# 计算机学院 高级语言程序设计 课程实验报告

| | | |
|---|---|---|
| 实验题目：类与对象（三），数据共享与保护 | | 学号：202200400053 |
| 日期：2024-03-21 | 班级： 2202 | 姓名： 王宇涵 |
| Email：1941497679@qq.com | | |

**实验目的：**

1. 掌握 UML 图的绘制。
2. 了解 struct、union 在 C++中的扩展应用
3. 掌握联合体和位域应用
4. 数据共享与保护：

（1）掌握标识符的不同作用域和生命周期，对象的静态、动态生存期在编程中的应用

（2）掌握静态（static）类成员的使用

（3）分析友元函数、友元类的使用

**实验软件和硬件环境：**

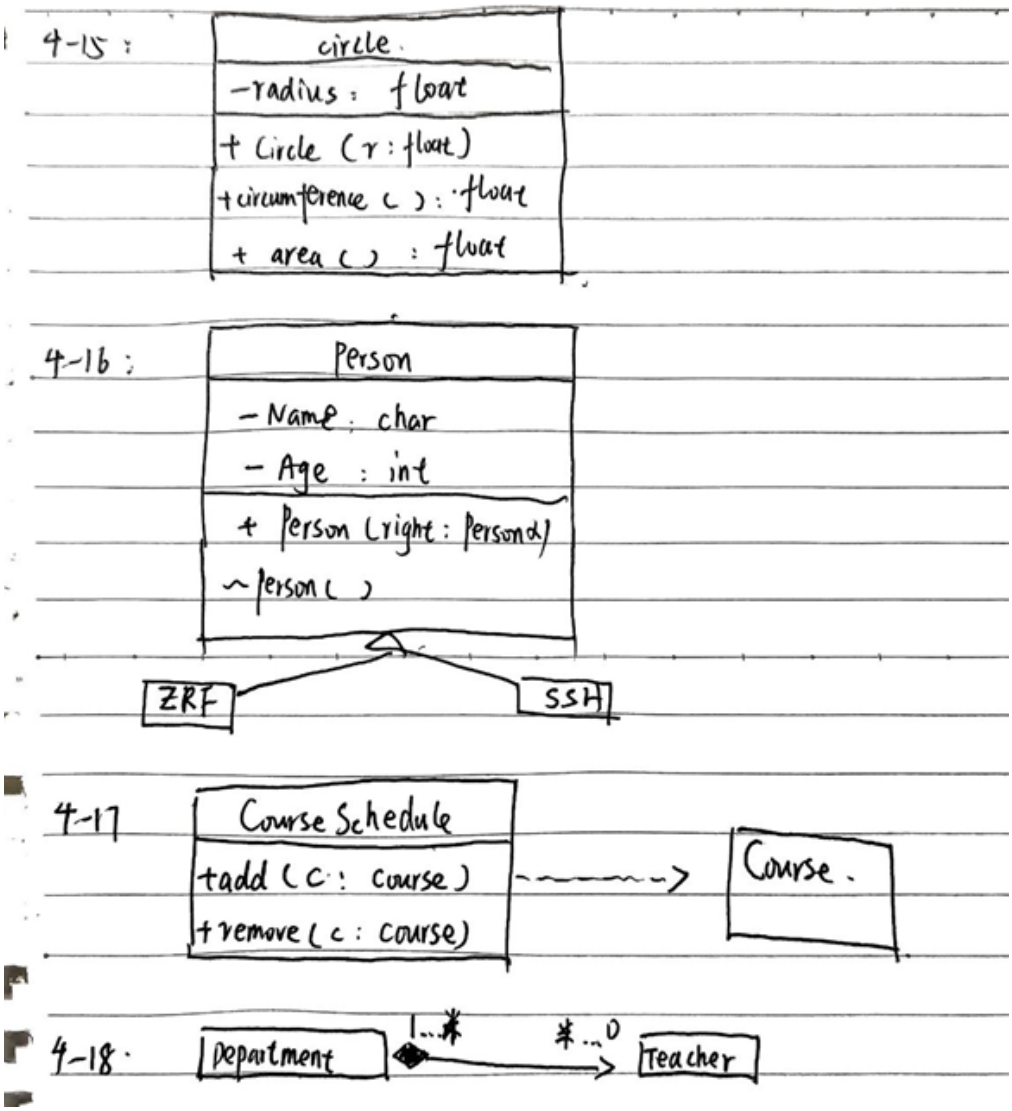
软件环境 ：VSCODE + DEV-C++

硬件环境 ：Legion Y7000P

**实验步骤与内容：**

**1. 习题：**

**画图习题 4-15 至 4-18 的 UML 图。问 4-17 中有聚合关系吗？**




答：4-17 有聚集关系

## 2. 请分析对比 union.zip 和 bitfield.zip 中的代码

```
English: B
Calculus: PASS
C++ Programming: 85
```

union.zip 输出

在 union.zip 中

4_8_1 采用的是无名联合体作为一个类的内嵌函数，这种联合体可以直接用联合体中成员项的名字直接访问.

4_8_1 也是类内无名的联合体，m 是联合体的一个对象，这种对象内的成员项具有相同的内存地址，不能同时使用.

4_8_2 是独立的有名联合体，与前两种不同的是，它可以定义出别的对象，也可以重载自己的构造函数.

```
8
8
```
bitfield.zip 输出

在 bitfield.zip 中

Bitfield 和 Bitfield1 中, 定义变量的时候, char 和 int 类型的变量会根据值的大小来判断是否共用一个字节, 需要注意的是 char 类型与 int 类型产生对齐, 因此系统会分配 4 * 2 = 8byte 的空间, 因此结果输出 8.

两者不同的是: bitfield 中额外直接定义了类并打印出所占字节大小, 而 bitfield 和 bitfield 均在函数里定义了类并打印出所占字节大小, 结果是相等的.

**3. 运行 lab5_1.cpp 程序，尝试在“cout<<"Step into fn1()..."之前添加如下代码块：**

```
{
int x=3,y=4;
  cout<<"in code block: "<<endl;
cout << "x = " << x << endl;
cout << "y = " << y << endl;
}
```

**分析修改后的运行结果。**

输出结果

```
Begin...
x = 1
y = 2
Evaluate x and y in main()...
x = 10
y = 20
in code block:
x = 3
y = 4
Step into fn1()...
x = 1
y = 200
Back in main
x = 3
y = 4
```

分析 ： 首先定义了全局变量 x = 1, y = 2, 后在 main 函数中定义了局部变量 x = 10, y = 20, 此时进行了对于局部变量的修改, x = 3 , y = 4, 进入函数 fn1(), 其中定义了局部变量 y = 200, 打印出了局部变量 y 和全局变量 x, 回到 main 函数, 局部变量有效, 因此打印出 x = 3, y = 4.

**4. 类的静态成员：**

**练习第 5 章习题 5-7 的程序，分析运行结果。问：**

**（1）代码中 istAge(age)的含义；**

**（2）讨论静态变量 numOfCats 与全局变量的区别。**

**（3）int getNumofCats() 前的 static 可以去掉吗？**

**（4）telepathicFuction() 除了原本作为孤立函数，还可以怎样设计？**

答 : (1) 初始化列表, 将私有变量 itsAge 初始化为 age

(2) 静态变量的作用域限制在声明它的类中，而全局变量的作用域是整个程序。

同时: 静态变量为该类的所有对象共享,具有静态生存期, 由类外定义和初始化.

(3) 不能去掉 ：只有静态成员函数才可以调用静态成员变量

(4) TelepathicFunction() 可以作为类 Cat 的静态成员函数，因为它与类 Cat 相关，可以访问 Cat 的静态成员变量和方法

```
static void TelepathicFunction() {
    std::cout << "There are " << numOfCats << " cats alive!\n";
}
```

**5. 友元：分析第 5 章习题 5-4，讨论 class fuel 中的 friend class engine 有必要加吗？**

答: 没有必要加, 如果加上代表允许 engine 访问 fuel 中的私有和保护成员, 而题目要求允许 fuel 成员访问 engine 私有和保护成员, 因此在 class engine 中加上 friend class fuel 即可

## 结论分析与体会：

通过这次实验, 我掌握了 UML 图的绘制方法和类之间的关系类型, 联合体变量位域和对齐概念, 深入理解了各个变量的作用域和生命周期, 类的静态成员和友元等知识点, 锻炼了自己的编码能力, 收获良多.

## 就实验过程中遇到的问题及解决处理方法，自拟 1－3 道问答题：

**1. 分析类/结构体所占空间大小?**

答 ：按照变量类型和值先按照位域依次排列, 再根据对齐原则分配空间.

**2. 探讨在面向对象的程序设计中，有哪些可以替代全局变量的共享数据方案？**

答 ：(1)局部化数据：尽可能地将数据局部化到需要使用它的类中，避免将数据暴露给整个程序。这样可以降低数据的可见性，减少不必要的依赖关系，提高代码的可维护性和可测试性.

(2) 单例模式确保类只有一个实例，并提供一个全局访问点来访问该实例。通过单例模式，可以将需要共享的数据作为单例类的成员变量，从而避免使用全局变量.

(3) 创建一个上下文对象，将需要共享的数据存储在这个上下文对象中，然后将上下文对象作为参数传递给需要使用这些数据的类的构造函数或者方法.