面向对象实验报告

姓名	
班级	
学号	
日期	

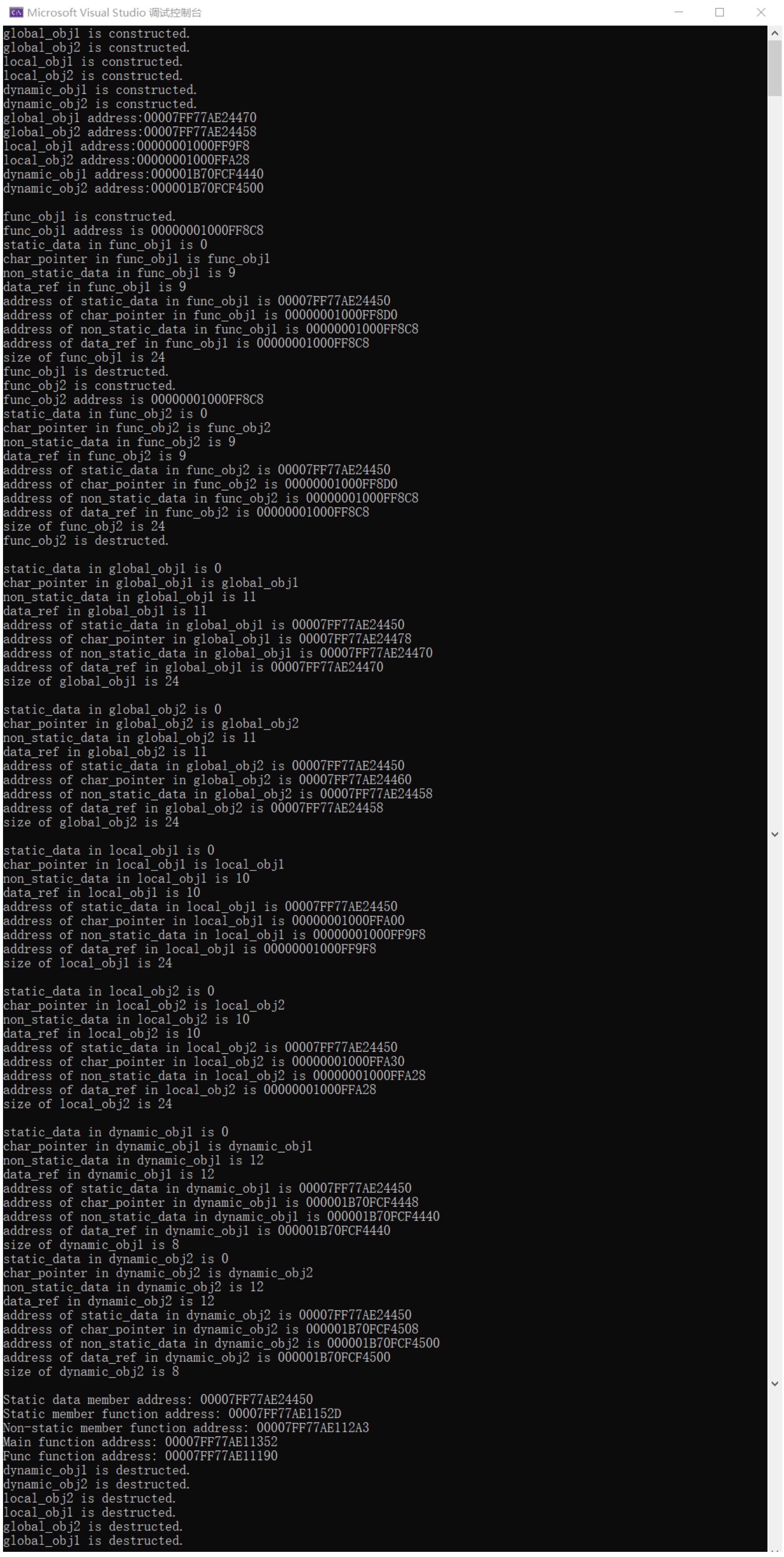
实验内容:
1、定义一个类，其中有静态数据成员、各种类型非静态数据成员（含字符指针），甚至包括引用（可选，不要求），静态和非静态成员函数（含分配空间的构造函数、析构函数）。
2、定义全局对象、main 函数中局部对象、另一个被 main 调用的外部函数 func 中定义局部对象（可以是形参），main 函数中动态创建对象，每种对象至少 2 个。观察、分析各种对象地址。
3、输出对象中各个静态与非静态数据成员的值、地址、对象的存储空间大小等信息。由此理解对象的本质、静态数据成员是本类对象共享一份拷贝等问题。
4、对于上述各种对象，输出静态非静态成员函数地址，以及 main、func 的地址，并分析。

实验代码:
```
#define _CRT_SECURE_NO_WARNINGS // 在 VS 环境下，在 CPP 中使用 C 的字符串处理函数
#include <iostream>
#include <cstring>

using namespace std;

//定义类，含有静态与非静态成员与函数
class MyClass {
public:
    static int static_data; // 静态数据成员
    int non_static_data; // 非静态数据成员
    char* char_pointer; // 字符指针
    int& data_ref; // 引用
    //构造函数
    MyClass(const char* str) :data_ref(non_static_data) {
        char_pointer = new char[30];
        strcpy(char_pointer, str);
        cout << str << " is constructed." << endl;
        non_static_data = strlen(str);
    }
    //拷贝构造
    MyClass(const MyClass& obj) :data_ref(non_static_data) {
        char_pointer = new char[30];
        strcpy(char_pointer, obj.char_pointer);
        cout << obj.char_pointer << " is copied." << endl;
        non_static_data = obj.non_static_data;

    }
    //析构函数
    ~MyClass() {
        cout << char_pointer << " is destructed." << endl;
        delete[] char_pointer;
```

```
	}
	static void static_member_function() {
		cout << "static_member_function is called." << endl;
	}
	void non_static_member_function() {
		cout << char_pointer << "non_static_member_function is called." << endl;

	}
};

//初始化静态成员
int MyClass::static_data = 0;

//定义函数
void func(const char*str) {
	MyClass obj(str);
	cout << obj.char_pointer << " address is " << &obj << endl;
	cout << "static_data in " << obj.char_pointer << " is " << obj.static_data << endl;
	cout << "char_pointer in " << obj.char_pointer << " is " << obj.char_pointer << endl;
	cout << "non_static_data in " << obj.char_pointer << " is " << obj.non_static_data << 
endl;
	cout << "data_ref in " << obj.char_pointer << " is " << obj.data_ref << endl;
	cout << "address of static_data in " << obj.char_pointer << " is " << &obj.static_data << 
endl;
	cout << "address of char_pointer in " << obj.char_pointer << " is " << &obj.char_pointer 
<< endl;
	cout << "address of non_static_data in " << obj.char_pointer << " is " << 
&obj.non_static_data << endl;
	cout << "address of data_ref in " << obj.char_pointer << " is " << &obj.data_ref << endl;
	cout << "size of " << obj.char_pointer<<" is " << sizeof(obj) << endl;
}

//创建全局对象
MyClass global_obj1("global_obj1");
MyClass global_obj2("global_obj2");

int main() {

	//创建局部对象
	MyClass local_obj1("local_obj1");
	MyClass local_obj2("local_obj2");

	//动态创建对象
	MyClass* dynamic_obj1 = new MyClass("dynamic_obj1");
```

```
    MyClass* dynamic_obj2 = new MyClass("dynamic_obj2");

    cout << "global_obj1 address:" << &global_obj1 << endl;
    cout << "global_obj2 address:" << &global_obj2 << endl;
    cout << "local_obj1 address:" << &local_obj1 << endl;
    cout << "local_obj2 address:" << &local_obj2 << endl;
    cout << "dynamic_obj1 address:" << dynamic_obj1 << endl;
    cout << "dynamic_obj2 address:" << dynamic_obj2 << endl;
    cout << endl;

    //在函数中定义的局部对象
    func("func_obj1");
    func("func_obj2");

    // 全局对象
    cout << endl;
    cout << "static_data in global_obj1 is " << global_obj1.static_data << endl;
    cout << "char_pointer in global_obj1 is " << global_obj1.char_pointer << endl;
    cout << "non_static_data in global_obj1 is " << global_obj1.non_static_data << endl;
    cout << "data_ref in global_obj1 is " << global_obj1.data_ref << endl;
    cout << "address of static_data in global_obj1 is " << &global_obj1.static_data << endl;
    cout << "address of char_pointer in global_obj1 is " << &global_obj1.char_pointer <<
endl;
    cout << "address of non_static_data in global_obj1 is " << &global_obj1.non_static_data
<< endl;
    cout << "address of data_ref in global_obj1 is " << &global_obj1.data_ref << endl;
    cout << "size of global_obj1 is " << sizeof(global_obj1) << endl;

    cout << endl;
    cout << "static_data in global_obj2 is " << global_obj2.static_data << endl;
    cout << "char_pointer in global_obj2 is " << global_obj2.char_pointer << endl;
    cout << "non_static_data in global_obj2 is " << global_obj2.non_static_data << endl;
    cout << "data_ref in global_obj2 is " << global_obj2.data_ref << endl;
    cout << "address of static_data in global_obj2 is " << &global_obj2.static_data << endl;
    cout << "address of char_pointer in global_obj2 is " << &global_obj2.char_pointer <<
endl;
    cout << "address of non_static_data in global_obj2 is " << &global_obj2.non_static_data
<< endl;
    cout << "address of data_ref in global_obj2 is " << &global_obj2.data_ref << endl;
    cout << "size of global_obj2 is " << sizeof(global_obj2) << endl;

    // 局部对象
    cout << endl;
    cout << "static_data in local_obj1 is " << local_obj1.static_data << endl;
    cout << "char_pointer in local_obj1 is " << local_obj1.char_pointer << endl;
    cout << "non_static_data in local_obj1 is " << local_obj1.non_static_data << endl;
    cout << "data_ref in local_obj1 is " << local_obj1.data_ref << endl;
    cout << "address of static_data in local_obj1 is " << &local_obj1.static_data << endl;
    cout << "address of char_pointer in local_obj1 is " << &local_obj1.char_pointer << endl;
    cout << "address of non_static_data in local_obj1 is " << &local_obj1.non_static_data <<
endl;
    cout << "address of data_ref in local_obj1 is " << &local_obj1.data_ref << endl;
    cout << "size of local_obj1 is " << sizeof(local_obj1) << endl;

    cout << endl;
    cout << "static_data in local_obj2 is " << local_obj2.static_data << endl;
    cout << "char_pointer in local_obj2 is " << local_obj2.char_pointer << endl;
    cout << "non_static_data in local_obj2 is " << local_obj2.non_static_data << endl;
    cout << "data_ref in local_obj2 is " << local_obj2.data_ref << endl;
    cout << "address of static_data in local_obj2 is " << &local_obj2.static_data << endl;
    cout << "address of char_pointer in local_obj2 is " << &local_obj2.char_pointer << endl;
    cout << "address of non_static_data in local_obj2 is " << &local_obj2.non_static_data <<
endl;
    cout << "address of data_ref in local_obj2 is " << &local_obj2.data_ref << endl;
    cout << "size of local_obj2 is " << sizeof(local_obj2) << endl;

    // 动态对象
    cout << endl;
    cout << "static_data in dynamic_obj1 is " << dynamic_obj1->static_data << endl;
    cout << "char_pointer in dynamic_obj1 is " << dynamic_obj1->char_pointer << endl;
    cout << "non_static_data in dynamic_obj1 is " << dynamic_obj1->non_static_data <<
endl;
    cout << "data_ref in dynamic_obj1 is " << dynamic_obj1->data_ref << endl;
    cout << "address of static_data in dynamic_obj1 is " << &dynamic_obj1->static_data <<
endl;
    cout << "address of char_pointer in dynamic_obj1 is " << &dynamic_obj1->char_pointer
<< endl;
    cout << "address of non_static_data in dynamic_obj1 is " <<
&dynamic_obj1->non_static_data << endl;
    cout << "address of data_ref in dynamic_obj1 is " << &dynamic_obj1->data_ref << endl;
    cout << "size of dynamic_obj1 is " << sizeof(dynamic_obj1) << endl;


    cout << "static_data in dynamic_obj2 is " << dynamic_obj2->static_data << endl;
    cout << "char_pointer in dynamic_obj2 is " << dynamic_obj2->char_pointer << endl;
    cout << "non_static_data in dynamic_obj2 is " << dynamic_obj2->non_static_data <<
endl;
    cout << "data_ref in dynamic_obj2 is " << dynamic_obj2->data_ref << endl;
    cout << "address of static_data in dynamic_obj2 is " << &dynamic_obj2->static_data <<
endl;
    cout << "address of char_pointer in dynamic_obj2 is " << &dynamic_obj2->char_pointer
<< endl;
    cout << "address of non_static_data in dynamic_obj2 is " <<
&dynamic_obj2->non_static_data << endl;
    cout << "address of data_ref in dynamic_obj2 is " << &dynamic_obj2->data_ref << endl;
    cout << "size of dynamic_obj2 is " << sizeof(dynamic_obj2) << endl;


    // 静态成员地址
    cout << endl;
    cout << "Static data member address: " << &MyClass::static_data << endl;
    // 静态成员函数地址和非静态成员函数地址
    union {
        void* pv;
        void(MyClass::* f)();
    } u;
    u.f = &MyClass::non_static_member_function;
    cout << "Static member function address: " << & MyClass::static_member_function <<
endl;
    cout << "Non-static member function address: " << u.pv << endl;
    // main 函数地址和 func 函数的地址
    cout << "Main function address: " << &main << endl;
    cout << "Func function address: " << &func << endl;

    delete dynamic_obj1;
    dynamic_obj1 = NULL;
    delete dynamic_obj2;
    dynamic_obj2 = NULL;

    return 0;
}
```

运行结果:



```
global_obj1 is constructed.
global_obj2 is constructed.
local_obj1 is constructed.
local_obj2 is constructed.
dynamic_obj1 is constructed.
dynamic_obj2 is constructed.
global_obj1 address:00007FF77AE24470
global_obj2 address:00007FF77AE24458
local_obj1 address:00000001000FF9F8
local_obj2 address:00000001000FFA28
dynamic_obj1 address:000001B70FCF4440
dynamic_obj2 address:000001B70FCF4500

func_obj1 is constructed.
func_obj1 address is 00000001000FF8C8
static_data in func_obj1 is 0
char_pointer in func_obj1 is func_obj1
non_static_data in func_obj1 is 9
data_ref in func_obj1 is 9
address of static_data in func_obj1 is 00007FF77AE24450
address of char_pointer in func_obj1 is 00000001000FF8D0
address of non_static_data in func_obj1 is 00000001000FF8C8
address of data_ref in func_obj1 is 00000001000FF8C8
size of func_obj1 is 24
func_obj1 is destructed.
func_obj2 is constructed.
func_obj2 address is 00000001000FF8C8
static_data in func_obj2 is 0
char_pointer in func_obj2 is func_obj2
non_static_data in func_obj2 is 9
data_ref in func_obj2 is 9
address of static_data in func_obj2 is 00007FF77AE24450
address of char_pointer in func_obj2 is 00000001000FF8D0
address of non_static_data in func_obj2 is 00000001000FF8C8
address of data_ref in func_obj2 is 00000001000FF8C8
size of func_obj2 is 24
func_obj2 is destructed.

static_data in global_obj1 is 0
char_pointer in global_obj1 is global_obj1
non_static_data in global_obj1 is 11
data_ref in global_obj1 is 11
address of static_data in global_obj1 is 00007FF77AE24450
address of char_pointer in global_obj1 is 00007FF77AE24478
address of non_static_data in global_obj1 is 00007FF77AE24470
address of data_ref in global_obj1 is 00007FF77AE24470
size of global_obj1 is 24

static_data in global_obj2 is 0
char_pointer in global_obj2 is global_obj2
non_static_data in global_obj2 is 11
data_ref in global_obj2 is 11
address of static_data in global_obj2 is 00007FF77AE24450
address of char_pointer in global_obj2 is 00007FF77AE24460
address of non_static_data in global_obj2 is 00007FF77AE24458
address of data_ref in global_obj2 is 00007FF77AE24458
size of global_obj2 is 24

static_data in local_obj1 is 0
char_pointer in local_obj1 is local_obj1
non_static_data in local_obj1 is 10
data_ref in local_obj1 is 10
address of static_data in local_obj1 is 00007FF77AE24450
address of char_pointer in local_obj1 is 00000001000FFA00
address of non_static_data in local_obj1 is 00000001000FF9F8
address of data_ref in local_obj1 is 00000001000FF9F8
size of local_obj1 is 24

static_data in local_obj2 is 0
char_pointer in local_obj2 is local_obj2
non_static_data in local_obj2 is 10
data_ref in local_obj2 is 10
address of static_data in local_obj2 is 00007FF77AE24450
address of char_pointer in local_obj2 is 00000001000FFA30
address of non_static_data in local_obj2 is 00000001000FFA28
address of data_ref in local_obj2 is 00000001000FFA28
size of local_obj2 is 24

static_data in dynamic_obj1 is 0
char_pointer in dynamic_obj1 is dynamic_obj1
non_static_data in dynamic_obj1 is 12
data_ref in dynamic_obj1 is 12
address of static_data in dynamic_obj1 is 00007FF77AE24450
address of char_pointer in dynamic_obj1 is 000001B70FCF4448
address of non_static_data in dynamic_obj1 is 000001B70FCF4440
address of data_ref in dynamic_obj1 is 000001B70FCF4440
size of dynamic_obj1 is 8
static_data in dynamic_obj2 is 0
char_pointer in dynamic_obj2 is dynamic_obj2
non_static_data in dynamic_obj2 is 12
data_ref in dynamic_obj2 is 12
address of static_data in dynamic_obj2 is 00007FF77AE24450
address of char_pointer in dynamic_obj2 is 000001B70FCF4508
address of non_static_data in dynamic_obj2 is 000001B70FCF4500
address of data_ref in dynamic_obj2 is 000001B70FCF4500
size of dynamic_obj2 is 8

Static data member address: 00007FF77AE24450
Static member function address: 00007FF77AE1152D
Non-static member function address: 00007FF77AE112A3
Main function address: 00007FF77AE11352
Func function address: 00007FF77AE11190
dynamic_obj1 is destructed.
dynamic_obj2 is destructed.
local_obj2 is destructed.
local_obj1 is destructed.
global_obj2 is destructed.
global_obj1 is destructed.
```

结果分析:
本实验在 VS 的 debug 模式下进行，分析如下:

1. 对象构造与析构顺序：先构造全局对象再构造局部对象，先构造的对象后析构，动态对象需要手动虚构，在函数中构造的对象在函数结束的时候会进行销毁，同时，发现两个在函数内创建的两个对象的地址相同，均为 00007FF632494450，表明函数在内部可能使用堆栈实现，创建函数的时候压入堆栈，销毁函数的时候弹出。
2. 两个全局对象，两个局部对象，两个动态创建的对象地址相近，通过计算可以发现，两个对象是相邻放置的，全局对象、局部对象、动态对象的存放地址较远，表明其分别放在全局区、栈区、堆区。
3. 所用对象访问的静态对象的地址均为 00007FF632494450，表明所有对象共用一份静态对象。静态对象的地址与全局对象的地址较为相近，表明其存放在全局区，当代码执行到初始化语句的时候为其分配空间。
4. 对象中含有引用成员，表示的是一个 int 类型非静态成员 non_static_data 的引用，发现引用的地址与 non_static_data 的地址相同，表明引用指向所引用的对象的地址，实际上，引用相当于一个常指针，这里表示的就是一个 int* const 类型的指针。
5. 类在存放的时候包含了内存对齐，引用和字符串指针的大小均为 8B，int 类型成员大小为 4B，sizeof 计算对象大小的时候不包含静态成员的大小，所以总大小为 20B，在计算机内存了 24B，表明有内存对齐，此实验中动态对象的大小指的是所指向指针的大小，为 8B，实际上开辟内存存放的对象也为 24B.
6. 在对象内部，成员按定义的顺序逐个存放，在此实验中，先存放 non_static_data，而后 char_pointer，最后是 data_ref。
7. 我们利用联合体来打印了非静态成员的函数地址（若直接使用&Myclass:: non_static_member_function 进行访问会发现返回值为 1，不能正确打印出函数地址），发现其与与静态函数地址、main 函数地址和外部函数的地址相近，表示其均处于代码区，不在对象内部，实际上，编译器会把成员函数转换为外部函数，通过 this 指针进行传值并通过修改函数名，保证其在程序中独一无二，对于静态成员函数，将其视为全局外部函数，且不需要 this 指针

实验总结:

通过此次实验，我了解了 C++中的内存模型，知道了对象的开辟与虚构顺序，以及对象在内存中的存储位置，全局对象、局部对象、动态对象放在不同的区域，了解了静态成员在内存中的存储位置，所以的对象共用一份静态成员数据，知道了对象中的内存对齐现象，和对象内部的数据成员的存放顺序，非静态成员存放在对象内部。同时，我还利用联合体打印了非静态成员函数的地址，知道了为了节约内存，成员函数只在内存中存放一份，且成员函数的代码放在代码区而不是对象内部。