華中科技大學

课 程 实 验 报 告

课程名称：面向对象程序设计

实验名称：聚合对象的整型队列编程

院　　系：计算机科学与技术

专业班级：物联网工程 1601

学　　号：U201614898

姓　　名：潘　　翔

指导教师：吕　新　桥

2018 年 10 月 16 日

1 需求分析

1.1 题目要求

整型队列是一种先进后出的存储结构，对其进行的操作通常包括判断队列是否为空、向队列顶添加一个整型元素、出队列等。整型队列类型及其操作函数采用面向对象的 C++语言定义，请将完成上述操作的所有函数采用 C++编程。然后写一个 main 函数对队列的所有操作函数进行测试，要求 main 按照《关于 C++实验自动验收系统说明》给定的方式工作。注意，请用实验三的 SATCK 继承形成新的类 QUEUE。分析说明除构造函数以外的函数，加 virtual 说明与不加 virtual 说明有无区别。并说明为什么不将 s2 也作为基类。

```
class QUEUE：public STACK
{
    STACK   s2;
public:
QUEUE(int m);                                  //每个栈最多 m 个元素，要求实现的队列最多能
                                               //入 2m 个元素
QUEUE(const QUEUE&s);                          //用队列 s 拷贝初始化队列
virtual operator int ( ) const;                //返回队列的实际元素个数
virtual int full ( ) const;                    //返回队列是否已满，满返回 1，否则返回 0
virtual int operator[ ](int x)const;           //取下标为 x 的元素，第 1 个元素下标为 0
virtual QUEUE& operator<<(int e);              //将 e 入队列,并返回队列
virtual QUEUE& operator>>(int &e);             //出队列到 e,并返回队列
virtual QUEUE& operator=(const QUEUE&s); //赋 s 给队列,并返回被赋值的队列
virtual void print( ) const;                   //打印队列
virtual ~QUEUE( );                             //销毁队列
};
```

在完成上述程序及测试无误后，请使用队列解决如下舞伴问题，此时 main 用非命令行的方式工作。假定在一次舞会上，男士排成一队，女士排成另一队。每次舞曲响起时，从男队和女队的队头各出一人，配成舞伴跳完此曲，跳完后各自进入自己队列的尾部。若男女两队的初始人数分别为 M 和 F（M 和 F 均为素数，且 M!=F），男队中排在位置 m（m<=M）的男士，非常想和女队位置 f（f<=F）的女士跳舞，问他在第几支曲舞曲才能和该女士跳舞?请编程解决上述问题。

1.2 需求分析

利用 C++语言实现整型栈，栈是一种具有后入先出(LIFO)性质的线性数据结构，此次实

验要求完成在栈上的一些基本操作，具体包括构造、取大小、取元素个数、下标访问元素、入栈、出栈、赋值、相等判定(自加功能)、(打印)输出、销毁等操作。

虽然实现的队列最多能入 2m 个元素，但是由于模拟中入出操作序列的限制，即操作不能违背先进先出的原则，可能还没入 2m 个元素队列就“满”了，所以要注意 full( )函数的实现：不能简单判断队列是否装了 2m 个元素。

2 系统设计

2.1 概要设计

2.2.1 系统整体架构图

本系统分为 4 个模块：平台判断模块，mian 调用模块，数据结构模块，调试模块。

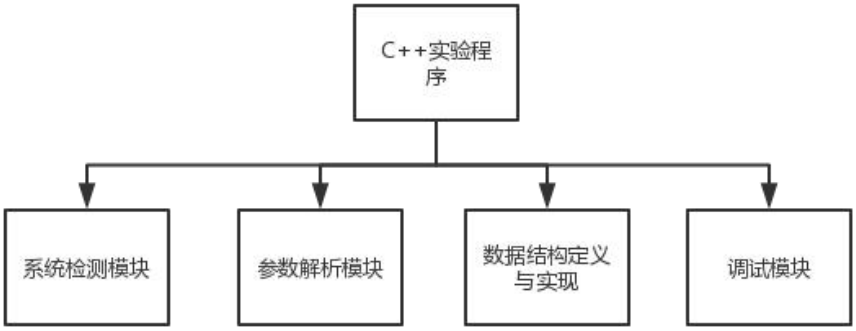

图 1-1 系统整体架构图

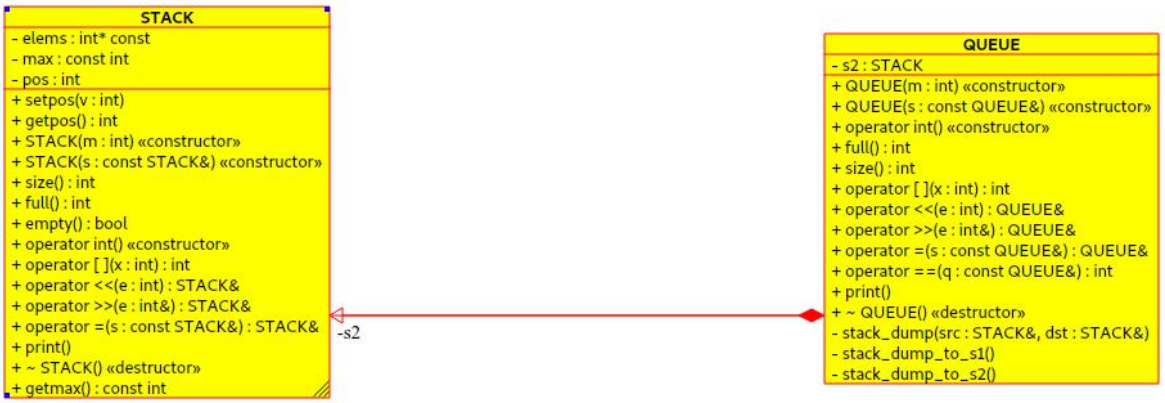

图 2-2 系统 UML 图

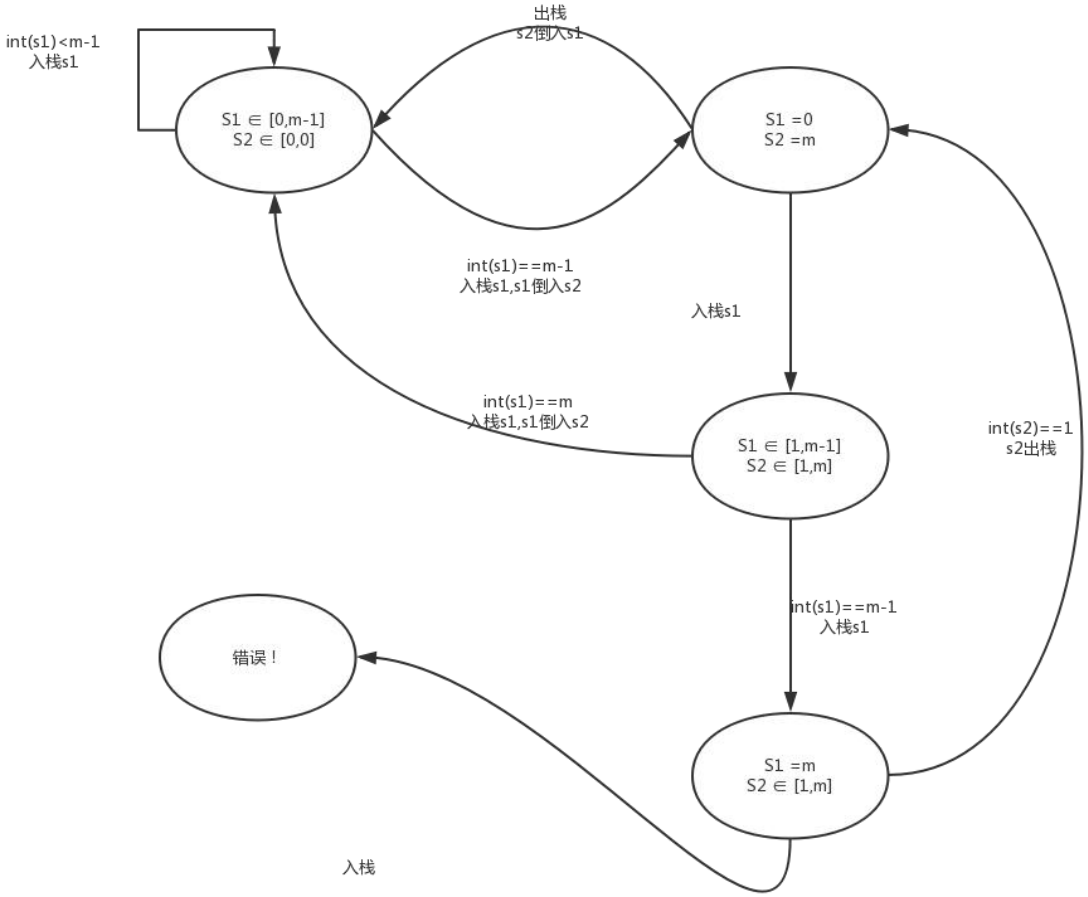

图 1-3 系统状态机图

2.2.1 系统总体流程

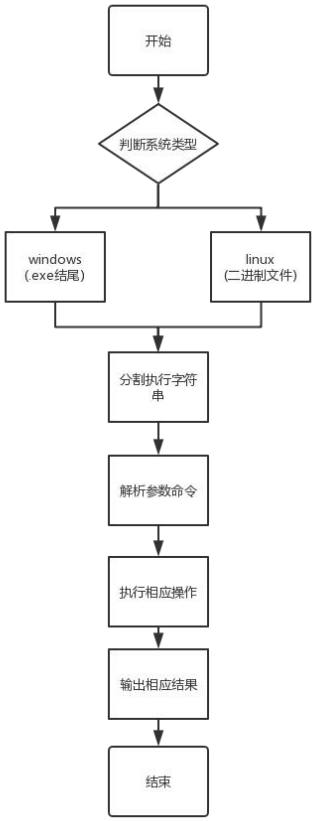

图 1-2 系统总体流程图

2.1.3 系统参数

表 1-1 系统参数表

命令	是否含有参数值	参数值含义
-S		设定栈队或队列大小为 a
-I i	√	入 i 个元素
-O o	√	出 o 个原色
-C		深拷贝构造
-A a	√	深拷贝赋值
-N		栈中剩余元素个数
-G g	√	表示得到栈中下标为 g 的元素

2.1.3 模块接口

1) 操作系统判断

描述: 进行系统判断，调用相应的函数接口

输入: 系统宏定义

输出: debug(操作系统类型)

2) 调试模块

描述: 利用宏定义开关进行编译选项

输入: 程序编译类型 debug/release

输出: 是否输出 log

3) 主函数

描述: 根据相应的操作系统类型调用不同的子程序

输入: 操作系统类型

调用: 模块主函数

4) 模块主函数

描述: 根据相应的操作系统类型调用不同的子程序

输入: 输入参数

调用: 相应操作函数

输出: 操作结果

5) Set/get 方法模块

描述: 利用宏定义封装需要 set/get

输入: 变量名称,变量权限

输出: 变量声明/相应的 set/get 方法。

2.2 详细设计

2.2.1 数据结构类定义

```
class QUEUE:public STACK{
    STACK s2;
    public:
    QUEUE(int m);                          //每个栈最多 m 个元素，要求实现的队列最多能入 2m 个元素
    QUEUE(const QUEUE&s);                  //用队列 s 拷贝初始化队列
    virtual operator int ( ) const;         //返回队列的实际元素个数
    virtual int full () const;              //返回队列是否已满，满返回 1，否则返回 0
    virtual int size(void) const;           //max size
    virtual int operator[ ](int x)const;    //取下标为 x 的元素，第 1 个元素下标为 0
    virtual QUEUE& operator<<(int e);       //将 e 入队列,并返回队列
    virtual QUEUE& operator>>(int &e);      //出队列到 e,并返回队列
    virtual QUEUE& operator=(const QUEUE&s);//赋 s 给队列,并返回被赋值的队列
```

```
    virtual int operator==(const QUEUE &q) const;
    virtual void print( ) const;                    //打印队列
    virtual ~QUEUE( );                              //销毁队列
    private:
    void stack_dump(STACK& src,STACK& dst);         //两个栈之间执行出入栈转换
    void stack_dump_to_s1();
    void stack_dump_to_s2();
};
```

2.2.2 数据操作设计

```
    virtual operator int ( ) const;                 //返回队列的实际元素个数
    virtual int full () const;                      //返回队列是否已满，满返回1，否则返回0
    virtual int size(void) const;                   //max size
    virtual int operator[ ](int x)const;            //取下标为x的元素，第1个元素下标为0
    virtual QUEUE& operator<<(int e);               //将e入队列,并返回队列
    virtual QUEUE& operator>>(int &e);              //出队列到e,并返回队列
    virtual QUEUE& operator=(const QUEUE&s);        //赋s给队列,并返回被赋值的队列
    virtual int operator==(const QUEUE &q) const;
    virtual void print( ) const;                    //打印队列
    virtual ~QUEUE( );                              //销毁队列
    private:
    void stack_dump(STACK& src,STACK& dst);         //两个栈之间执行出入栈转换
    void stack_dump_to_s1();
    void stack_dump_to_s2();
```

1) QUEUE(int m);
   a) 描述：
      每个栈最多 m 个元素，要求实现的队列最多能入 2m 个元素
   b) 入口参数：
      int m　　m 个元素
   c) 出口参数：
      void

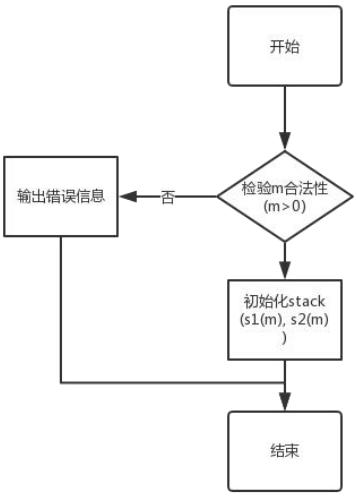

图 2-1 QUEUE(int m)流程图

2) QUEUE(const QUEUE&s);
   a) 描述：
      用队列 s 拷贝初始化队列
   b) 入口参数：
      const QUEUE&s　　现存栈 s 引用
   c) 出口参数：
      void

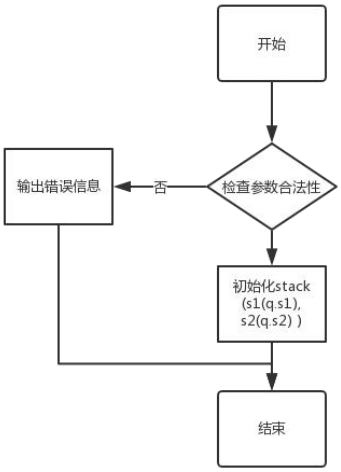

图 2-2 QUEUE(const QUEUE&s)

3) virtual int size ( ) const;
   a) 描述：
      返回当前状态的容量
   b) 入口参数：
      QUEUE
   c) 出口参数：
      int

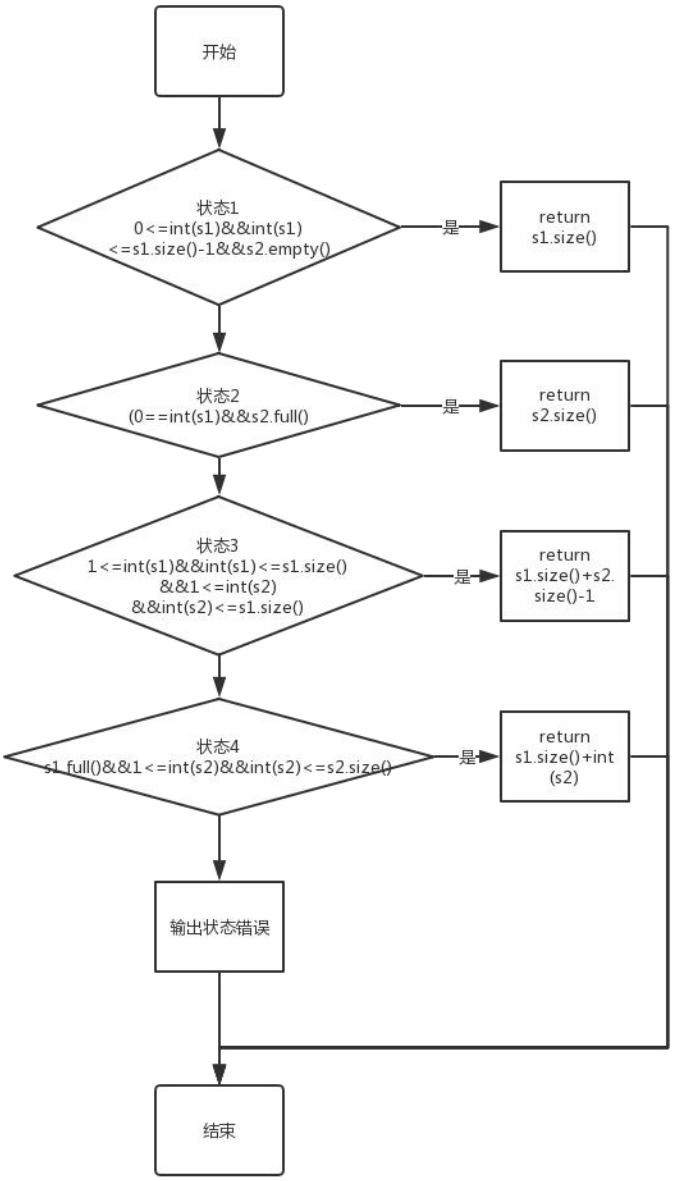

图 2-3 virtual int size ( ) const 流程图

4) virtual operator int ( ) const;

a) 描述：

返回队列的实际元素个数

b) 入口参数：

QUEUE

c) 出口参数：

void

5) virtual int full ( ) const;

a) 描述：

返回队列是否已满，满返回 1，否则返回 0

b) 入口参数：

QUEUE

c) 出口参数：

Int

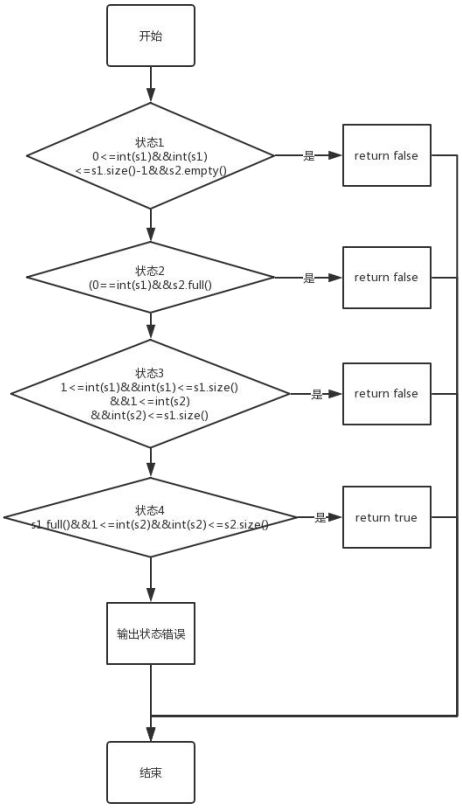

图 2-3 virtual int full ( ) const 流程图

6) virtual int operator[ ](int x)const;

a) 描述：

取下标为 x 的元素,第 1 个元素下标为 0

b) 入口参数：

Int x

c) 出口参数：

Int 下标 x 的数据值

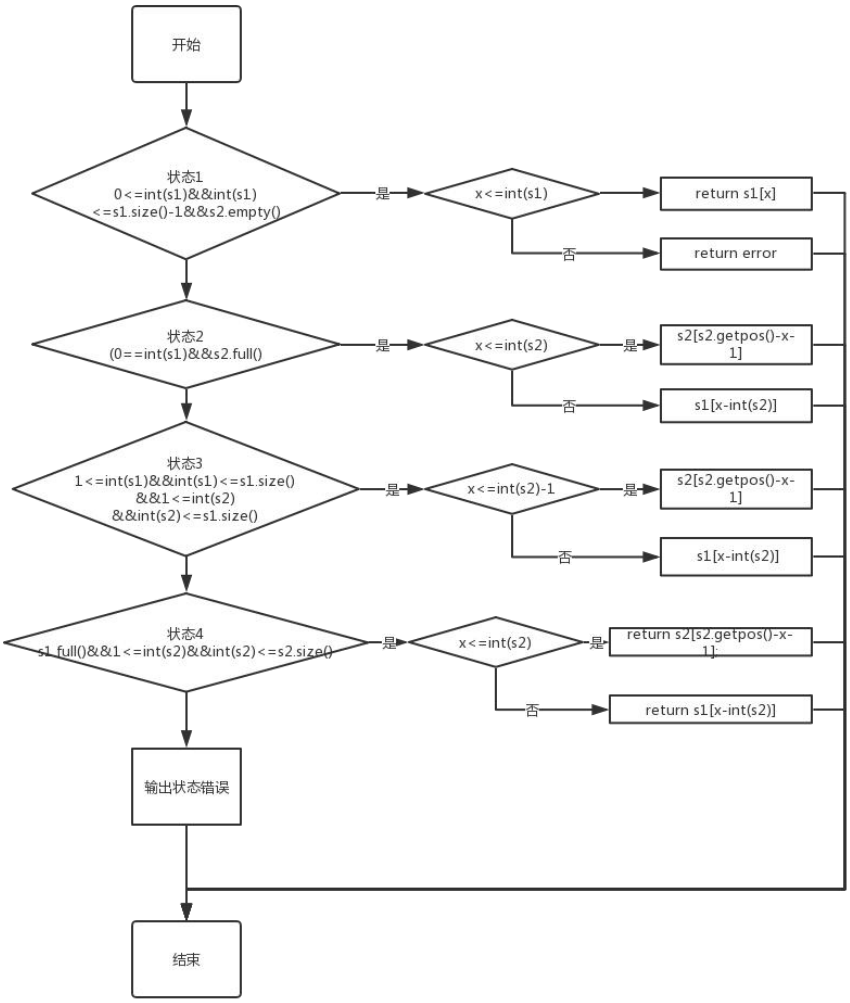

图 2-1 virtual int operator[ ](int x)const 流程图

7) virtual QUEUE& operator<<(int e);

a) 描述：

将 e 入队列,并返回队列

b) 入口参数：

Int e

c) 出口参数：

QUEUE& 用户 que 引用

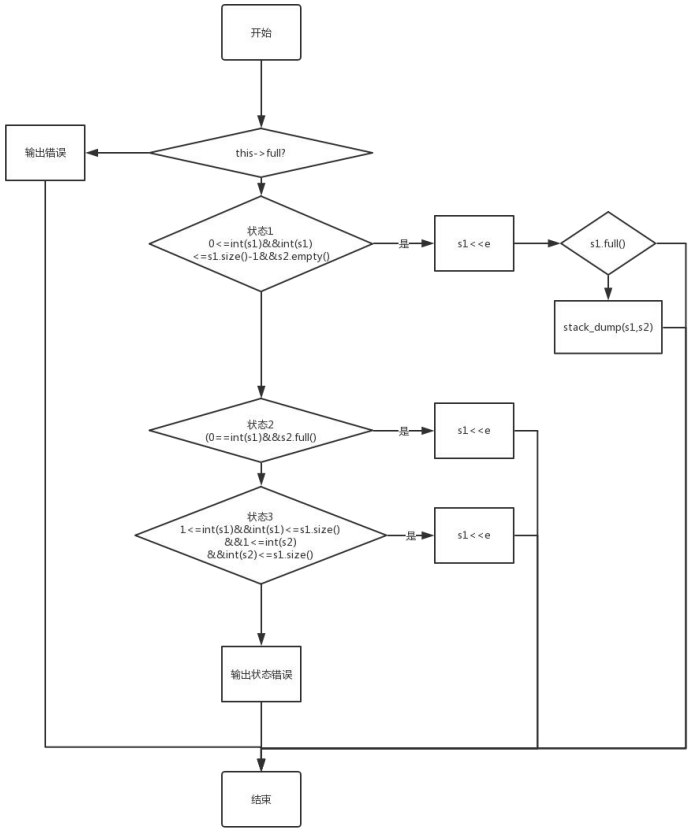

图 2-1 入队列流程图

8) virtual QUEUE& operator>>(int &e);
   a) 描述：
      出队列到 e,并返回队列
   b) 入口参数：
      Int e
   c) 出口参数：

QUEUE&　　用户 QUE 引用

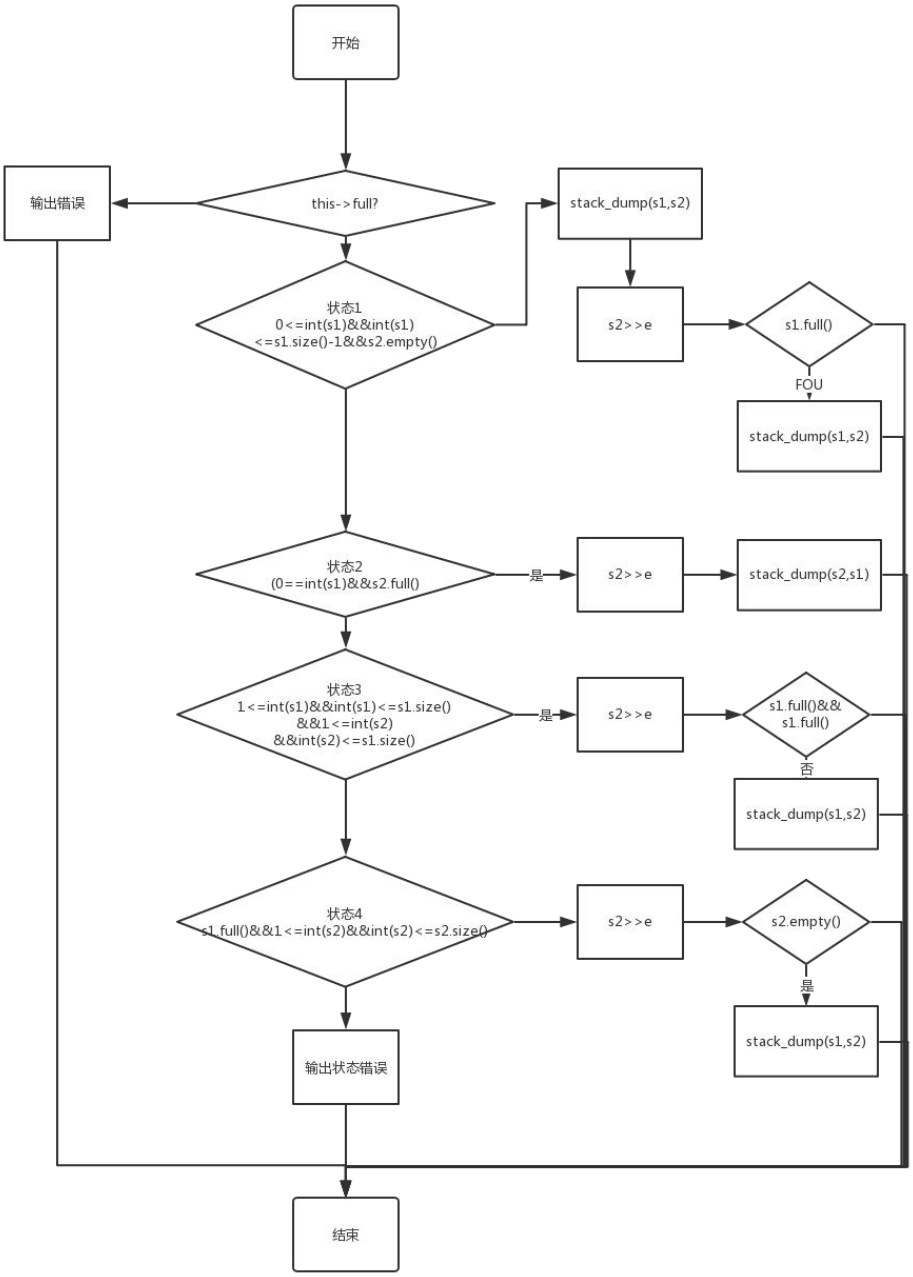

图 2-1　出队列流程图

9) virtual QUEUE& operator=(const QUEUE&s);

a) 描述：

赋 s 给队列,并返回被赋值的队列

b) 入口参数：

const QUEUE&s

c) 出口参数：

QUEUE&　　用户 QUE 引用

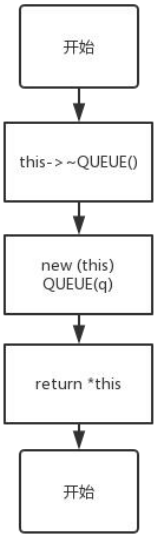

图 2-1 赋值流程图

10) virtual void print( ) const;

a) 描述：

打印队列

b) 入口参数：

void

c) 出口参数：

void

11) virtual ~QUEUE( );

a) 描述：

销毁队列，栈成员的析构函数将被自动调用，故此处不显示调用

b) 入口参数：

void

c) 出口参数：

Void

12) void stack_dump_to_s1();

a) 描述：

S2 倒入 S1;

b) 入口参数：

void

c) 出口参数：

void

13) void stack_dump_to_s2();

a) 描述：

S1 倒入 S2;

b) 入口参数：

void

c) 出口参数：

void

2.2.3 分模块设计

1) 操作系统判断

描述: 进行系统判断，调用相应的函数接口

输入: 系统宏定义

输出: debug(操作系统类型)

2) 调试模块

描述: 利用宏定义开关进行编译选项

输入: 程序编译类型 debug/release

输出: 是否输出 log

3) 主函数

描述: 根据相应的操作系统类型调用不同的子程序

输入: 操作系统类型

调用: 模块主函数

4) 模块主函数

描述: 根据相应的操作系统类型调用不同的子程序

输入: 输入参数

调用: 相应操作函数

输出: 操作结果

5) Set/get 方法模块

描述: 利用宏定义封装需要 set/get

输入: 变量名称

输出: 变量声明/相应的 set/get 方法。

```
#define PropertyBuilderByName(type, name, access_permission)\
    access_permission:\
        type name;\
    public:\
    inline void set##name(type v) {\
        name = v;\
    }\
    inline type get##name() {\
        return name;\
    }\

#define PointerPropertyBuilderByName(type, name, access_permission)\
    access_permission:\
        type* name;\
    public:\
        inline void set##name(type* v){\
            name = v;\
        }\
        inline type* get##name(){\
            return name;\
        }\

#define constPropertyBuilderByName(type, name, access_permission)\
    access_permission:\
        const type name;\
    public:\
    inline void set##name(type v) {\
        name = v;\
    }\
    inline type get##name() {\
        return name;\
    }\
```

6) dance 模块

描述: 进行 dance 功能

输入:

男性位置

女性位置

男性总人数

女性总人数

输出:

跳舞曲数

2.3 Dance_main 设计

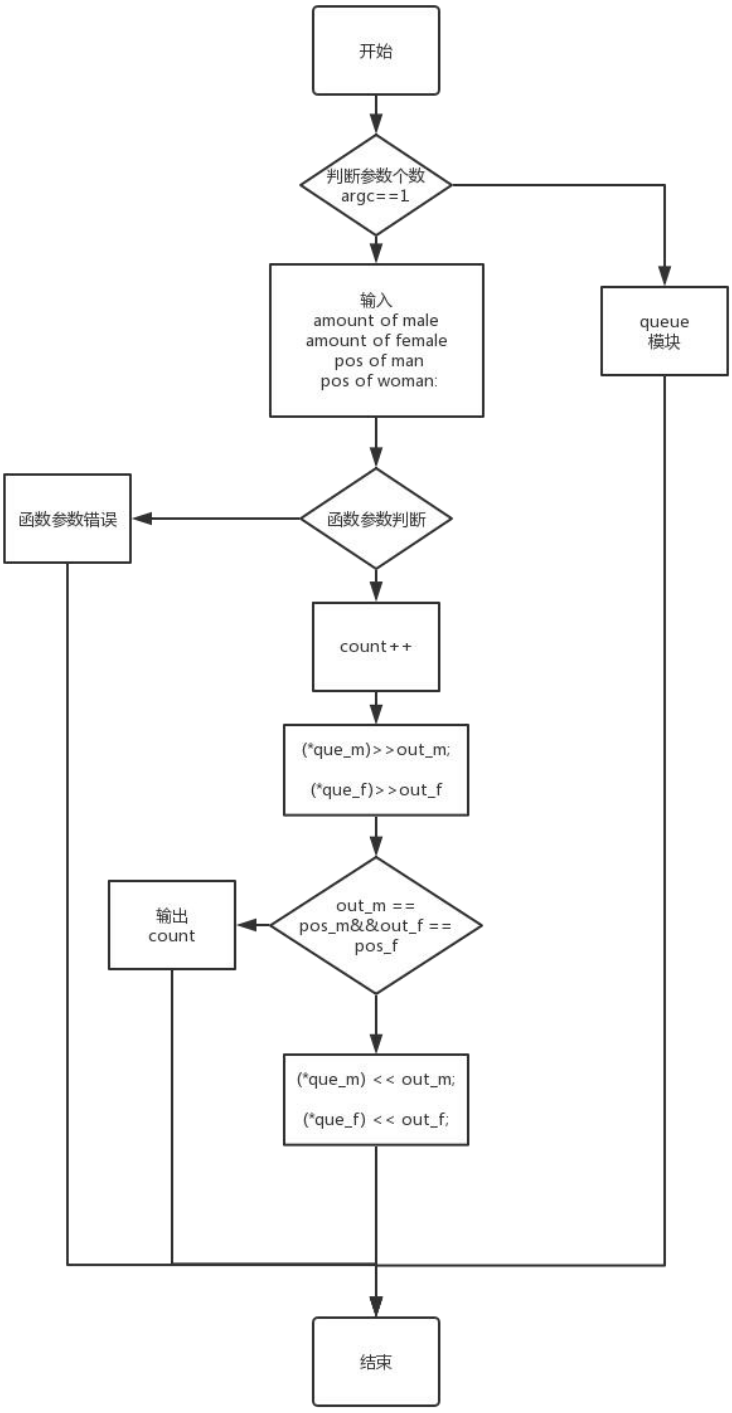

图 2-1 Dance 函数流程图

3 软件开发

3.1 软件开发环境

Language:	C++
Text Editor:	Visual Studio Code 1.29.1
Compiler:	8.1.1 20180531 (GCC)
	Mingw-gcc 交叉编译
Debugger:	GNU gdb (GDB) 8.2
Architecture:	x64
OS:	Manjaro 18.0.0 Illyria
Kernel:	x86_64 Linux 4.19.1-1-MANJARO

3.2 软件构建过程

在 Linux 下使用 x86_64-w64-mingw32-g++构建，或者在 Windows 下使用 msys2 中的 mingw 交叉编译器构建。

3.2.1 编译生成

分别编译生成

lab2.o

lab3.o

main.o

3.2.2 链接文件

链接 lab2.o,lab3.o 和 main.o

4 软件测试

4.1 软件测试环境

Language:	C++
Text Editor:	Visual Studio Code 1.29.1
Compiler:	8.1.1 20180531 (GCC)
	Mingw-gcc 交叉编译
Architecture:	x64

OS: Manjaro 18.0.0 Illyria
Windows 10
Kernel: x86_64 Linux 4.19.1-1-MANJARO

4.2 软件测试内容

4.2.1 测试程序

图 4-1 软件运行测试截图

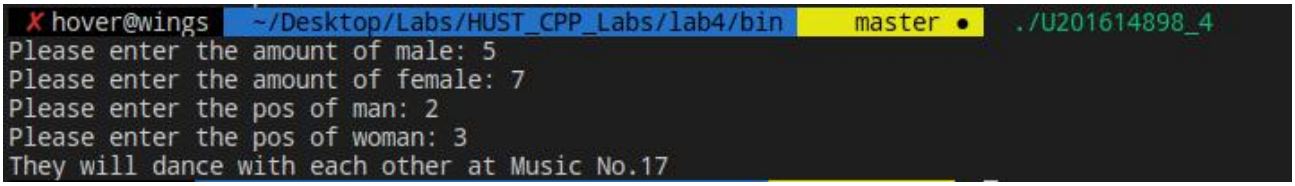

图 4-2 dance 测试截图 1

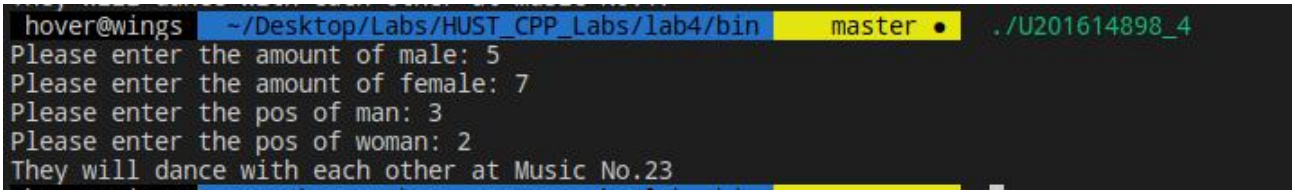

图 4-2 dance 测试截图 2

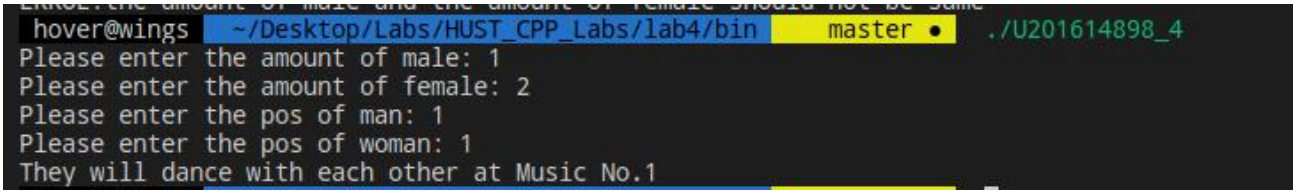

图 4-2 dance 测试截图 3

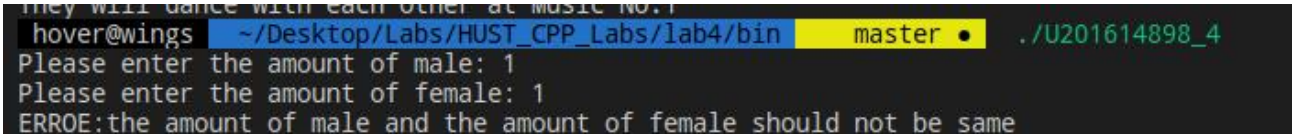

图 4-2 dance 测试截图 4

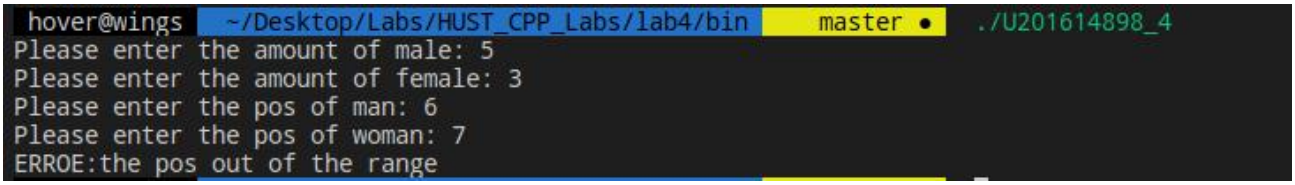

图 4-2 dance 测试截图 5

4.2.2 测试样例

正在测试 C:/msys64/HUST_CPP_Labs/lab4/bin/U201614898_4.exe.....

执行命令：C:/msys64/HUST_CPP_Labs/lab4/bin/U201614898_4.exe -S 5 -I 1 2 3 4 -O 2 -O 2

用户输出:S 5 I 1 2 3 4 O 3 4 O

标准输出:S 5 I 1 2 3 4 O 3 4 O

答案正确！

执行命令：C:/msys64/HUST_CPP_Labs/lab4/bin/U201614898_4.exe -S 5 -O 0 -I 1 2 3 4 -O 5 -I 1

用户输出:S 5 O I 1 2 3 4 O E

标准输出:S 5 O I 1 2 3 4 O E

答案正确！

执行命令：C:/msys64/HUST_CPP_Labs/lab4/bin/U201614898_4.exe -S 5 -I 1 2 3 4 -O 2 -I 5 6 7 -I 8

用户输出:S 5 I 1 2 3 4 O 3 4 I 3 4 5 6 7 I 3 4 5 6 7 8

标准输出:S 5 I 1 2 3 4 O 3 4 I 3 4 5 6 7 I 3 4 5 6 7 8

答案正确！

执行命令：C:/msys64/HUST_CPP_Labs/lab4/bin/U201614898_4.exe -S 5 -I 1 2 3 4 -O 2 -A 4 -I 5 6 -I 7 -I 8

用户输出:S 5 I 1 2 3 4 O 3 4 A 3 4 I 3 4 5 6 I 3 4 5 6 7 I 3 4 5 6 7 8

标准输出:S 5 I 1 2 3 4 O 3 4 A 3 4 I 3 4 5 6 I 3 4 5 6 7 I 3 4 5 6 7 8

答案正确！

执行命令：C:/msys64/HUST_CPP_Labs/lab4/bin/U201614898_4.exe -S 5 -I 1 2 3 4 -O 2 -C -I 5 6 -A 2
用户输出:S 5 I 1 2 3 4 O 3 4 C 3 4 I 3 4 5 6 A 3 4 5 6
标准输出:S 5 I 1 2 3 4 O 3 4 C 3 4 I 3 4 5 6 A 3 4 5 6
答案正确！

执行命令：C:/msys64/HUST_CPP_Labs/lab4/bin/U201614898_4.exe -S 5 -I 1 2 3 4 -O 2 -N
用户输出:S 5 I 1 2 3 4 O 3 4 N 2
标准输出:S 5 I 1 2 3 4 O 3 4 N 2
答案正确！

执行命令：C:/msys64/HUST_CPP_Labs/lab4/bin/U201614898_4.exe -S 5 -I 1 2 3 4 -G 3 -G 7
用户输出:S 5 I 1 2 3 4 G 4 G E
标准输出:S 5 I 1 2 3 4 G 4 G E
答案正确！

执行命令：C:/msys64/HUST_CPP_Labs/lab4/bin/U201614898_4.exe -S 5 -I 1 2 3 4 -G 3 -I 5 6 7 8 -O 3 -I 9 0 -G 6 -I 1
用户输出:S 5 I 1 2 3 4 G 4 I 1 2 3 4 5 6 7 8 O 4 5 6 7 8 I 4 5 6 7 8 9 0 G 0 I E
标准输出:S 5 I 1 2 3 4 G 4 I 1 2 3 4 5 6 7 8 O 4 5 6 7 8 I 4 5 6 7 8 9 0 G 0 I E
答案正确！

执行命令：C:/msys64/HUST_CPP_Labs/lab4/bin/U201614898_4.exe -S 3 -I 1 2 3 -O 1 -I 5 6 -G 1 -G 6
用户输出:S 3 I 1 2 3 O 2 3 I 2 3 5 6 G 3 G E
标准输出:S 3 I 1 2 3 O 2 3 I 2 3 5 6 G 3 G E
答案正确！

执行命令：C:/msys64/HUST_CPP_Labs/lab4/bin/U201614898_4.exe -S 3 -I 1 2 3 4 -G 1 -I 5 6 -G 5 -O 6 -O 1

用户输出:S 3 I 1 2 3 4 G 2 I 1 2 3 4 5 6 G 6 O O E
标准输出:S 3 I 1 2 3 4 G 2 I 1 2 3 4 5 6 G 6 O O E
答案正确!

共 10 个测试样例
正确个数：10
错误个数：

5 特点与不足

5.1 技术特点

1) 使用宏开关进行 debug 调试生成 debug 版本，在正式 release 版本时关闭宏开关，不进行调试信息输出，从而达到实验要求。
2) 提供公共的函数接口达到整个实验系统通用，对于不同的可执行文件，执行不同的 stack_main 或者 queen_main，从而达到代码的复用。
3) 对于不同的操作系统进行了判断，从而达到跨平台兼容。
4) 代码命名规范，采用驼峰命名法，且注释规范

5.2 不足和改进的建议

1) 数据结构元素为整型，未采用模板进行实现，故栈的可复用性不高
2) 数据结构使用的正确性约束由程序进行

6 过程和体会

6.1 问题

请说明如果删掉 virtual 有什么不同？
如果删掉 virtual 在继承之后的同名函数进行动态绑定的时候无法根据函数指针类型进行绑定，virtual 能够很好的实现多态，对于父类指针进行

6.2 遇到的主要问题和解决方法

采用 C++语言进行开发，过程较为简单，主要为了体会继承，交叉编译遇到一些麻烦，因为某些 Linux API 在存在环境依赖，故需要将依赖打包。

6.3 课程设计的体会

运用状态机的思想对进栈过程进行建模，同时进行正确性约束，能够较为充分的完成任务要求。

在文档阶段，借用 UML 图使接口清晰。继承在这种情况下感觉不太自然，因为继承通常为表示继承类(子类)，此题情境下为持有。

附录 源码和说明

文件清单及其功能说明

文件目录结构

采用驼峰命名法，所有的类名为当前类的数据结构类型

1) .h 文件为头文件用于函数和数据结构的声明
2) .cpp 文件用于函数的实现和相关数据的实现
3) UTF8 编码

- bin
  - 对应的二进制文件和所依赖的库打包
- include
  - 头文件：函数和数据结构的声明
- src
  - .cpp 文件：相应的函数和数据结构实现和接口调用
- obj
  - .obj 中间文件
- debugLog
  - debugLog：程序的输出校验文件
- MakeFile
  - MakeFile 文件

图 附录-1 文件目录结构

用户使用说明书

Make

cd lab3

make

源代码

lab3.h

```
/* FileName:lab3.h
 * Author:        Hover
 * E-Mail:        hover@hust.edu.cn
 * GitHub:        HoverWings
 * Description:   The definenation of STACK
 */
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include "my_debug.h"
int stack_main(int argc, char *argv[]) ;

class STACK
{
    int   *const   elems; //申请内存用于存放栈的元素
    const   int     max;   //栈能存放的最大元素个数
     // int     pos;              //栈实际已有元素个数，栈空时 pos=0;

    // PointerPropertyBuilderByName(int *const, elems, public)
    // PropertyBuilderByName(const int,max, public)
    // constPropertyBuilderByName(int,max,private)
    PropertyBuilderByName(int,pos,private)
public:
    STACK(int m);                           //初始化栈：最多存 m 个元素
    STACK(const STACK&s);                   //用栈 s 拷贝初始化栈
    virtual int    size ( ) const;          //返回栈的最大元素个数 max
    virtual bool    full ( ) const;         //返回栈是否满了
```

```
    virtual operator int ( ) const;              //返回栈的实际元素个数 pos
    virtual int operator[ ] (int x) const; //取下标 x 处的栈元素，第 1 个元素 x=0
    virtual STACK& operator<<(int e);          //将 e 入栈,并返回栈
    virtual STACK& operator>>(int &e);         //出栈到 e,并返回栈
    virtual STACK& operator=(const STACK&s);//赋 s 给栈,并返回被赋值的栈
    virtual void print( ) const;             //打印栈
    virtual ~STACK( );                              //销毁栈
    const int    getmax()
    {
        return this->max;
    }
};
```

lab4.h

```
/* FileName:lab4.h
 * Author:          Hover
 * E-Mail:          hover@hust.edu.cn
 * GitHub:          HoverWings
 * Description:   The definenation of QUEUE
 */
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include "my_debug.h"
#include "lab3.h"
int queue_main(int argc, char *argv[]);
int dance_main(int argc, char *argv[]);
class QUEUE:public STACK
{
    STACK s2;
    public:
    QUEUE(int m);                                   //每个栈最多 m 个元素,要求实现的队列最
多能入 2m 个元素
    QUEUE(const QUEUE&s);                       //用队列 s 拷贝初始化队列
    virtual operator int ( ) const;             //返回队列的实际元素个数
```

```
    virtual int full () const;                    //返回队列是否已满，满返回 1，否则返回 0
    virtual int size(void) const;                 //max size
    virtual int operator[ ](int x)const;        //取下标为 x 的元素，第 1 个元素下标为 0
    virtual QUEUE& operator<<(int e);             //将 e 入队列,并返回队列
    virtual QUEUE& operator>>(int &e);          //出队列到 e,并返回队列
    virtual QUEUE& operator=(const QUEUE&s);//赋 s 给队列,并返回被赋值的队列
    virtual int operator==(const QUEUE &q) const;
    virtual void print( ) const;                  //打印队列
    virtual ~QUEUE( );                                //销毁队列
    private:
   void stack_dump(STACK& src,STACK& dst);             //两个栈之间执行出入栈转换
    void stack_dump_to_s1();
    void stack_dump_to_s2();
};
```

my_debug.h

```
/* FileName:       my_debug.h
  * Author:          Hover
  * E-Mail:         hover@hust.edu.cn
  * GitHub:         HoverWings
  * Description:   The debug macro and the file redirect macro
  * Set/Get operation
  */
#include <stdio.h>

using namespace std;
// FILE *fd;//文件指针
// strcat(fname,".txt");
// fd=fopen(fname,"w+");



#define ECHO_COLOR_NONE              "\033[0;0m"
#define ECHO_COLOR_GREEN             "\033[0;32m"


#define __DEBUG
```

```
#undef __DEBUG

#ifdef __DEBUG
#define debug(fmt,args...)          \
            printf(ECHO_COLOR_GREEN "\nDebug: " fmt "\n" ECHO_COLOR_NONE, ##args);
// printf(ECHO_COLOR_GREEN "Debug: " fmt "(file: %s, func: %s, line: %d)\n"
ECHO_COLOR_NONE, ##args, __FILE__, __func__, __LINE__);
            // printf(ECHO_COLOR_GREEN fmt ECHO_COLOR_NONE);
            // printf(ECHO_COLOR_GREEN "Debug: " fmt "\n");

#else
#define debug(fmt, args...)
#endif

#ifdef __DEBUG
#define cdebug(fmt)    cout <<    fmt;
#else
#define cdebug(fmt)
#endif



// PropertyBuilderByName  用于生成类的成员变量
//  并生成 set 和 get 方法
// type  为变量类型
// access_permission  为变量的访问权限(public, priavte, protected)

#define PropertyBuilderByName(type, name, access_permission)\
      access_permission:\
            type name;\
      public:\
      inline void set##name(type v) {\
            name = v;\
      }\
      inline type get##name() {\
            return name;\
```

```
    }\

#define PointerPropertyBuilderByName(type, name, access_permission)\
    access_permission:\
        type* name;\
    public:\
        inline void set##name(type* v){\
            name = v;\
        }\
        inline type* get##name(){\
            return name;\
        }\

#define constPropertyBuilderByName(type, name, access_permission)\
    access_permission:\
        const type name;\
    public:\
    inline void set##name(type v) {\
        name = v;\
    }\
    inline type get##name() {\
        return name;\
    }\
```

osplatformutil.h

```
/* FileName:osplatformutil.h
 * Author:       Hover
 * E-Mail:       hover@hust.edu.cn
 * GitHub:       HoverWings
 * Description:  osplatformutil marco judgement
 */
#ifndef OSPLATFORMUTIL_H
#define OSPLATFORMUTIL_H

/*
    The operating system, must be one of: (I_OS_x)
```

```
        DARWIN      - Any Darwin system (macOS, iOS, watchOS, tvOS)
       ANDROID    - Android platform
       WIN32        - Win32 (Windows 2000/XP/Vista/7 and Windows Server 2003/2008)
       WINRT        - WinRT (Windows Runtime)
       CYGWIN      - Cygwin
       LINUX        - Linux
       FREEBSD    - FreeBSD
       OPENBSD    - OpenBSD
       SOLARIS    - Sun Solaris
       AIX            - AIX
        UNIX         - Any UNIX BSD/SYSV system
*/

#define OS_PLATFORM_UTIL_VERSION 1.0.0.180723

// DARWIN
#if defined(__APPLE__) && (defined(__GNUC__) || defined(__xlC__) || defined(__xlc__))
#    include <TargetConditionals.h>
#    if defined(TARGET_OS_MAC) && TARGET_OS_MAC
#        define I_OS_DARWIN
#        ifdef __LP64__
#            define I_OS_DARWIN64
#        else
#            define I_OS_DARWIN32
#        endif
#    else
#        error "not support this Apple platform"
#    endif
// ANDROID
#elif defined(__ANDROID__) || defined(ANDROID)
#    define I_OS_ANDROID
#    define I_OS_LINUX
// Windows
#elif !defined(SAG_COM) && (!defined(WINAPI_FAMILY) ||
WINAPI_FAMILY==WINAPI_FAMILY_DESKTOP_APP) && (defined(WIN64) ||
```

```
defined(_WIN64) || defined(__WIN64__))
#   define I_OS_WIN32
#   define I_OS_WIN64
#elif !defined(SAG_COM) && (defined(WIN32) || defined(_WIN32) || defined(__WIN32__) ||
defined(__NT__))
#   if defined(WINAPI_FAMILY)
#       ifndef WINAPI_FAMILY_PC_APP
#           define WINAPI_FAMILY_PC_APP WINAPI_FAMILY_APP
#       endif
#       if defined(WINAPI_FAMILY_PHONE_APP) &&
WINAPI_FAMILY==WINAPI_FAMILY_PHONE_APP
#           define I_OS_WINRT
#       elif WINAPI_FAMILY==WINAPI_FAMILY_PC_APP
#           define I_OS_WINRT
#       else
#           define I_OS_WIN32
#       endif
#   else
#       define I_OS_WIN32
#   endif
//CYGWIN
#elif defined(__CYGWIN__)
#   define I_OS_CYGWIN
// sun os
#elif defined(__sun) || defined(sun)
#   define I_OS_SOLARIS
// LINUX
#elif defined(__linux__) || defined(__linux)
#   define I_OS_LINUX
// FREEBSD
#elif defined(__FreeBSD__) || defined(__DragonFly__) || defined(__FreeBSD_kernel__)
#   ifndef __FreeBSD_kernel__
#       define I_OS_FREEBSD
#   endif
#   define I_OS_FREEBSD_KERNEL
// OPENBSD
```

```
#elif defined(__OpenBSD__)
#    define I_OS_OPENBSD
// IBM AIX
#elif defined(_AIX)
#    define I_OS_AIX
#else
#    error "not support this OS"
#endif

#if defined(I_OS_WIN32) || defined(I_OS_WIN64) || defined(I_OS_WINRT)
#    define I_OS_WIN
#endif

#if defined(I_OS_WIN)
#    undef I_OS_UNIX
#elif !defined(I_OS_UNIX)
#    define I_OS_UNIX
#endif

#ifdef I_OS_DARWIN
#define I_OS_MAC
#endif
#ifdef I_OS_DARWIN32
#define I_OS_MAC32
#endif
#ifdef I_OS_DARWIN64
#define I_OS_MAC64
#endif

#endif // OSPLATFORMUTIL_H
```

lab3.cpp

```
/* FileName:lab3.cpp
 * Author:      Hover
 * E-Mail:      hover@hust.edu.cn
 * GitHub:      HoverWings
```

```
 * Description:   The implementation of STACK
 */
#include "lab3.h"
#include <iostream>
#include <stdlib.h>
#include <cctype>
/*
 “设定栈队或队列大小-S”
 “入-I”、
 “出-O”、
 “深拷贝构造-C”、
 “深拷贝赋值-A”、
 “栈中剩余元素个数-N”
*/
using namespace std;

int stack_main(int argc, char *argv[])
{
    int num;//元素个数&&入栈数字
    int out;//接受出栈元素
    STACK *s;
    STACK *p;
    int ch;
    bool fail=false;
    while ((ch = getopt(argc, argv, "S:I:O:CA:NG:")) != -1)
    {
        if(fail)
        {
            cdebug("false");
            break;
        }
        fail=false;
        debug("optind: %d\n", optind);
        switch (ch)
        {
            case 'S':
```

```
        debug("HAVE option: -S");
        debug("The argument of -S is %s", optarg);
        num=atoi(optarg);
        debug("%d",num);
        printf("S    %d", num);
        s = new STACK(num);
        break;
    case 'I':
        debug("HAVE option: -I");
        debug("The argument of -I is %s", optarg);
        num=atoi(optarg);
        debug("%d",num);
        if(s->getpos()==s->getmax())
        {
            fail=true;
            printf("    I");
            printf("    E");
            break;
        }
        (*s)<<num;
        // debug(argv[optind][0]);
        while(isdigit(argv[optind][0]))
        {
            num=atoi(argv[optind]);
            debug("%d",num);
            if(s->getpos()==s->getmax())
            {
                fail=true;
                printf("    I");
                printf("    E");
                break;
            }
            (*s)<<num;
            optind++;
            if(optind==argc)
            {
```

```
                break;
            }
        }
        if(fail==false)
        {
            printf("    I");
            s->print();
        }
        break;
    case 'O':
        debug("HAVE option: -O");
        debug("The argument of -O is %s", optarg);
        num=atoi(optarg);
        debug("%d",num);
        for (int j = 0; j < num; j++)
        {
            if (int(*s)== 0)
            {
                printf("    O    E");
                exit(0);
            }
            (*s)>>out;
        }
        printf("    O");
        s->print();
        break;
    case 'C':
        debug("HAVE option: -C");
        printf("    C");
        p = s;
        s= p;
        s->print();
        break;
    case 'A':
        debug("HAVE option: -A");
        debug("The argument of -A is %s", optarg);
```

```
                num=atoi(optarg);
                printf("    A");
                p = new STACK(num);
                (*p)=*s;              //assign p to s
                s = p;
                s->print();                //print current stack
                break;
            case 'N':
                debug("HAVE option: -N");
                printf("    N");
                printf("    %d", int(*s));
                break;
            case 'G':
                debug("HAVE option: -G");
                debug("The argument of -G is %s", optarg);
                num=atoi(optarg);
                printf("    G");
                if(num>s->getpos())
                {
                    printf("    E");
                    break;
                }
                printf("    %d", (*s)[num]);
                break;
            default:
                debug("Unknown option: %c",(char)optopt);
                break;
            }

    }
    return 0;
}

//Impletation Stack Fun

//Overload
```

```
STACK::STACK(int m): elems(m > 0 ? new int[m] : new int[0]), max(m > 0 ? m : 0)
{
	this->pos = 0;
	for (int i = 0; i < this->size(); i++)
	{
		this->elems[i] = 0;
	}
}

STACK::STACK(const STACK &s): elems(s.max > 0 ? new int[s.max] : new int[0]), max(s.max > 0 ? s.max : 0)
{
	this->pos = 0;
	for (int i = 0; i < (int)s && i < this->size(); i++)
	{
		(*this)<<s[i];
	}
}

int STACK::size() const
{
	return this->max;
}

bool STACK::full() const
{
	return (this->max==(int)(this->pos));
}

STACK::operator int(void) const
{
	return (int)(this->pos);
}
```

```
int STACK::operator[](int x) const
{
	// out of range check
	if (x < 0 || x >= (int)(*this)) return 0;
	return this->elems[x];
}

STACK& STACK::operator<<(int e)
{
	// full check
	// if (this->size() <= (int)(*this)) return *this;
	cdebug(this->elems[this->pos]);
	this->elems[this->pos++] = e;
	return *this;
}

STACK& STACK::operator>>(int &e)
{
	// empty check
	if ((int)(*this) <= 0)
	{
		e = 0;
		return *this;
	}

	e = this->elems[--this->pos];
	return *this;
}

STACK& STACK::operator=(const STACK &s)
{
	this->~STACK();
	new (this) STACK(s);
	return *this;
}
```

```
void STACK::print(void) const
{
	for (int i = 0; i < (int)(*this); i++)
	{
		cout<<"    "<<(*this)[i];
	}
}

STACK::~STACK(void)
{
	delete this->elems;
	this->pos = 0;
}
```

lab4.cpp

```
/* FileName:lab4.cpp
 * Author:		Hover
 * E-Mail:		hover@hust.edu.cn
 * GitHub:		HoverWings
 * Description:	The implementation of QUEUE
 */
#include "lab4.h"
#include <iostream>
#include <cmath>
#include <stdlib.h>
#include <ctype.h>

/*
“设定栈队或队列大小-S”
“入-I”、
“出-O”、
“深拷贝构造-C”、
“深拷贝赋值-A”、
“栈中剩余元素个数-N”
```

```
*/
using namespace std;

int queue_main(int argc, char *argv[])
{
    int num;      //元素个数&&入队数字
    int out;      //接受出队元素
    QUEUE *q;
    QUEUE *p;
    int ch;
    bool fail=false;
    while ((ch = getopt(argc, argv, "S:I:O:CA:NG:")) != -1)
    {
        if(fail)
        {
            debug("false");
            break;
        }
        fail=false;
        debug("optind: %d\n", optind);
        switch (ch)
        {
            case 'S':
                debug("HAVE option: -S");
                debug("The argument of -S is %s", optarg);
                num=atoi(optarg);
                debug("%d",num);
                printf("S    %d", num);
                q = new QUEUE(num);
                // printf("S");
                // q->print();
                break;
            case 'I':
                debug("HAVE option: -I");
                debug("The argument of -I is %s", optarg);
                num=atoi(optarg);
```

```
debug("%d",num);
if(q->full())
{
    fail=true;
    printf("    I");
    printf("    E");
    break;
}
(*q)<<num;
  if(optind==argc)
  {
      printf("    I");
      q->print();
      break;
  }
// debug(argv[optind][0]);
while(isdigit(argv[optind][0]))
{
    num=atoi(argv[optind]);
    debug("%d",num);
    if(q->full())
    {
        fail=true;
        printf("    I");
        printf("    E");
        break;
    }
    (*q)<<num;
    optind++;
    if(optind==argc)
    {
        break;
    }
}
printf("    I");
q->print();
```

```
        break;
    case 'O':
        debug("HAVE option: -O");
        debug("The argument of -O is %s", optarg);
        num=atoi(optarg);
        debug("%d",num);
        for (int j = 0; j < num; j++)
        {
            if (int(*q)== 0)
            {
                printf("    O    E");
                exit(0);
            }
            (*q)>>out;
        }
        printf("    O");
        q->print();
        break;
    case 'C':
        debug("HAVE option: -C");
        printf("    C");
        p = q;
        q->print();
        break;
    case 'A':
        debug("HAVE option: -A");
        debug("The argument of -A is %s", optarg);
        num=atoi(optarg);
        printf("    A");
        q->print();
        break;
    case 'N':
        debug("HAVE option: -N");
        printf("    N");
        printf("    %d", int(*q));
        break;
```

```
            case 'G':
                debug("HAVE option: -G");
                debug("The argument of -G is %s", optarg);
                num=atoi(optarg);
                if(num>int(*q))
                {
                    printf("    G    E");
                    break;
                }
                printf("    G");
                printf("    %d", (*q)[num]);
                break;
            default:
                debug("Unknown option: %c",(char)optopt);
                break;
            }

    }
    return 0;
}
//判断素数，是素数返回 true
bool is_prime(int n)
{
    for(int i=2; i<=sqrt(n); i++)
    {
         if(n%i == 0)
        {
            cout << n << " isn't a prime" << endl;
            return false;
         }
     }
     return true;
}


int dance_main(int argc, char *argv[])
```

```
{
    //male and female num
    int M = 0, F = 0;

    //male and female queue
    QUEUE *que_m=nullptr;
    QUEUE *que_f=nullptr;

    //male and female position
    int pos_m = 0;
    int pos_f = 0;
    int count = 0;

    cout << "Please enter the amount of male: ";
    cin >> M;
    cout << "Please enter the amount of female: ";
    cin >> F;
    if (!is_prime(M))
    {
        cerr << "ERROE:the Input number (amount of male) is not prime number!" << endl;
        return -1;
    }
    if (!is_prime(F))
    {
        cerr << "ERROE:the Input number (amount of female) is not prime number!" << endl;
        return -1;
    }
    if(M==F)
    {
        cerr << "ERROE:the amount of male and the amount of female should not be same" <<
endl;
        return -1;
    }
    cout << "Please enter the pos of man: ";
    cin >> pos_m;
    cout << "Please enter the pos of woman: ";
```

```
cin >> pos_f;
if (!(pos_m <= M&&pos_f <= F))
{
    cerr << "ERROE:the pos out of the range" << endl;
    return -1;
}
// new queue
que_m = new QUEUE(M);
que_f = new QUEUE(F);
for (int i = 0; i < M; i++)
{
    (*que_m) << (i + 1);
}
for (int i = 0; i < F; i++)
{
    (*que_f) << (i + 1);
}
int out_m=0;
int out_f=0;
for (int i = 0;; i++)
    {
        count++;
        (*que_m)>>out_m;
        (*que_f)>>out_f;
        if (out_m == pos_m&&out_f == pos_f)
        {
            cout << "They will dance with each other at Music No." << count << endl;
            delete que_m;
            delete que_f;
            que_m=nullptr;
            que_f=nullptr;
            return 0;
        }
        (*que_m) << out_m;
        (*que_f) << out_f;
    }
```

```
}

//Impletation QUEUE Fun
QUEUE::QUEUE(int m): STACK(m), s2(m)
{

}

QUEUE::QUEUE(const QUEUE &q): STACK(q), s2(q.s2)
{

}

int QUEUE::size() const
{
      return STACK::size();
}

QUEUE::operator int(void) const
{
      return STACK::operator int() + (int)s2;
}

int QUEUE::operator[](int x) const
{
//    assert(0<=x&&x<=int(*this)-1);
     if(0<=STACK::operator int()&&STACK::operator int()<=STACK::size()-1&&s2.empty())
//state 1
     {

          if(x<=STACK::operator int())
          {
               return STACK::operator [](x);
          }
     }
     if(0==STACK::operator int()&&s2.full()) //state 2
```

```
    {
        if(x<=int(s2))
        {
            return s2[s2.getpos()-x-1];
        }
        else
         {
             return STACK::operator [](x-int(s2));
         }
    }
    if(1<=STACK::operator int()&&STACK::operator
int()<=STACK::size()&&1<=int(s2)&&int(s2)<=STACK::size()) //state 3
    {
         if(x<=int(s2)-1)
         {
             return s2[s2.getpos()-x-1];
         }
         else
         {
             return STACK::operator [](x-int(s2));
         }
    }
     if(STACK::full()&&1<=int(s2)&&int(s2)<=s2.size()) //state 4
    {
        if(x<=int(s2))
        {
            return s2[s2.getpos()-x-1];
        }
        else
        {
            return STACK::operator [](x-int(s2));
        }
    }
}

QUEUE& QUEUE::operator<<(int e)
```

```
{
    debug("s1:%d",STACK::operator int());
    debug("s2:%d",int(s2));
    debug("insert:%d",e);
     // // full check
     // if (this->size() <= (int)(*this)) return *this;
    if(0<=STACK::operator int()&&STACK::operator int()<=STACK::size()-1&&int(s2)==0)
//state 1
    {
         debug("state 1");
        STACK::operator <<(e); // s1 full
        if(STACK::full())
        {
            stack_dump_to_s2();
//                  stack_dump_to_s2(); // to state 2
            debug("s1:%d",STACK::operator int());
            debug("s2:%d",int(s2));
            return *this;
        }
        debug("s1:%d",STACK::operator int());
        debug("s2:%d",int(s2));
        return *this;
    }
    if(0==STACK::operator int()&&s2.full()) //state 2
    {
         debug("state 2");
        STACK::operator <<(e); // to state 3
        debug("s1:%d",STACK::operator int());
        debug("s2:%d",int(s2));
        return *this;
    }
    if(1<=STACK::operator int()&&STACK::operator
int()<=STACK::size()-1&&1<=int(s2)&&int(s2)<=s2.size()) //state 3
    {
         debug("state 3");
        STACK::operator <<(e);
```

```
            debug("s1:%d",STACK::operator int());
            debug("s2:%d",int(s2));
            if(STACK::full())
            {
                //to state 4
                return *this;
            }
            return *this;
        }
        if(STACK::full()&&1<=int(s2)&&int(s2)<=s2.size()) //state 4
        {
             debug("state 4");
            return *this;
        }
        cout<<"error!";
        return *this;
}


QUEUE& QUEUE::operator>>(int &e)
{
        debug("s1:%d",STACK::operator int());
        debug("s2:%d",int(s2));
        if(0<=STACK::operator int()&&STACK::operator int()<=STACK::size()-1&&s2.empty())
//state 1
        {
            debug("state 1");
            stack_dump_to_s2();
            s2>>e;
            stack_dump_to_s1();// to state 1
        }
        if(0==STACK::operator int()&&s2.full()) //state 2
        {
            s2>>e; // to state 1
            stack_dump_to_s1();
            return *this;
```

```
    }
    if(1<=STACK::operator int()&&STACK::operator
int()<=STACK::size()&&1<=int(s2)&&int(s2)<=STACK::size()) //state 3
    {
        s2>>e;
        if(s2.empty())
        {
            if(0<=STACK::operator int()&&STACK::operator int()<=STACK::size()-1) //to state 1
            {

            }
            if(STACK::full()) //to state 2
            {
                stack_dump_to_s2();
            }
        }
        return *this;
    }
     if(STACK::full()&&1<=int(s2)&&int(s2)<=s2.size()) //state 4
    {
        s2>>e;
        if(s2.empty())      // to state 1
        {
            stack_dump_to_s2();
        }
        return *this;
    }
     return *this;
}

int QUEUE::full() const
{
    // if(STACK::full()&&1<=int(s2)&&int(s2)<=s2.size())
    // {
    //     return 1;
    // }
```

```
    // else
    // {
    //     return 0;
    // }
    if(0<=int(*this)&&int(*this)<=this->size()-1&&s2.empty()) //state 1
    {
        return false;
    }
    if(0==int(*this)&&s2.full()) //state 2
    {
        return false;
    }
    if(1<=int(*this)&&int(*this)<=this->size()&&1<=int(s2)&&int(s2)<=this->size()) //state 3
    {
        return false;
    }
    if(this->full()&&1<=int(s2)&&int(s2)<=s2.size()) //state 4
    {
        return true;
    }
}

QUEUE& QUEUE::operator=(const QUEUE &q)
{
     this->~QUEUE();
     new (this) QUEUE(q);
     return *this;
}

int QUEUE::operator==(const QUEUE &q) const
{
    // size or pos should equal
    if (this->size() != q.size() || (int)(*this) != (int)q)
        return 0;
    // every single element should equal
    for (int i = 0; i < (int)(*this); i++)
```

```
    {
        if ((*this)[i] != q[i]) return 0;
    }

    return 1;
}

void QUEUE::print() const
{
    for (int i = 0; i < (int)(*this); i++)
    {
        cout<<"    "<<(*this)[i];
    }
    // cout<<"\n";
}

QUEUE::~QUEUE()
{
    // stack member will be destructed automatically when queue vanish
}


void QUEUE::stack_dump(STACK& src,STACK& dst)
{
    // push all elements of src into dst
    if ((int)src <= dst.size()-int(dst))
    {
        int elem;
        while ((int)src)
        {
            src>>elem;
            dst<<elem;
        }
    }
}
```

```
void QUEUE::stack_dump_to_s2()
{
    if (STACK::operator int() <= s2.size()-int(s2))
    {
        int elem;
        while (STACK::operator int())
        {
            STACK::operator >>(elem);
            s2<<elem;
        }
    }
}

void QUEUE::stack_dump_to_s1()
{
    if (int(s2)<= STACK::size()-STACK::operator int())
    {
        int elem;
        while ((int)s2)
        {
            s2>>elem;
            STACK::operator <<(elem);
        }
    }
}
```

main.cpp

```
/* FileName:main.cpp
 * Author:        Hover
 * E-Mail:        hover@hust.edu.cn
 * GitHub:        HoverWings
 * Description:   The main of program and call the interface funciton math the OS
 */
#include <iostream>
#include <unistd.h>
#include <string.h>
#include "my_debug.h"
#include "lab3.h"
#include "osplatformutil.h"

using namespace std;


//Helper Function
const char *find_file_name(const char *name)
{
    char *name_start = NULL;
    int sep = '/';
    if (NULL == name)
     {
         printf("the path name is NULL\n");
        return NULL;
    }
    name_start = (char *)strrchr(name, sep);
    return (NULL == name_start)?name:(name_start + 1);
}

//Judge Funciton


int main(int argc, char *argv[])
{
```

```
#if defined I_OS_LINUX
debug("this is linux");
// cout<<argv[0];
const char* file_name;
file_name=find_file_name(argv[0]);
if(strcmp(file_name, "U201614898_1") == 0)
{
	debug("stack!");
	stack_main(argc,argv);
}
else if(strcmp(file_name, "U201614898_2") == 0)
{
	debug("stack!");
	stack_main(argc,argv);
}
else if(strcmp(file_name, "U201614898_3") == 0)
{
	debug("queue!");
	stack_main(argc,argv);
}
else if(strcmp(file_name, "U201614898_4") == 0)
{
	debug("queue!");
	queue_main(argc,argv);
}
#elif defined I_OS_WIN32
cout<<"this is windows"<<endl;
#elif defined I_OS_CYGWIN
debug("this is cygwin");
  const char* file_name;
file_name=find_file_name(argv[0]);
if(strcmp(file_name, "U201614898_1") == 0)
{
	debug("stack!");
	stack_main(argc,argv);
}
```

```
    else if(strcmp(file_name, "U201614898_2") == 0)
    {
        debug("stack!");
        stack_main(argc,argv);
    }
    else if(strcmp(file_name, "U201614898_3") == 0)
    {
        debug("queue!");
        stack_main(argc,argv);
    }
    else if(strcmp(file_name, "U201614898_4") == 0)
    {
        debug("queue!");
        queue_main(argc,argv);
    }
    #endif
    return 0;
}
```

MakeFile

```
SOURCES = $(shell find . -path ./test -prune -o -name "*.cpp" -print)
OBJECTS = $(patsubst %.cpp, %.o, $(SOURCES))
C_OBJ = $(patsubst %.o, $(OBJ_DIR)/%.o, $(notdir $(OBJECTS)))
PROG=U201614898_4

# macro for tools
# CC = x86_64-w64-mingw32-g++
CC = g++
RM = rm -fr
MV = mv
CP = cp -fr
MKDIR = mkdir -p
# BROWSER = google-chrome

# macro for flags
```

```
FLAGS = -c -Wall -g $(addprefix -I, $(INCLUDE))

# path macro
BIN_DIR = ./bin
OBJ_DIR = ./obj



# include macro
INCLUDE += ./include/

all: $(OBJECTS) link

.cpp.o:
	@echo Compiling C Source Files $< ...
	$(MKDIR) $(OBJ_DIR)
	$(CC) $(FLAGS) $< -o $@
	$(MV) $@ $(OBJ_DIR)/$(notdir $@)

link:
	@echo Linking Binary File
	$(MKDIR) $(BIN_DIR)
	$(CC) $(C_OBJ) -o $(BIN_DIR)/$(PROG)
	@echo
	@echo Build Success!



.PHONY:run
run:
	./bin/$(PROG)
```