实验报告（一）

#### 一、实验目的

(1) 掌握类的声明和使用

(2) 掌握类的声明和对象的声明

(3) 复习具有不同访问属性的成员的访问方式

(4) 观察构造函数和析构函数的执行过程

(5) 学习类的组合使用方法

(6) 观察程序运行中变量的作用域、生存期和可见性。

(7) 学习类的静态成员的使用。

(8) 学习多文件结构在C++程序中的使用。

(9) 学会使用Visual Studio 2008 开发环境中的debuge调试功能：单步执行、设置断点、观察变量值。

#### 二、实验内容

声明一个CPU类，包含等级（rank），频率（frequency）,电压（voltage）等属性，有两个共有成员函数run,stop,其中，rank为枚举类型CPU\_Rank,声明为enumCPU\_Rank{P1=1,P2,P3,P4,P5,P6,P7},frequency为单位是MHz的整型数，voltage为浮点型的电压值，观察构造函数和析构函数的调用顺序。

#### 三、实验环境

硬件环境：AMD A12；操作系统：Windows 10；C++环境：Visual Studio 2017

#### 四、程序设计原理和方法

##### 1、枚举类型的设置。

首先，根据题意定义一个枚举类型，它包含了一系列枚举常量。将它放在所有代码的前面，以便后面代码的使用需求。它具有静态生存期。

##### 2、相关对象设置。

根据题意定义一个CPU类，其中包含构造函数、题目中所提到的stop和run函数成员、整型frequency、浮点型voltage和枚举量rank的数据成员、以及析构函数。

为了能在输出结果时明确调用了哪些成员函数，所以在成员函数编译时，都使用cout，让其在调用时输出该函数的名称。

##### 3、主函数定义。

由于实验一的目的是观察构造函数和析构函数的调用顺序，所以在主函数中声明CPU的对象cpu，并通过cpu.run()与cpu.stop()调用其成员函数，并观察此对象创建与消灭时构造函数和析构函数的调用情况。

#### 五、程序代码

#include "pch.h"

#include <iostream>

using namespace std;

enum CPU\_RANK { P1 = 1, P2, P3 = 5, P4, P5 };//union是每个数据公用内存；struct和enum不共用内存，而是每个都会占用内存。union和enum每次只能出现一次，而struct可以同时出现许多值

//若给了P1=1，则往后都默认加一。除非自己赋值了

class CPU {

public:

CPU() {

cout << "CPU的构造函数1" << endl;

}

CPU(CPU\_RANK rank, int frequency, float voltage) {

this->rank = rank;

this->frequency = frequency;

this->voltage = voltage;

cout << rank << endl;

cout << frequency << endl;

cout << voltage << endl;

cout << "CPU的构造函数2" << endl;

}

void run();

void stop();

~CPU() { cout << "CPU的析构函数" << endl; }

private:

CPU\_RANK rank;//将CPU\_RANK的值赋给私有成员

int frequency;

float voltage;

};

void CPU::run() { cout << "CPU:run函数" << endl; }

void CPU::stop() { cout << "CPU:stop函数" << endl; }

int main() {

CPU cpu(P3, 10, 2.5);

cpu.run();

cpu.stop();

return 0;

}

#### 六、运行结果

5

10

2.5

CPU的构造函数2

CPU:run函数

CPU:stop函数

CPU的析构函数

#### 七、解释运行结果

“5”、“10”、“2.5”和“CPU的构造函数2”是调用了CPU的构造函数的结果，其中：出现5是因为以enum CPU\_RANK中的P3赋值给了rank，同理对于frequency和voltage。

“CPU:run函数”和“CPU:stop函数”是调用了成员函数run和stop的结果。

调用完毕后，执行析构函数，得到“CPU的析构函数”。

#### 八、实验小结

通过实验一，我重新复习了构造函数和析构函数的调用顺序，以及类对象如何通过构造函数给私有成员赋值。同时，我也重新加深了对枚举类型数据的理解，填补了之前的只是空缺。