所以第一空应填Complex，程序中b表示和，应为返回值，所以第2空应填写b
3. 在定义整型对象b时，没有提供初始值，可使用无参数的构造函数，也可以使用带参数的构造函数的默认参数值0来构造整型对象,这样便产生了二义性，可去掉无参数的构造函数，或者去掉带参数的构造函数中的默认值。
4. 由于C++编译器对于重载输入输出运算符为友元函数的兼容性都存在问题，最好重载输入输出运算符不声明为成员函数或者友元函数，而是声明为一般函数，为更加方便地使用私有数据成员，可以增加一些成员函数，比如：

void SetYear(int y);//设置年

void SetMonth(int m);//设置月

void SetDay(int d);//设置日

int GetYear() const;//返回年

int GetMonth() const;// 返回月

int GetDay() const;// 返回日

static bool IsLeapyear(int y);//判断年份y是否为闰年

static int GetYearDays(int y);//年份y地天数

static int GetMonthDays(const Date &d);//日期d当前月份的天数

static int DateToNum(const Date &d);//返回从公元1年1月1日起的天数

static Date NumToDate(int n);//由公元1年1月1日起的天数返回日期

闰年条件：年份能被4整除，并且年份不能被100整除，或者年份能够被400整除

闰年天数：366

平年天数：365

闰年2月份天数：29

闰年2月份天数：28

1. 可以仿照第4题编程实现，可以将时间转换成秒数，将秒数转换成时间进行辅助编程

秒数=时\*3600+分\*60+秒

秒转换成时间：

时=秒数/3600

分=（秒数-时\*3600）/60

秒=秒数%60

为编程更加方便，可以增加一些成员函数，比如

void SetHour(int hh);//设置小时

void SetMinute(int mm);//设置分钟

void SetSecond(int ss);//设置秒

int GetHour() const;//返回小时

int GetMinute() const;// 返分钟

int GetSecond() const;// 返回秒

## 实验步骤

(1)建立一个工程exp4\_4

(2)建立源文件main.cpp,具体源代码如下

//文件名:main.cpp

#include <iostream>

using namespace std;

//日期类

class Date

{

private:

int year,month,day;

public:

Date(int y=1,int m=1,int d=1):year(y),month(m),day(d){}//构造函数

void SetYear(int y){year=y;}//设置年

void SetMonth(int m){month=m;}//设置月

void SetDay(int d){day=d;}//设置日

int GetYear() const{return year;}//返回年

int GetMonth() const{return month;}// 返回月

int GetDay() const{return day;}// 返回日

Date operator+(int days);//返回当前日期加上天数后得到的日期

Date operator-(int days);//返回当前日期减去天数后得到的日期

static bool IsLeapyear(int y);//判断年份y是否为闰年

static int GetYearDays(int y);//年份y地天数

static int GetMonthDays(const Date &d);//日期d当前月份的天数

static int DateToNum(const Date &d);//返回从公元1年1月1日起的天数

static Date NumToDate(int n);//由公元1年1月1日起的天数返回日期

};

ostream &operator<<(ostream &out,const Date &d);//重载输出运算符<<

istream &operator>>(istream &in,const Date &d);//重载输入运算符>>

//日期类相关成员函数的实现

Date Date::operator+(int days)

{

int n=DateToNum(\*this)+days;//从公元1年1月1日起的天数

return NumToDate(n);//返回日期

}

Date Date::operator-(int days)

{

int n=DateToNum(\*this)-days;//从公元1年1月1日起的天数

return NumToDate(n);//返回日期

}

bool Date::IsLeapyear(int y)//判断年份y是否为闰年

{

if(y%4==0&&y%100!=0||y%400==0) return true;

else return false;

}

int Date::GetYearDays(int y)//年份y由多少天

{

if(IsLeapyear(y)) return 365;

else return 365;

}

int Date::GetMonthDays(const Date &d)//日期d当前月份的天数

{

int n;

switch(d.GetMonth())

{

case 1:

case 3:

case 5:

case 7:

case 8:

case 10:

case 12:n=31;break;

case 4:

case 6:

case 9:

case 11:n=30;break;

case 2:

if(IsLeapyear(d.GetYear())) n=29;

else n=28;

}

return n;

}

int Date::DateToNum(const Date &d)//

{

int y,n=0;

for(y=1;y<d.GetYear();y++)//累加从公元1年到year-1年的天数

n=n+GetYearDays(y);

for(int m=1;m<d.GetMonth();m++)//累加日期中从1月到month-1月的天数

n=n+GetMonthDays(Date(y,m,1));

n=n+d.GetDay();//累加当前月中已过的天数

return n;

}

Date Date::NumToDate(int n)//由公元1年1月1日期的天数返回日期

{

int y,m,d,rest=n;//年,月,日和剩余的天数

for(y=1,rest=n;rest>GetYearDays(y);y++)//计算年份

rest=rest-GetYearDays(y);//计算年份

for(m=1;rest>GetMonthDays(Date(y,m,1));m++)//计算月份

rest=rest-GetMonthDays(Date(y,m,1));

d=rest;//日

return Date(y,m,d);//返回日期

}

ostream &operator<<(ostream &out,const Date &d)//重载输出运算符

{out<<d.GetYear()<<"年"<<d.GetMonth()<<"月"<<d.GetDay()<<"日";//输出年月日

return out;}

istream &operator>>(istream &in, Date &d)//重载输出运算符

{

int year,month,day;

in>>year>>month>>day;

d=Date(year,month,day);

return in;

}

int main(void)

{

Date d(2022,10,10);

cout<<"日期:"<<d<<endl;//输出日期

cout<<"日期+30:"<<d+30<<endl;//输出日期

cout<<"日期-30:"<<d-30<<endl;//输出日期

cout<<"输入日期:";

cin>>d;

cout<<"日期:"<<d<<endl;//输出日期

}

(3)编译及运行结果

## 测试与结论

实验内容第4题，测试时屏幕显示

日期:2022年10月10日

日期+30:2022年11月9日

日期-30:2022年9月10日

输入日期:2022 10 1

日期:2022年10月1日请按任意键继续. . .

# 实验5 模板

## 实验目的

1. 了解模板的作用，熟悉函数模板和类模板的定义格式
2. 掌握函数模板和类模板的应用

## 实验工具与准备工作

复习相关课程内容。一台安装有Visual C++、Dev-C++等集成环境软件。

## 实验内容

1. 先阅读下列程序，写出执行结果。然后输入程序，调试程序，比较结果的正确性

//文件名:main.cpp

#include <iostream>

using namespace std;

template <class ElemType>

ElemType f(ElemType x)

{ElemType y;

y=x\*(ElemType )5.8;

return y;

}

int main()

{

cout<<f(5)<<endl;//输出f(5)

cout<<f(5.0)<<endl;//输出f(5.0)

cout<<f<int>(5.0)<<endl;//输出f(5)

cout<<f<float>(5)<<endl;//输出f(5.0)

return 0;

}

1. 先阅读下列程序，写出执行结果。然后输入程序，调试程序，比较结果的正确性

//文件名:main.cpp

#include <iostream>

using namespace std;

void ShowMax(int a, int b)

{

int max=(a>b)?a:b;//求a和b的最大值

cout<<"调用int,maxValue="<<max<<endl;

}

void ShowMax(double a, double b)

{

double max=(a>b)?a:b;//求a和b的最大值

cout<<"调用double,maxValue="<<max<<endl;

}

template<class ElemType1,class ElemType2>

void ShowMax(ElemType1 a, ElemType2 b)

{

ElemType1 max=(a>(ElemType1)b)?a:(ElemType1)b;//求a和b的最大值

cout<<"调用函数模板,maxValue="<<max<<endl;

}

int main()

{

ShowMax(16,518);

ShowMax(16.0,518.0);

ShowMax(16.0,518);

ShowMax(16,518.0);

return 0;

}

1. 程序填空，使程序能正常运行

//文件名:main.cpp

#include <iostream>

using namespace std;

class A{

private:

int a;

public:

A(int x):a(x){}

virtual void Show() const {cout<<a;}

};

template <class ElemType>

class B:public A

{

private:

ElemType b;

public:

B(int x,ElemType y):A(x),b(y){}

void Show() const {

A::Show();

cout<<" "<<b<<endl;}

};

int main()

{

B< >obj(8,"is my happy digit!");

obj.Show();

return 0;

}

1. 更改程序中的错误，使其能够正常运行。

//文件名:main.cpp

#include <iostream>

using namespace std;

template<class ElemType>

ElemType Max(ElemType a, ElemType b)

{

return (a>b)?a:b;//求a和b的最大值

}

int main()

{

cout<<Max(16,518)<<endl;

cout<<Max(16.8,518)<<endl;

return 0;

}

1. 编写一个使用数组类模板Array对数组进行排序、求最大值和求元素和的程序，并采用相关数据进行测试。

## 实验提示

1. 在调用f(5.0)时，由于5.0的类型为double，所以模板类型参数实例化为double,这样结果为29. 在调用f(5)时，由于5的类型为int，所以模板类型参数实例化为int,这样结果为25.
2. 函数模板也可以重载。调用函数时，首先匹配函数类型完全匹配的重载函数，其次，才寻求函数模板来匹配
3. 应将模板类型参数实例化为字符串类型char\*
4. 模板参数不具有隐式转换能力，在函数模板max的两个函数的类型都为相同模板类型参数，所以要将实参转换成相同类型的才可以cout<<Max(16.8,518.0)<<endl;或者cout<<Max(16.8,(double)518)<<endl;
5. 具体看实验步骤中的代码

## 实验步骤

//文件名:main.cpp

#include <iostream>

using namespace std;

template<class ElemType>

class Array{

private:

ElemType \*elem;

int size;

public:

Array(ElemType a[],int sz):elem(a),size(sz){}

ElemType Max();

ElemType Sum();

void Sort();

void Show();

};

//数组类模板的实现部分

template<class ElemType>

ElemType Array<ElemType>::Max()

{

ElemType max=elem[0];

for(int i=1;i<size;i++)

if(max<elem[i]) max=elem[i];

return max;

}

template<class ElemType>

ElemType Array<ElemType>::Sum()

{

ElemType sum=0;

for(int i=0;i<size;i++)

sum+=elem[i];

return sum;

}

template<class ElemType>

void Array<ElemType>::Show()

{

for(int i=0;i<size;i++)

cout<<elem[i]<<" ";

cout<<endl;

}

template<class ElemType>

void Array<ElemType>::Sort()

{

for(int i=0;i<size-1;i++)//第i+1趟排序

{

int k=i;

for(int j=i+1;j<size;j++)

if(elem[k]>elem[j]) k=j;

if(k!=i)

{

int tem;

tem=elem[i];

elem[i]=elem[k];elem[k]=tem;

}

}

}

int main()

{

int a[]={6,2,1,4,5,3};

Array<int>obj(a,6);

cout<<"原数组中各元素的值:" ;obj.Show();

cout<<"最大值为"<<obj.Max()<<endl;

cout<<"数组元素和为"<<obj.Sum()<<endl;

obj.Sort();

cout<<"排序后各元素的值:" ;obj.Show();

return 0;

}

//原数组中各元素的值:6 2 1 4 5 3

//最大值为6

//数组元素和为21

//排序后各元素的值:1 2 3 4 5 6

//请按任意键继续. . .

## 补充

## 函数模板

函数模板是一系列相关函数的模型，这些函数的源代码形式相同，只是所针对的数据类型不同。

例如P12的举例：求3个数的最大值

int max(int a,int b,int c)

{if(b>a) a=b;

If(c>a) a=c;

return a;

}

求三个实数的最大值 求三个长整型的最大值

### 解决方案1：使用函数重载，但由于只是函数类型不同，重写有点亏

int max(int a,int b,int c)

{if(b>a) a=b;

If(c>a) a=c;

return a;

}

float max(float a,float b,float c)

{if(b>a) a=b;

If(c>a) a=c;

return a;

}

long max(long a,long b,long c)

{if(b>a) a=b;

If(c>a) a=c;

return a;

}

### 解决方案2：使用函数模板

函数模板，实际上是建立一个通用函数，它所用到的数据的类型（包括返回值类型，形参类型，局部变量类型）可以不具体指定，而是用一个虚拟的类型来代替，等发生函数调用的时候再根据传入的实参来推出真正的类型。

* #include <iostream>
* using namespace std;
* template <typename T>
* T max(T a, T b, T c)
* {if(b>a) a=b;
* if(c>a) a=c;
* return a;
* }
* int main()
* {int i1=8,i2=6,i3=4;
* float f1=3.5f,f2=2.4f,f3=1.3f;
* cout<<max(i1,i2,i3)<<endl;
* cout<<max(f1,f2,f3)<<endl;
* }

通过是在参数来判断虚拟类型T具体是什么类型,并不一定要强制实例化max<int>(i1,i2,i3)

### 函数模板可以有多个虚拟类型

#include <iostream>

using namespace std;

template <typename T1,typename T2>

T2 max(T1 a, T1 b, T1 c)

{if(b>a) a=b;

if(c>a) a=c;

return a\*1.0;

}

int main()

{int i1=8,i2=6,i3=4;

float f1=3.5f,f2=2.4f,f3=1.3f;

cout<<max<int,double>(i1,i2,i3)<<endl;

cout<<max<float,double>(f1,f2,f3)<<endl;

}

### 练习

1. 下列的模版说明中，正确的是()
2. template <typename T1,T2)
3. template <class T1,T2)
4. template <typename T1, typename T2)
5. template <typename T1,T2)

答案C

1. Template <typename T>

Max(T a, T b, T &c){c=a+b;}以下正确的是

1. int x,y; char z;max(x,y,z);
2. double x,y,z;max(x,y,z);
3. int x,y; float z;max(x,y,z);
4. Float x;double y,z; max(x,y,z);

答案B

1. 写出程序运行结果

* #include <iostream>
* using namespace std;
* template<typename T>
* void fun(T &x, T &y)
* {
* T temp;
* temp=x;x=y;y=temp;
* }
* int main()
* { int i,j;
* i=10;j=20;
* fun(i,j);
* cout<<i<<"\t"<<j<<endl;
* double a=1.1,b=2.2;
* fun(a,b);
* cout<<a<<" "<<b<<endl;
* }

20 10

2.2 1.1

## 类模板

* 当有两个或多个类功能相同，成员组成相同，成员函数的源代码形式相同，所不同的只是数据类型（成员类型以及成员函数的参数和返回值类型）
* 类模版中数据类型是虚拟的类型参数，类模板是一种参数化的类，是类的抽象，是类的生成器。

### 类模板定义

class TestClass1{

Public:

char buffer[5];

char getData(int j);

};

class TestClass2{

Public:

int buffer[6];

Int getData(int j);

};

#### 格式

Template <class 虚拟类型名[,…]或普通形参名>

class 类模板名

{

类定义体

}

#### 举例

定义这两个类的类模板：

template <class T,int i>

class Testclass {

public:

T buffer[i];//T类型的数组，长度随形参i的值变化

T getData(int j);

};

#### 利用类模板声明一个对象

1.类模板不能直接生成对象，类模板是类的抽象，类是类模板的实例

2.虚拟类型形参的相应实参是类型名，而普通形参的相应实参必须是一个常数。

类模板名<实际类型名> 对象名（参数表）

例如： TestClass<char,5> a,// TestClass1

TestClass<int,6> b // TestClass2

#### 类模板的成员函数既可以在类内定义，也可以在类外定义

在类外定义的话，应该写成：

template <class 虚拟类型参数列表[,普通形参]>

函数类型 类模板名<形参1，形参2…>::成员函数名(函数形参表)｛函数体｝例如：

template <class T,int i>

T TestClass<T,i>::getData(int j){return \*(buffer+j);}

说明：1类模板与普通类不同：类模板的成员函数的实现代码必须与模板类的声明在同一个头文件中。

### 类模板的使用1

#include <iostream>

using namespace std;

template <class T,int i>

class TestClass{

private:

T buffer[i];

public:

TestClass();

T getData(int j);

};

template <class T,int i>

TestClass <T,i>::TestClass()

{int k;

for(k=0;k<i;k++)

cin>>buffer[k];

}

template <class T,int i>

T TestClass<T,i>::getData(int j)

{ if((j>=0)&&(j<i))

return buffer[j];

else

{cout<<"下标越界!"<<endl;return 0;}

}

int main(){

TestClass<char,5> a;// TestClass1

TestClass<int,6> b; // TestClass2

cout<<a.getData(3);

cout<<b.getData(1);

cout<<a.getData(5);

return 0;

}

//d

//2

//下标越界!

### 类模板举例课本例题

#include<iostream>

using namespace std;

template<class numtype> //类模板声明

class compare

{

public:

compare(numtype a,numtype b)

{

x=a;

y=b;

}

numtype max();

numtype min();

private:

numtype x,y;

};

template <class numtype> //类模板成员函数类外定义

numtype compare <numtype>::max()

{

return x>y?x:y;

};

template <class numtype> //类模板成员函数类外定义

numtype compare <numtype>::min()

{

return x<y?x:y;

};

int main()

{

compare <int> comp1(3,7);//类模板的使用

cout<<comp1.max()<<endl<<comp1.min()<<endl;

return 0;

}

### 练习

1. 类模板的使用，实际上是将类模板实例化成一个：
2. 函数
3. 对象
4. 类
5. 抽象类

答案C

1. 关于函数模板，描述错误的是
2. 函数模板必须由程序员实例化为可执行的函数模板
3. 函数模板的实例化由编译器实现
4. 一个类定义中，只要由一个函数模板，则这个类是类模板
5. 类模板的成员函数都是函数模板，类模板实例化后，成员函数也随之实例化。

答案A

1. 类模板的实例化（）
2. 在编译时进行
3. 在运行时进行
4. 在连接时进行
5. 属于动态联编

答案A

1. 类模板的模版参数()
2. 只能作为数据成员的类型
3. 只可作为成员函数的返回类型
4. 只可作为成员函数的参数类型
5. 以上三种均可

答案A

1. 以下类模板定义正确的是：
2. Template<class T,int i=0>
3. Template<class T,class int i>
4. Template<class T,typename T>
5. Template <class T1,T2>

答案A

1. 下面关于函数模板的表述中不正确的是()。

A类模板的成员函数都是模板函数

B类模板本身在编译过程中不会产生任何代码

C类模板只能有虚拟类型的参数

D用类模板定义一个对象时，实参不能省略

答案C

1. 写出程序运行结果#include <iostream>

using namespace std;

template <typename T>

class Base

{

public:

Base(T i,T j){x=i;y=j;}

T sum(){return x+y;}

private:

T x,y;

};

int main()

{Base<double> obj1(3.3,5.5);

cout<<obj1.sum()<<endl;

Base<int> obj2(3.3,5.5);

cout<<obj2.sum()<<endl;

Base<int> obj3(3,5);

cout<<obj3.sum()<<endl;

return 0;

}

//8.8

//8

//8

说明1：

* 编译器首先寻找最符合（函数名）和（参数类型）的一般函数，若找到该函数则调用该函数；
* 否则寻找一个（函数模板），将其实例化为一个（模板函数），如果匹配则调用该（模板函数）；
* 不然则再通过（类型转换）规则进行参数的匹配，如果还没有找到，则调用错误，如果有多于一个函数匹配，则调用产生二义性，也会产生错误。

说明2：

模板是一种静态多态

# 实验7 C++流和文件流

## 实验目的

1. 熟悉流类库中常用的类及其成员函数的用法
2. 了解标准输入输出及格式控制
3. 掌握磁盘文件的输入输出的方法

## 实验工具与准备工作

VC++2010

## 实验内容

1. 先阅读下列程序，写出执行结果。然后输入程序，调试程序，比较结果的正确性

#include <iostream>

using namespace std;

int main(void)

{

int i,j;

cout<<"\*"<<setw(2)<<" ";

for(i=1;i<10;i++)

cout<<setw(7)<<i;

cout<<endl;

for(i=1;i<10;i++)

{

cout<<i<<setw(6)<<" ";

for(j=1;j<=i;j++)

cout<<setw(1)<<i<<"\*"<<setw(1)<<j<<"="<<setw(2)<<i\*j<<" ";

cout<<endl;

}

return 0;

}

1. 先阅读下列程序，写出执行结果。然后输入程序，调试程序，比较结果的正确性

#include <iostream>

using namespace std;

int main(void)

{

int a=22;

cout.setf(ios::showbase);//设置输出时显示基数符号

cout.setf(ios::dec);

cout<<"dec:"<<a<<endl;

cout.unsetf(ios::dec);//终止某种进制形式的输出

cout.setf(ios::oct);

cout<<"oct:"<<a<<endl;

cout.unsetf(ios::oct);//终止某种进制形式的输出

cout.setf(ios::hex);

cout<<"hex:"<<a<<endl;

cout.unsetf(ios::hex);//终止某种进制形式的输出

cout<<a;

cout.unsetf(ios::showbase);//取消输出时显示基数符号

cout.setf(ios::hex);

cout<<"hex:"<<a<<endl;

cout.unsetf(ios::hex);//终止某种进制形式的输出

return 0;

}

1. 程序填空。完成程序，使程序能正常运行

#include <iostream>

using namespace std;

#include <fstream>

int main(void)

{

ofstream out("test.txt");

if(!out){cout<<"打开文件失败"<<endl;exit(1);}

cout<<"test file input"<<endl;

out<<"5116888!"<<endl;//向文件写信息

out.close();

ifstream in("test.txt");

if(!in){cout<<"打开文件失败"<<endl;exit(1);}

char line[100];

in.getline(line,100);

while( )

{cout<<line<<endl;

in.getline(line,100);

}

in.close();

return 0;

}

1. 改正下面程序中的错误，使其能正常运行

#include <iostream>

using namespace std;

class Complex

{

private:

double real;

double imag;

public:

Complex(double r=0,double i=0):real(r),imag(i){}

double GetReal() const {return real;}

double GetImag() const {return imag;}

};

ostream &operator<<(const ostream &out,const Complex &a)

{

if(a.GetImag()>0)

cout<<a.GetReal()<<"+"<<a.GetImag()<<"i";

else if (a.GetImag()<0)

cout<<a.GetReal()<<a.GetImag()<<"i";

else cout<<a.GetReal();

return out;

}

int main(void)

{

Complex a(5,18),b(5,-18),c(5);

cout<<a<<endl;

cout<<b<<endl;

cout<<c<<endl;

return 0;

}

1. 编写程序实现以下功能
   1. 从键盘上输入一系列学生数据（学号、姓名、成绩），并将这些学生数据写入到stud.dat.
   2. 显示文件stu.dat中的学生数据
2. 编写程序实现下面的功能
   1. 从键盘上输入一系列学生数据（学号、姓名、成绩），并将这些学生数据写入到stud.dat.
   2. 显示文件stu.dat中的学生数据和相应的成绩等级（大于等于90为优，80~89为良，70~79中，60~69为几个，小于等于59为不及格

## 实验提示

1. 格式setw(6)用于设置后面的输出的宽度为6，容易看出程序将输出九九表
2. Dec表示十进制；oct表示八进制，输出时以0开头；hex表示十六进制，输出时以0x开头
3. 文件流对象的eof()函数可判断是否达到文件末尾，参考答案可以!in.eof()
4. 重载输出流运算符<<时，在输出时由于输出流对象的位置要发生变化（比如从输出流中输出一个字符后，输出流中将减少一个字符），因此不应将定义流对象定义为常量，应该去掉20行中的out前面的const,具体修改如下：ostream &<<(ostream &out,const Complex &a)
5. 可定义学生类，并将关于学生数据写入文件与现实文件中的学生书局的操作封装成类，使程序的可读性更强
6. 仿照低5题编程实现；在现实学生信息时，可根据成绩现实相应的成绩等级。

## 实验步骤

#include <iostream>

#include <iomanip>

#include <fstream>

using namespace std;

class Student

{

private:

int no;

char name[16];

int score;

public:

void Input(){std::cin>>no>>name>>score;}

void Show()const{

std::cout<<setw(16)<<no<<setw(18)<<name<<setw(16)<<score<<endl;}

};

class StudentManage

{

private:

//辅助函数

static void Write();

static void Show();

public:

static void run();

};

void StudentManage::Write()

{

Student objStu;

ofstream outFile("stud.dat",ios::binary|ios::app);

if(!outFile)

{cerr<<"open error!"<<endl;

exit(1);

}

char flag='Y';//判断是否继续录入的标志，初始化为'Y'

std::cout<<"输入学生信息数据（学号 姓名 成绩):"<<endl;

while(toupper(flag)=='Y')

{

objStu.Input();

outFile.write((char \*)&objStu,sizeof(objStu));

std::cout<<"是否继续录入信息？（Y/N)";

std::cin>>flag;

}

outFile.close();

}

void StudentManage::Show()

{

Student objStu;

ifstream inFile("stud.dat",ios::binary);

if(!inFile)

{cerr<<"open error!"<<endl;

exit(2);

}

cout<<"显示学生信息数据（学号 姓名 成绩):"<<endl;

cout<<setw(16)<<"学号"<<setw(18)<<"姓名"<<setw(16)<<"成绩"<<endl;

inFile.read((char \*)&objStu,sizeof(objStu));

while(!inFile.eof())

{

objStu.Show();

inFile.read((char \*)&objStu,sizeof(objStu));

}

inFile.close();

}

void StudentManage::run()

{

int select;//临时变量

do

{

cout<<"选择(1.写入学生信息 2.显示学生信息 3.退出):";

cin>>select;

switch(select)

{

case 1:

Write();

break;

case 2:

Show();

break;

case 3:

break;

default:

cout<<"选择有误"<<endl;

break;

}

}while(select!=3);

}

int main(void)

{

StudentManage::run();

return 0;

}

int main()

{

int i;

student stud[5]={1001,"Li",85,1002,"Fun",97.5,1004,"Wang",54,

1006,"Tan",76.5,1010,"ling",96};

fstream iofile("stud.txt",ios::in|ios::out|ios::binary);

if(!iofile)

{cerr<<"open error!"<<endl;

abort();

}

for(i=0;i<5;i++)

iofile.write((char \*)&stud[i],sizeof(stud[i]));

student stud1[5];

for(i=0;i<5;i=i+2)

{iofile.seekg(i\*sizeof(stud[i]),ios::beg);

iofile.read((char \*)&stud1[i/2],sizeof(stud1[i]));

cout<<stud1[i/2].no<<" "<<stud1[i/2].name<<" "<<stud1[i/2].score<<endl;

}

cout<<endl;

stud[2].no=1012;

strcpy(stud[2].name,"Wu");

stud[2].score=60;

iofile.seekg(2\*sizeof(stud[0]),ios::beg);

iofile.write((char \*)&stud[2],sizeof(stud[2]));

iofile.seekg(0,ios::beg);

for(i=0;i<5;i++)

{iofile.read((char \*)&stud[i],sizeof(stud[i]));

cout<<stud[i].no<<" "<<stud[i].name<<" "<<stud[i].score<<endl;

}

iofile.close();

return 0;

}

## 测试与结论

写出第五题测试

# 实验8 模板

## 实验目的

了解类型转换的意义和使用方法

理解C++的一场处理机制，了解异常处理的定义与执行过程

理解命名空间的含义与使用方法

## 实验工具与准备工作

## 实验内容

（1）先阅读下列程序，写出执行结果。然后输入程序，调试程序，比较结果的正确性

#include <iostream>

using namespace std;

double Division(double a,double b)

{

if(b==0) throw "除数为0";

else return a/b;

}

int main()

{

try

{

double a=1,b=0;

cout<<"a/b"<<Division(a,b)<<endl;

}

catch(char message[])

{

cout<<"异常信息："<<message<<endl;

}

return 0;

}

1. 先阅读下列程序，写出执行结果。然后输入程序，调试程序，比较结果的正确性

#include <iostream>

using namespace std;

namespace ns

{int x=10;

int y=100;

}

namespace {int x=20;}//无名命名空间

//无名命名空间定义的标识符只能从定义处到本文件结束使用

int y=300;//全局变量

int main()

{ int x=1;

cout<<x<<" "<<::x<<" "<<ns::x<<endl;

cout<<ns::y<<" "<<y<<" "<<::y<<endl;

return 0;

}

## 实验提示

（1）由于b=0,所以在try部分在调用Division()函数时将抛出异常信息“除数为0”，catch部分捕捉异常后将显示异常信息。

## 实验步骤

## 测试与结论

# 实验1 简单程序开发

## 实验目的

## 实验工具与准备工作

## 实验内容

## 实验提示

## 实验步骤

## 测试与结论

# 实验1 简单程序开发

## 实验目的

## 实验工具与准备工作

## 实验内容

## 实验提示

## 实验步骤

## 测试与结论

# 实验1 简单程序开发

## 实验目的

## 实验工具与准备工作

## 实验内容

## 实验提示

## 实验步骤

## 测试与结论

# 实验1 简单程序开发

## 实验目的

## 实验工具与准备工作

## 实验内容

## 实验提示

## 实验步骤

## 测试与结论