江苏海洋大学计算机工程学院

实验报告书

课程名称： 《数据结构》

实验名称： 实验2：栈和队列及其应用

第3章 实习题3

班 级： 计科1211

学 号： 2019120573

姓 名： 王胡鑫

评语：

成绩： 指导教师：

批阅时间： 年 月 日

1. 目的与要求

**目的**：通过本实验，学生应能理解栈和队列抽象数据类型的表示及实现方法，栈和队列的应用。

**要求**：

1. 应用已学的栈及队列的基本算法，用C/C++语言编程实现实习题，源程序中必需对算法、程序功能等加适当的注释；
2. 用适当的测试数据对程序进行调试，并生成正常运行的可执行文件；
3. 编写符合内容完整、格式规范、独立完成的实验报告；
4. 将实现的源程序、可执行文件、实验报告电子稿压缩成包，以自己姓名命名，在规定时间内提交到超星泛雅教学平台上。
5. 实验内容与题目

**内容**：实现栈和队列的常用算法。

**题目**：第3章 实习题3

1. 实验步骤与源程序

#include <stdio.h>

#include <stdlib.h>

#include <time.h>

#include <stdbool.h>

#include <Windows.h>

#define MAXSIZE 100

typedef struct

{

char name[50];

int year;

int month;

int day;

} GoodsInfo; //定义商品信息类型

typedef struct

{

GoodsInfo goods[MAXSIZE];

int top; //指向栈顶的下一位下标（空栈==0）（满栈==MAXSIZE）

} SeqStack; //定义商品货架、零时货架类型

typedef struct

{

GoodsInfo goods[MAXSIZE];

int front, rear; // front队头下标，rear对尾下标的下一位

} SeqQueue; //定义商品上货顺序队列类型

//程序状态

#define OK 1

#define ERROR 0

#define TRUE 1

#define FALSE 0

typedef int Status;

//基本函数

void MenuPrint(void); //功能菜单打印函数

void ScanfGoodsInfo(GoodsInfo &g); //输入商品信息函数

void PrintGoodsInfo(GoodsInfo gp); //打印商品信息函数

//队列函数

Status InitQueue(SeqQueue &Q); //初始化清空商品队列

int QueueLength(SeqQueue Q); //求当前商品队列中的商品数量

Status EnQueue(SeqQueue &Q); //商品入队

Status DeQueue(SeqQueue &Q, GoodsInfo &g); //商品出队

void PrintQueueGoods(SeqQueue Q); //打印队列中所用商品（从队头到队尾）

Status GetHead(SeqQueue Q, GoodsInfo &g); //获取队头元素

//栈函数

Status InitStack(SeqStack &S); //初始化清空商品货架

int StackLength(SeqStack S); //求当前商品货架中的商品数量

Status Push(SeqStack &S, GoodsInfo g); //商品压入货架

Status Pop(SeqStack &S, GoodsInfo &g); //商品弹出货架

void PrintStackGoods(SeqStack S); //打印货架中所用商品（从栈顶到栈底）

Status GetTop(SeqStack S, GoodsInfo &g); //获取栈顶元素

//关联函数

//队头日期 > 栈顶日期，返回TRUE；（需要倒栈）

//队中无商品，栈中无商品，队头日期 <= 栈顶日期，返回FALSE；（不需要倒栈）

bool CompareDate(SeqQueue Q, SeqStack S); //比较队头和栈顶商品的日期

bool CompareDate\_2(GoodsInfo Q, SeqStack S);

//将货架S1中的货物压入货架S2中

Status ChangeGoods(SeqStack &S1, SeqStack &S2);

//将货物队列Q中的队头商品加入到商品货架中

Status AddGoods(SeqQueue &Q, SeqStack &S1, SeqStack &S2);

Status AddGoods\_2(SeqQueue &Q, SeqStack &S1, SeqStack &S2);

//依次输入要加入货架的商品信息

void InputGoods(SeqQueue &Q);

//取出货架中的第一件商品

void GetOutFirstGoods(SeqStack &S);

//取出货架中的第n件商品

void GetOutNGoods(SeqStack &S1, SeqStack &S2);

int main(void)

{

SeqStack GoodsStack, TempStack;

SeqQueue GoodsQueue;

InitStack(GoodsStack);

InitStack(TempStack);

InitQueue(GoodsQueue);

int i;

do

{

MenuPrint();

printf("\n\n");

printf("\*请输入您的选择(0 - 8):");

scanf("%d", &i);

printf("\n\n");

switch (i)

{

case 1:

InputGoods(GoodsQueue);

break;

case 2:

PrintQueueGoods(GoodsQueue);

break;

case 3:

PrintStackGoods(GoodsStack);

break;

case 4:

AddGoods\_2(GoodsQueue, GoodsStack, TempStack);

break;

case 5:

InitQueue(GoodsQueue);

break;

case 6:

InitStack(GoodsStack);

break;

case 7:

GetOutFirstGoods(GoodsStack);

break;

case 8:

GetOutNGoods(GoodsStack, TempStack);

break;

case 0:

printf("正在退出该系统，欢迎下次使用！\n\n");

for (int i = 0; i < 30; i++)

{

Sleep(150);

printf("<<");

}

system("cls");

break;

default:

printf("请输入正确的选项！\n\n");

system("pause");

system("cls");

;

break;

}

} while (i != 0);

return 0;

}

void ScanfGoodsInfo(GoodsInfo &g) //输入商品信息函数

{

printf("请输入商品信息：\n");

printf("商品名称：");

scanf("%s", &g.name);

getchar();

printf("生产年份：");

scanf("%d", &g.year);

getchar();

printf("生产月份：");

scanf("%d", &g.month);

getchar();

printf("生产日期：");

scanf("%d", &g.day);

printf("\n商品录入完成！\n\n");

}

void PrintGoodsInfo(GoodsInfo g) //打印商品信息函数

{

printf("商品名称：%s\n", g.name);

printf("生产年份：%d\n", g.year);

printf("生产月份：%d\n", g.month);

printf("生产日期：%d\n", g.day);

putchar('\n');

}

Status InitQueue(SeqQueue &Q) //初始化清空商品队列

{

Q.front = Q.rear = 0;

return OK;

}

int QueueLength(SeqQueue Q) //求当前商品队列中的商品数量

{

return (Q.rear + MAXSIZE - Q.front) % MAXSIZE;

}

Status EnQueue(SeqQueue &Q) //商品入队

{

if ((Q.rear + 1) % MAXSIZE == Q.front)

return ERROR;

ScanfGoodsInfo(Q.goods[Q.rear]);

Q.rear = (Q.rear + 1) % MAXSIZE;

return OK;

}

Status DeQueue(SeqQueue &Q, GoodsInfo &g) //商品出队

{

if (Q.front == Q.rear)

return ERROR;

g = Q.goods[Q.front];

Q.front = (Q.front + 1) % MAXSIZE;

return OK;

}

void PrintQueueGoods(SeqQueue Q) //打印队列中所用商品（从队头到队尾）

{

printf("货物队列中共有%d件货物。\n\n", QueueLength(Q));

for (int i = 0; i < QueueLength(Q); i++)

{

printf("第%d件货物的信息：\n", i + 1);

PrintGoodsInfo(Q.goods[(Q.front + i) % MAXSIZE]);

putchar('\n');

}

}

Status GetHead(SeqQueue Q, GoodsInfo &g) //获取队头元素

{

if (Q.front == Q.rear)

return ERROR;

g = Q.goods[Q.front];

return OK;

}

Status InitStack(SeqStack &S) //初始化清空商品货架

{

S.top = 0;

return OK;

}

int StackLength(SeqStack S) //求当前商品货架中的商品数量

{

return S.top;

}

Status Push(SeqStack &S, GoodsInfo g) //商品压入货架

{

if (S.top == MAXSIZE)

return ERROR;

S.goods[S.top] = g;

S.top++;

return OK;

}

Status Pop(SeqStack &S, GoodsInfo &g) //商品弹出货架

{

if (S.top == 0)

return ERROR;

S.top--;

g = S.goods[S.top];

return OK;

}

void PrintStackGoods(SeqStack S) //打印货架中所用商品（从栈顶到栈底）

{

printf("商品货架中共有%d件货物。\n\n", StackLength(S));

int i = S.top, j = 1;

while (i > 0)

{

printf("第%d件货物的信息：\n", j++);

PrintGoodsInfo(S.goods[i - 1]);

putchar('\n');

i--;

}

}

Status GetTop(SeqStack S, GoodsInfo &g) //获取栈顶元素

{

if (S.top == NULL)

return ERROR;

g = S.goods[S.top - 1];

return OK;

}

//队头日期 > 栈顶日期，返回TRUE；（需要倒栈）

//队中无商品，栈中无商品，队头日期 <= 栈顶日期，返回FALSE；（不需要倒栈）

bool CompareDate(SeqQueue Q, SeqStack S) //比较队头和栈顶商品的日期

{

GoodsInfo qh, st;

if (!(GetHead(Q, qh) && GetTop(S, st)))

return FALSE; //队空或栈空

int i = qh.year \* 366 + qh.month \* 31 + qh.day;

int j = st.year \* 366 + st.month \* 31 + st.day;

if (i <= j)

return FALSE;

else

return TRUE;

// else if (GetHead(Q, qh) && GetTop(S, st))

// {

// if (qh.year > st.year) return TRUE;

// if (qh.year < st.year) return FALSE;

// if (qh.year == st.year)

// {

// if (qh.month > st.month) return TRUE;

// if (qh.month < st.month) return FALSE;

// if (qh.month == st.month)

// {

// if (qh.day > st.day) return TRUE;

// if (qh.day <= st.day) return FALSE;

// }

// }

// }

}

//队头日期 > 栈顶日期，返回TRUE；（需要倒栈）

//队中无商品，栈中无商品，队头日期 <= 栈顶日期，返回FALSE；（不需要倒栈）

bool CompareDate\_2(SeqQueue Q, SeqStack S)

{

if (StackLength(S) == 0)

return TRUE;

GoodsInfo qh, st;

GetHead(Q, qh);

GetTop(S, st);

int i = qh.year \* 366 + qh.month \* 31 + qh.day;

int j = st.year \* 366 + st.month \* 31 + st.day;

if (i <= j)

return FALSE;

else

return TRUE;

}

//将货架S1中的货物压入货架S2中

Status ChangeGoods(SeqStack &S1, SeqStack &S2)

{

GoodsInfo g;

Pop(S1, g);

Push(S2, g);

return OK;

}

//将货物队列Q中的队头商品加入到商品货架S1中

Status AddGoods(SeqQueue &Q, SeqStack &S1, SeqStack &S2)

{

while (QueueLength(Q) > 0)

{

if (!CompareDate(Q, S1)) //不需要倒货

{

GoodsInfo g1;

DeQueue(Q, g1);

Push(S1, g1);

}

else if (CompareDate(Q, S1)) //需要倒货

{

while (CompareDate(Q, S1))

{

ChangeGoods(S1, S2);

}

GoodsInfo g2;

DeQueue(Q, g2);

Push(S1, g2);

}

while (StackLength(S2) > 0)

{

GoodsInfo g3;

ChangeGoods(S2, S1);

}

}

return OK;

}

//优化加入货架的函数

Status AddGoods\_2(SeqQueue &Q, SeqStack &S1, SeqStack &S2)

{

while (QueueLength(Q) > 0)

{

while (CompareDate(Q, S1)) //倒货

{

ChangeGoods(S1, S2);

}

GoodsInfo g;

if (CompareDate\_2(Q, S2))

{

DeQueue(Q, g);

Push(S1, g);

}

if (!CompareDate\_2(Q, S2))

{

ChangeGoods(S2, S1);

}

}

while (StackLength(S2) > 0)

{

GoodsInfo g3;

ChangeGoods(S2, S1);

}

return OK;

}

//依次输入要加入货架的商品信息

void InputGoods(SeqQueue &Q)

{

char choice = 'y';

while (choice == 'Y' || choice == 'y')

{

EnQueue(Q);

getchar();

printf("是否继续输入商品信息？(Y\\N)：");

choice = getchar();

while (!(choice == 'Y' || choice == 'y' || choice == 'N' || choice == 'n'))

{

getchar();

printf("您的输入不合理！请重新输入（Y\\N）：");

choice = getchar();

}

if (choice == 'N' || choice == 'n')

break;

}

}

//取出货架中的第一件商品

void GetOutFirstGoods(SeqStack &S)

{

GoodsInfo g;

Pop(S, g);

printf("您取出的商品信息是：\n");

PrintGoodsInfo(g);

}

//取出货架中的第n件商品

void GetOutNGoods(SeqStack &S1, SeqStack &S2)

{

int n;

printf("您要取第几件商品？（目前总共有%d件商品）\n", StackLength(S1));

if (n <= StackLength(S1))

{

scanf("%d", &n);

for (int i = 0; i < n - 1; i++)

{

ChangeGoods(S1, S2);

}

GoodsInfo g;

Pop(S1, g);

printf("您取出的商品信息：\n");

PrintGoodsInfo(g);

for (int i = 0; i < n - 1; i++)

{

ChangeGoods(S2, S1);

}

}

}

//保存当前货架中的商品信息

//查看已保存的商品信息

//清空已保存的商品信息

//功能菜单打印函数

void MenuPrint(void)

{

printf("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

printf("\* 功能菜单： \*\n");

printf("\* 1、输入需要放入货架的商品信息。 \*\n");

printf("\* 2、查看当前需要放入货架的商品队列信息。 \*\n");

printf("\* 3、查看当前货架中的商品信息。 \*\n");

printf("\* 4、将商品队列中的商品放入商品货架中。 \*\n");

printf("\* 5、清空需要放入货架的商品队列。 \*\n");

printf("\* 6、清空当前货架中的商品。 \*\n");

printf("\* 7、取出货架中的第一件商品。 \*\n");

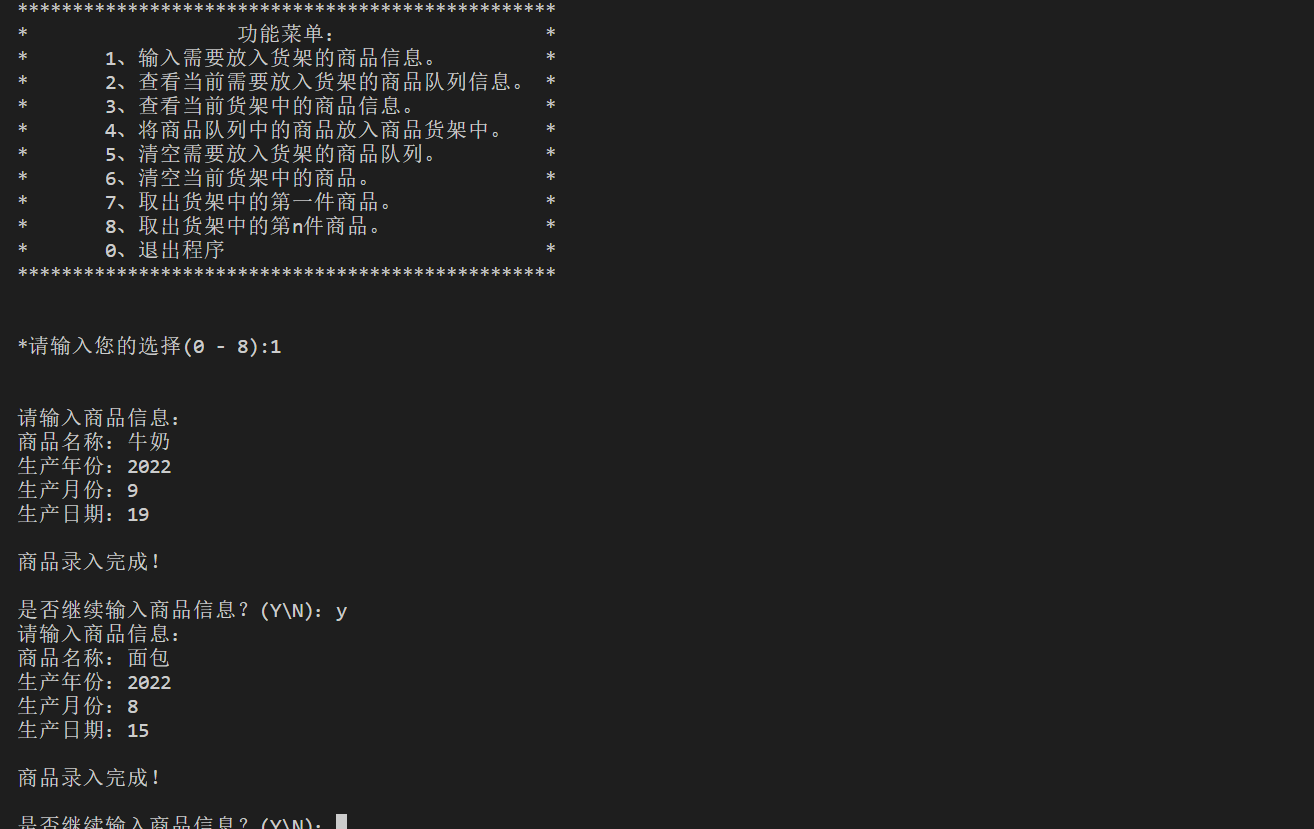
printf("\* 8、取出货架中的第n件商品。 \*\n");

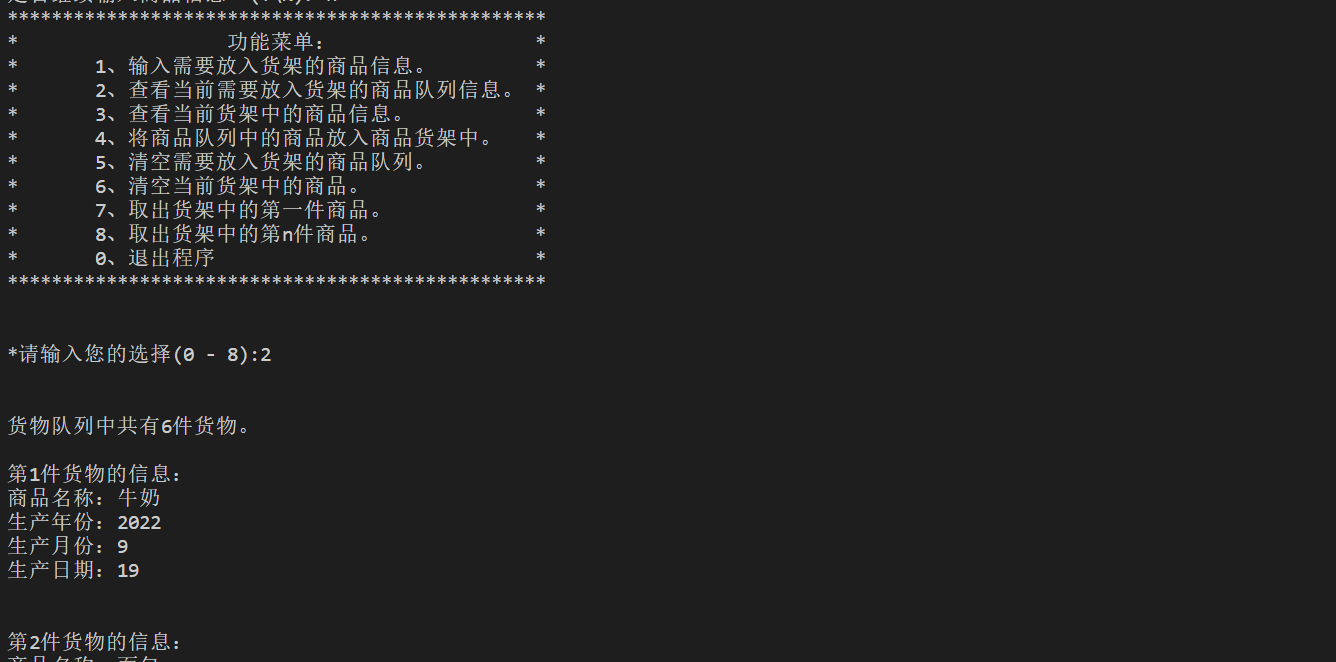
printf("\* 0、退出程序 \*\n");

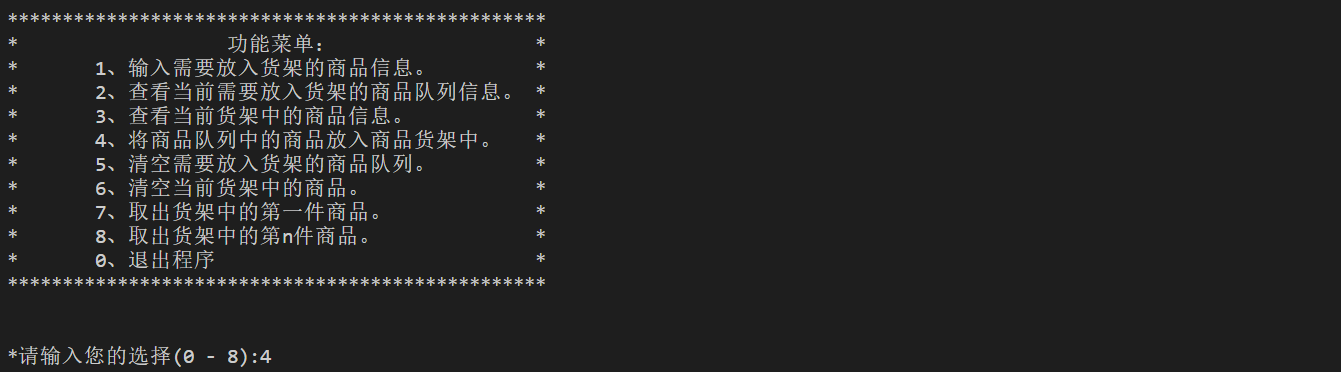
printf("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

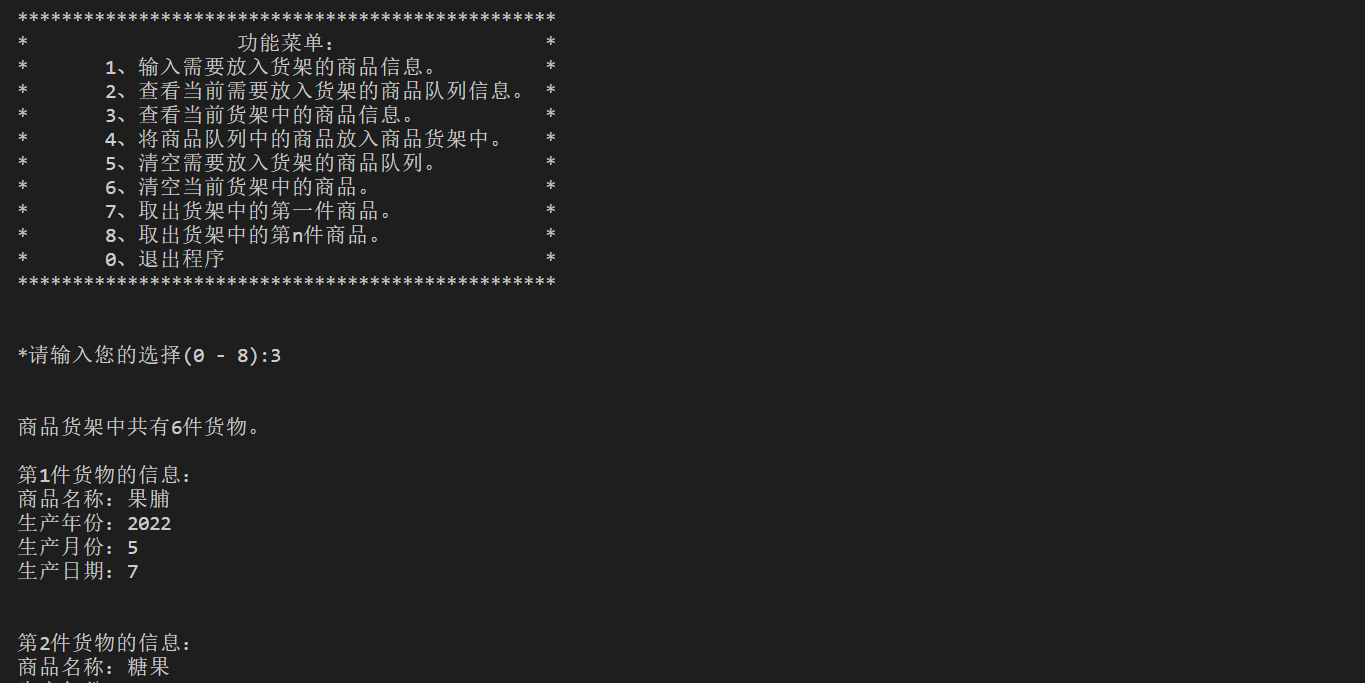
}

1. 测试数据与实验结果

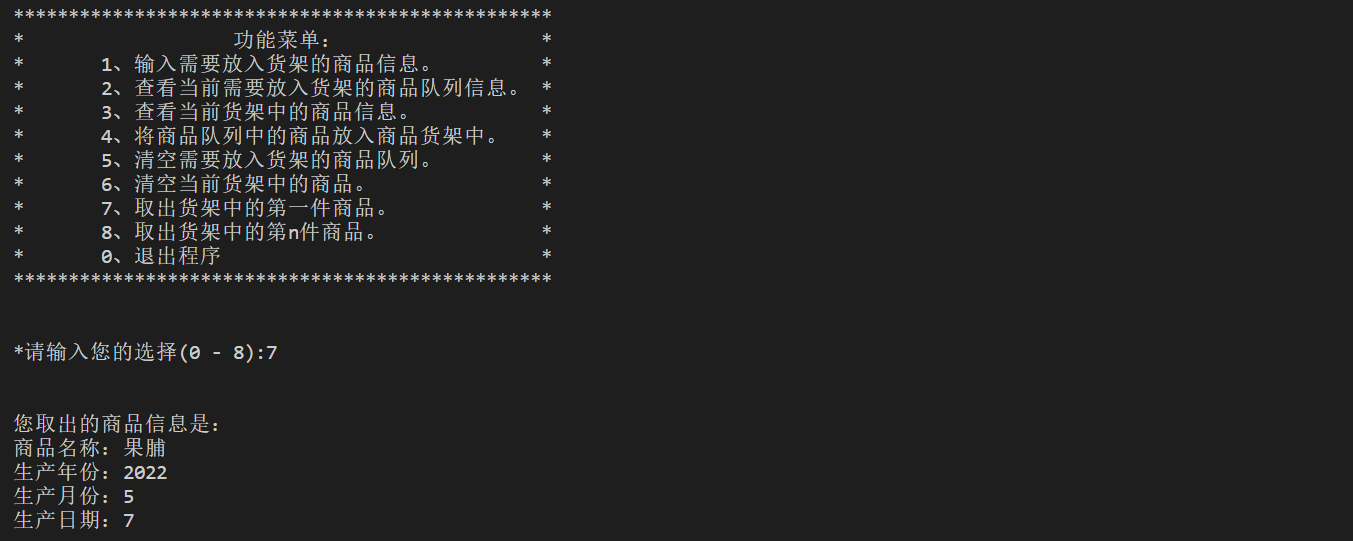


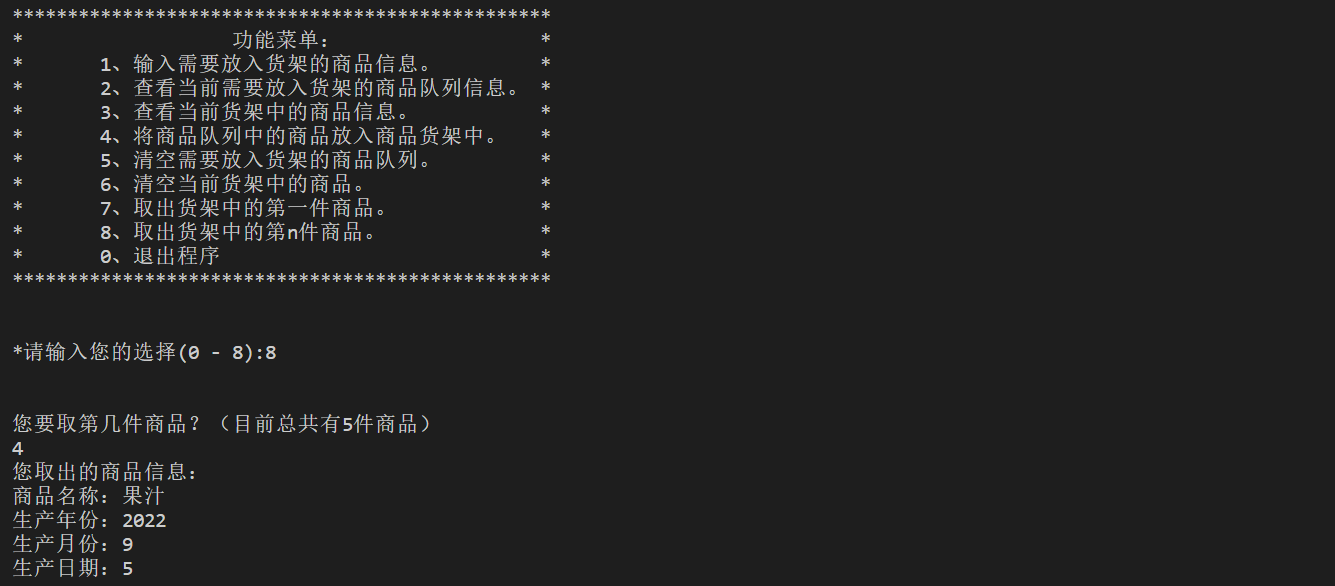


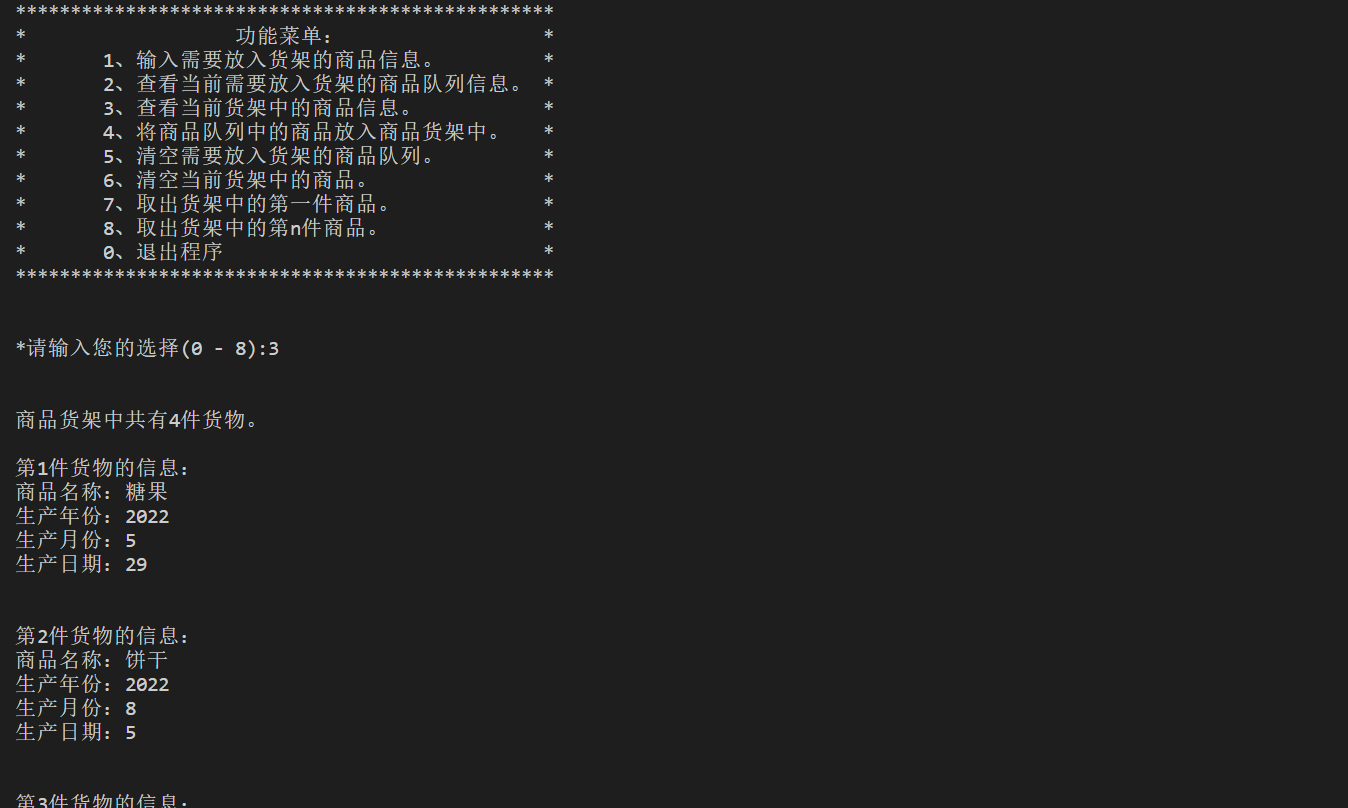


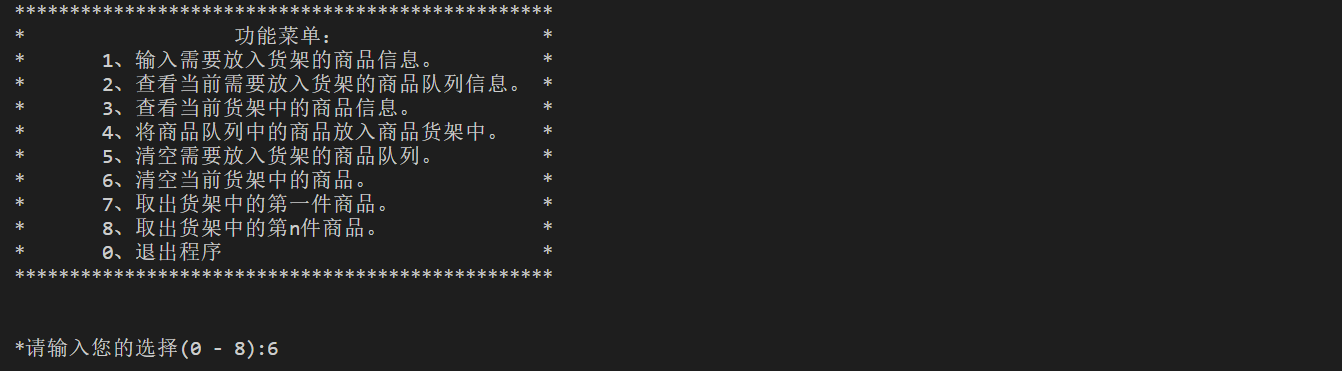


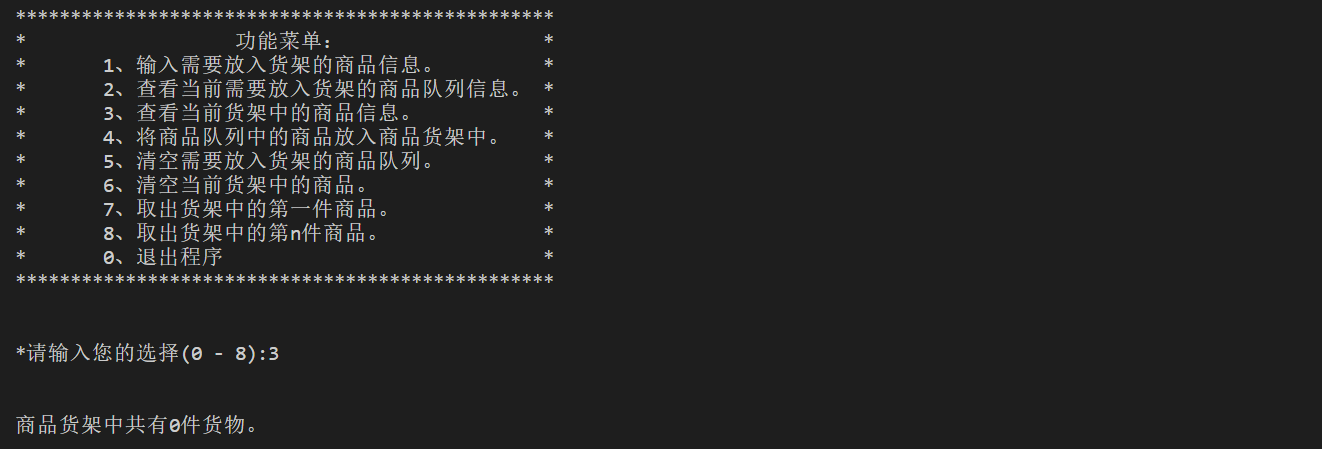


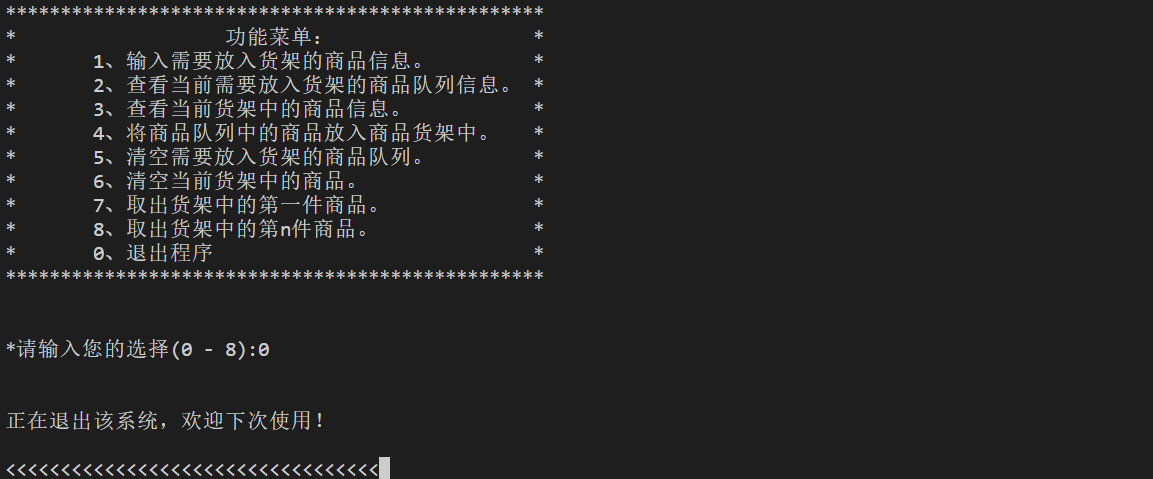












1. 结果分析与实验体会

尽管基本功能均可实现，但是程序的健壮性不强，面对不合理输入时总是存在各种问题；

诸多问题如算法不优，代码冗余；