实 验 报 告

**课程名称：** 面向对象技术(C++)

**实验项目：** 类与对象

**实验仪器：** 计算机

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **项目** | **报告格式**  **Report format** | **代码质量**  **Code quality** | **注释质量**  **Comment quality** | **逻辑或思想描述**  **Necessitate logical description** | **独创性**  **Originality** | **合计**  **Total** |
| **百分比(%)**  **percentage** | **10** | **25** | **25** | **25** | **15** | **100** |
| 得分（score） |  |  |  |  |  |  |

**系 别：** 计算机系

**专 业：** 计算机大类

**班级姓名：** 计类1805刘科宏

**学 号：** 2018011252

**日 期：** 4月2日

**成 绩：**

**同组成员：**

**指导教师：** 丁 濛

# 实验目的(Objects)

1. 了解编译器的使用方法，掌握利用命令行的方式进行编译和链接；Learn how to use C++ compiler to compile source code and link it to a binary program in a command-line style;
2. 学习引用型变量的使用方法，学会使用const关键字声明只读性变量，学习使用新关键字new和delete分配和删除内存空间；

Learn how to declare and use a variablewith type *reference*;

Learn how to use *const* prefix to declare constants with a specific type;

Learn how to use *new* and *delete* to allocate and release memory spaces from heap.

1. 学习和掌握C++对结构体的扩充；

Understand how C++ extended and modified the definition of the *struct* in C to provide the object-oriented feature.

1. 学习和掌握类的声明、定义方式以及如何声明、使用一个类的对象；验证构造函数、析构函数的调用时机；验证C++对于成员变量和函数的封装及访问控制；  
   Learn how to define a *class* and how to declare and use an object of a class;

Validate the mechanism of *constructor* and *destructor*: when they will be called automatically.

Validate how C++ encapsulates member properties and functions of a class and protects them from being accessed by outside the class anywhere.

1. 学习和掌握类的三种构造函数定义方式；拷贝构造函数的意义，默认拷贝构造函数的执行方式，何时需要显示地定义拷贝构造函数；

Learn how to define three different kinds of constructor function: Copy constructor, constructor without parameters (default constructor) and constructor w/ parameters;

Understand the essentials of a copy constructor, the way how a default copy constructor executes and when you need to define a copy constructor explicitly.

1. 学习和掌握类的对象的使用方法，包括自动分配和动态分配两种方式；

Learn how to declare and use an object of a class, in both dynamical and automatic way.

1. 学习和体会面向对象思想中的抽象概念，即封装数据及对数据的操作；

Understand and practice the concept of abstract in OOP: Combine *data* and operations on *data* together as a new user-defined type.

# 实验内容 (Contents)

1. 配置编译环境，通过命令行方式对源代码进行编译和链接；

Configure the command-line compiling environment in your computer, which includes installing the compiler and configure the environment path. Try to compile a C++ source code and link it to a binary program using command-line.

1. 用指针的形式编写函数，void swap(int \*d1, int \*d2)，该函数实现交换两个整形变量的值；用引用的形式重写该函数，void swap(int &d1, int &d2)；可否使用const修饰上述函数的形参，即d1和d2？如果使用const修饰，会出现什么问题？

Write a function *void swap(int \*, int \*)* to swap the values of the given two parameters; Using *reference* type to rewrite that *swap* function. Can you apply *const* prefix to the two parameters? If you do that, what will happen?

1. 设计 Person 类。该类包括五个成员变量，分别是 姓名 char \*name，性别 char gender，出生年月日 int year, int month, int day。定义该类的三种构造函数，即 Person::Person(), Person::Person(const char \*, char, int, int, int), Person::Person(const Person& other)。定义该类的析构函数 Person::~Person(), 实现必要的内存清理功能。定义该类的公有成员函数 void Person::Print()，实现打印姓名，性别，出生年月日的功能。定义该类的公有成员函数 void Person::setName(const char \*)，实现对姓名的赋值；void Person::setGender(const char)，实现对性别的赋值；void Person::setBirthday(const int , const int , const int)，实现对出生年月日的赋值，需要考虑输入的有效性。  
    编写main函数，声明一个Person类的对象实例，并用自己的姓名、性别、出生年月日初始化该对象。验证析构函数、构造函数的调用时机。通过动态分配的方式实例化一个Person类的对象，并使用拷贝构造函数对该初始化该对象。

Design class Person.

The class will have 5 member variables, which are name (char \*), gender (char), birthday: year, month, day (int, int, int);

Define its three kinds of constructor: copy constructor, constructor without parameters (default constructor) and constructor w/ parameters.

Define its destructor function to do the necessary clean-up.

Define a public member function *void Person::Print()* to print the name, gender, birthday; a public member function *void Person::setName(const char \*)* to modify name, *void Person::setGender(const char)* to modify gender and *void Person::setBirthday(const int , const int , const int)* to modify birthday. You should consider the availability of the input.

Write a main function, define an object of class Person, and initialize it with your name, gender and birthday. Validate the time of the calling of constructor and destroyer. Define another object of Person in dynamical way and initialize it using copy constructor.

1. （附加题）编写函数，实现以字符串形式完成两个正整数的加法：输入为两个字符串，分别代表两个正整数，输出为一个字符串，表示最终结果，即char \*add\_string\_version(const char \*d1, const char \*d2)；编写main函数验证正确性。

例： char \*a = “0001200000320987630”, char \*b = “1”;

char \*result = long\_integer\_addition(a, b);

printf(“%s\n”, result); // 🡺 1200000320987631

(Bonus) Write a function to implement the addition of two positive integers expressed with type char\*, output a char\* expressing the sum. The prototype of that function can be *char \*long\_integer\_addition (const char \*d1, const char \*d2)*. Write a main function to validate your *long\_integer\_addition*.

Example:

char \*a = “0001200000320987630”, char \*b = “1”;

char \*result = long\_integer\_addition(a, b);

printf(“%s\n”, result); // 🡺 1200000320987631

# 实验内容 (Your steps or codes, and Results)

1.

#include <stdio.h>

#include <iostream>

using namespace std;

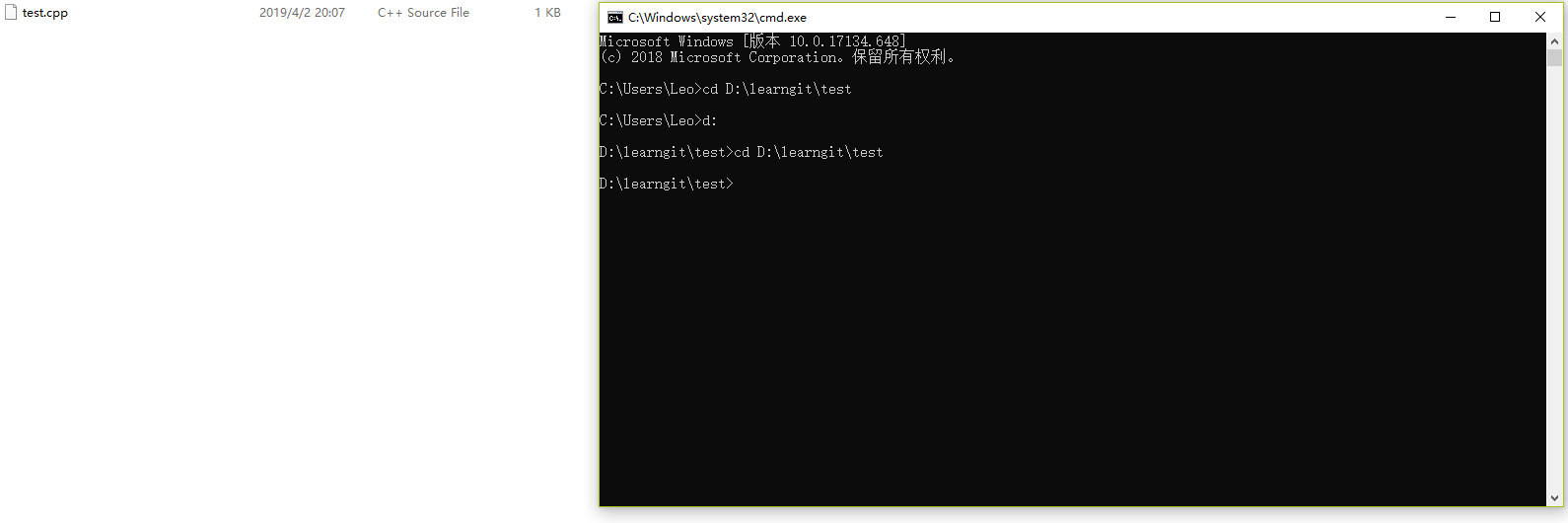
int main()

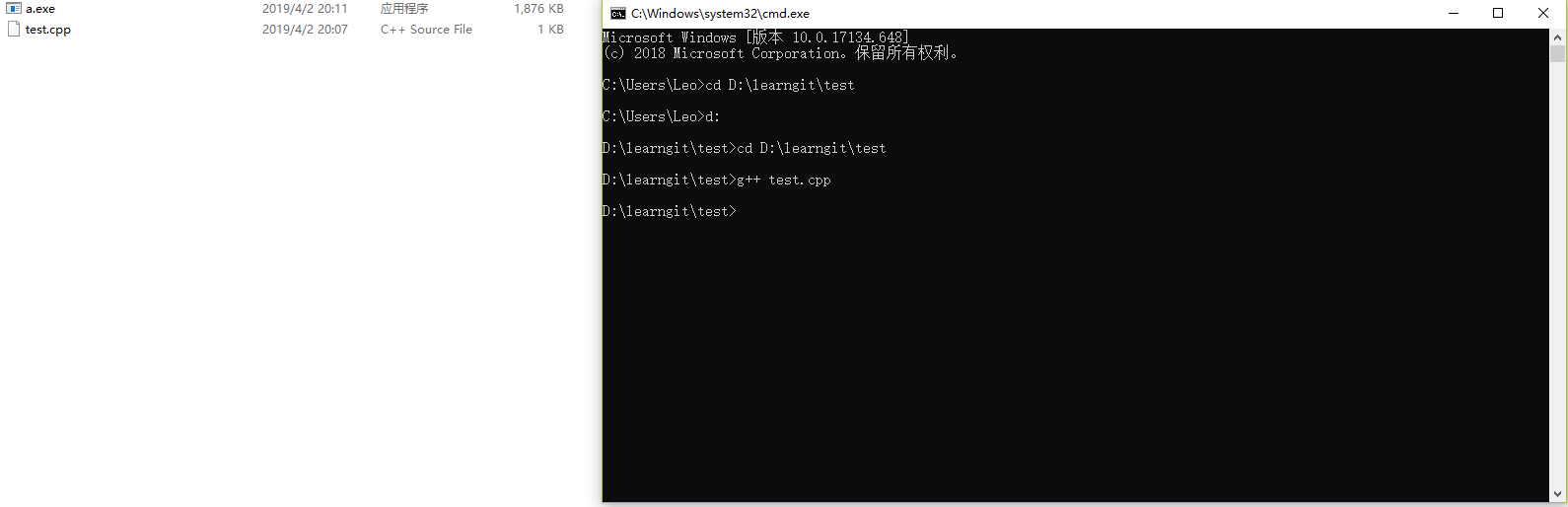
{

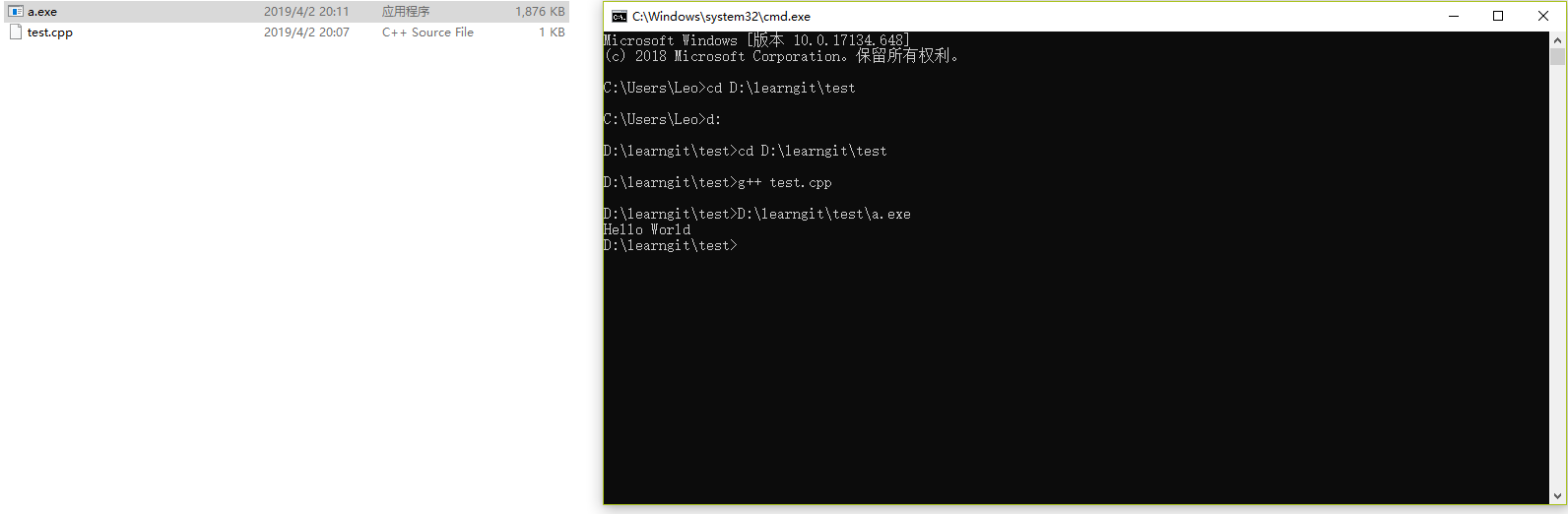
cout << "Hello World";

return 0;

}







2.

#include <stdio.h>

void swap(int \*d1, int \*d2) //第一种方法

{//把主函数中的定义的变量的地址传到函数中，通过交换指针指向的地址的数字来达到目的

int p; //定义一个整型自变量来用于交换两个指针，不能通过定义空指针来交换指针

p = \*d1;

\*d1 = \*d2;

\*d2 = p;

}

void swap(int &d1, int &d2)//第二种方法

{//使用引用符号“&”来把主函数的变量值赋给函数形参，这样如同指针一样，修改形参值也能修改原本变量的值

int p;

p = d1;

d1 = d2;

d2 = p;

}

int main(){

int a, b;

scanf("%d%d", &a, &b);

printf("a:%d, b:%d\n", a, b);

swap (&a, &b);

printf("a:%d, b:%d\n", a, b);

swap (a, b);

printf("a:%d, b:%d\n", a, b);

return 0;

}

不能使用const来修饰形参，因为在函数当中我们需要通过修改形参的值以达到修改主函数变量的目的，若加了const的话，将会无法修改形参，从而无法修改主函数中的变量。

3.

#include <stdio.h>

#include <string.h>

#include <stdlib.h>

class Person{

private: //Person类当中不希望别人直接修改成员所以设为私有

char \*name;

char gender;

int year;

int month;

int day;

public:

Person()//无参数的构造函数，把构造的对象赋上初值

{

name = NULL;

gender = '\0';

year = 1900;

month = 1;

day = 1;

}

Person(const char \*n,const char a,const int y,const int m,const int d)//带有参数的构造函数

{

printf("Person(const char\*, char, int, int, int)\n");

year = y;

month = m;

day = d;

gender = a;

if(n !=NULL)//判断传入的指针\*n是否为空，若为空则不需要新开辟空间

{

name = new char[strlen(n) + 1];//动态分配一个空间来拷贝传入的字符串，strlen()函数不能放空值

strcpy(name,n);

}

}

Person(const Person& other)//拷贝构造函数

{

printf("Person(const Person&)\n");

year = other.year;

month = other.month;

day = other.day;

gender = other.gender;

name = NULL;

if(other.name)//理由同上的带有参数的构造函数，先判断需要拷贝的对象名字是否为空

{

name = new char[strlen(other.name) + 1];

strcpy(name,other.name);

}

}

~Person()//析构函数

{

delete [] name;//用于删除之前对于\*name开出的动态空间，否者在某些情况中，空间会一直堆砌使程序崩溃

printf("~Person()\n");

}

void Print()//这个函数用于一次性输出对象的所有信息

{

printf("name:%s\n",name);

printf("gender:%c\n",gender);

printf("Birthday:%d-%02d-%02d\n",year,month,day);

}

void setName(const char \*n)//这个函数用于输出修改与赋予Person对象的name值

{

if(name)//得先判断name是否已经被赋值过，若是这要删除动态空间，以防空间堆砌

{

delete [] name;

name = NULL;

}

if(n)//理由同的带有参数的构造函数与拷贝构造函数

{

name = new char[strlen(n + 1) + 1];

strcpy(name, n);

}

}

void setGender(const char a)//这个函数用于设置Person对象的性别

{

gender = a;

}

void setBirthday(const int y,const int m,const int d)//这个函数用于设置Person对象的生日

{

year = y;

month = m;

day = d;

}

};

int main(){

char name[100];

int year, month, day;

char gender;

scanf("%s", name);

scanf("%d%d%d", &year, &month, &day);

getchar();//吸收上一个输入完后的回车

scanf("%c", &gender);

Person p(name, gender, year ,month, day);//使用带有参数的构造函数

p.Print();

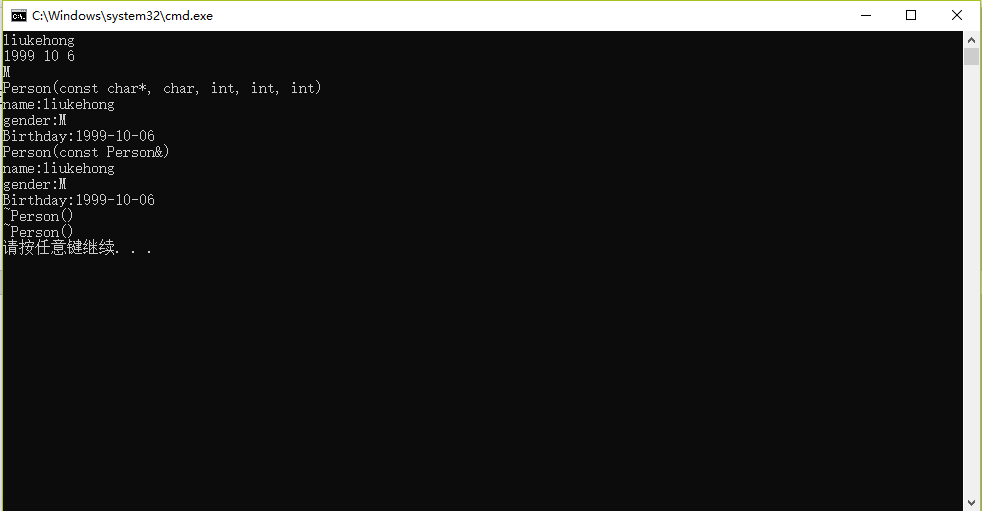
Person \*r = new Person(p);//通过动态分配的方式实例化一个Person类的对象，并使用拷贝构造函数对该初始化该对象

r->Print();

delete r;//清除在主函数中动态分配的空间

return 0;

}



4.

#include <stdio.h>

#include <string.h>

char\* long\_integer\_addition(const char \*a,const char \*b)

{

int max,p,s=0,i,min;

char \*c,\*d;

if(strlen(a) > strlen(b))//判断\*a与\*b那个的位数多，以比较长的字符串为主来加更加方便

{

max = strlen(a);

min = strlen(b);

c = new char[strlen(a) + 1];//由于\*a与\*b指向常量无法改变，需要开辟动态空间来存放，才能修改

strcpy(c,a);

d = new char[strlen(b) + 1];

strcpy(d,b);

}

else

{

max = strlen(b);

min = strlen(a);

c = new char[strlen(b) + 1];

strcpy(c,b);

d = new char[strlen(a) + 1];

strcpy(d,a);

}

while(max)//从个位开始相加，逐渐升高位数

{

int x = 0,y = 0;

if(min > 0)//判断位数少的数是否已经全部加完

x = d[min - 1] - 48;//用较小的数的从左往右第min位的ASCII码减去48即为那一位代表的数字

if(s == 1)//s用于判断上一位是否相加进位，是的话s为1，让x加一后，s回归0

{

x++;

s = 0;//s回归0

}

y = c[max - 1] + x;//让较大的数的从左往右第max位的ASCII码加入x

if(y >= 58)//判断这一位加完了是否要进位

{

y = y - 10;

s = 1;//需要进位，因此把s换为1，提醒下一位相加时要进位

}

c[max - 1] = y;//此时的y值的ASCII码即代表了我们那一位所需要的数，直接让较大数的从左往右第max位等上即可

max--;//向左移一位（即更高位）继续相加

min--;

if(min <= 0&&s==0)

break;//若较少位的数已经加完且不需要进位，则可以直接退出循环

}

if(s == 1)//如果在较多位数都已经加完，仍需要进位，说明最后数字比max多了一位

{

char \*f = new char[strlen(c) + 2];//需要重新开辟出位数多一位的字符串来存放数字

f[0] = '1';//由于进位，所以第一位一定是1

for(i = 0;i <strlen(c);i++)//才开辟的空间从第二位开始都和相加之后的较多位数的数的第一位相同，之后一样

{

f[i+1] = c[i];

}

f[strlen(c) + 1] = '\0'; //让f的最后一位变为/0来结束。

delete [] c;

delete [] d;//释放开辟出来的动态空间

return f;

}

else

{

delete [] d;//释放开辟出来的动态空间

return c;

}

}

int main()

{

char \*a ="0001200000320987630",\*b = "1";

char \*r = long\_integer\_addition(a,b);

printf("%s\n",r);

return 0;

}

这个方法有一个问题便是不知道如何释放开出来的作为返回值的c或f的空间，不知道专门为他们写一个+符号函数重载或许可以解决问题释放f的问题，但还是无法解决释放c的问题，因此打消了这个念头。

# 实验总结 (Conclusion)

这次实验是C++的第一次实验，是对一个月的学习以来的反馈，这一个月真正打破了我对电脑与以往对编程的固有看法。

首先是对编译器的理解的纠正，老师在第一节提问时才弄明白了编译器的实质作用是将将我们自己所写的代码转化为电脑可以识别的电脑语言。

紧接着在之后的课题当中真正认识到了电脑如何存放我们的代码定义的变量与函数，更加明白了解的指针的用法，完全补全了之前对指针知识的空缺与不理解。然后才会明白指针的指向，如何避免问题，接着所有的课程围绕着解决问题与发现问题环环相扣引出了C++开始。

起初在完全没了解C++时，我仅仅认为代码不过是对所给的函数与语法不断的堆砌来实现自己的目的，结果这狠狠的打了我的脸，在实验的过程中，老师教会了我们开始创建类，这便是一切的开始，我就如同发现了新大陆一般，每节课内容都让我新奇不已，这与上学期所学的结构体相似而又完全不同。“类”的出现让我们专门创建一个解决问题的“方法”，我们在里面不断加入新的内容不断改进发现问题，在学习当中也学到了不少新的语法与用法，然而仅仅记笔记，计算机的魅力就在于我们可以随意的实验，在开始有几次没有及时在学习完之后进行实验，第二次上课时，东西就忘了许多，所有这是实验的重要性，它可以加固我们对知识的巩固和强化。

其实发现在学习C++当中发现你得比其他所有的语言必须更加的注重细节，你必须对你开出的每一个变量负责，首先要确保它们的初值，确保它们后面不会因此受到影响，你得保证你对它们的操作必须应对各种情况，这对于C++的魅力与麻烦共存的指针来说尤其得注意，确保它们的指向与空间的开辟释放这些都得面面俱到，少了一个就会很可能使得程序崩溃。同时你对函数也必须得十分注意，它们的返回值类型，参数是否添加引用，是否可以改变，是否需要加const，等等都需要思考全面。代码的优化也是C++常常需要思考的一点，这会在以后更加深入。同时在实验当中我的debug能力也有了很大的提升，懂得看一些编译器所指出的问题，也可以更好的使用调试来发现改进程序，今后会更好的发扬。

才一个多月的学习C++当中就遇到了不少的问题，还好大多都顺利的解决了，这些问题肯定也仅仅只是开始，我也知道后面的问题只会更加的复杂，但是其实内心还是很欣慰当时放弃了去学习听说更加简单的Java，来学习了这个更加复杂但是更加带有挑战与更加接近电脑的语言。学习中领会到了不少C++的魅力，今后会更加努力学习，谢谢老师的教导。