西安交通大学实验报告

课程名称： 算法设计与问题求解 实验名称： 栈和队列

学 院： 机械工程学院 实验日期 2020 年 10 月 31 日

班 级： 机械97班 姓 名： 杨逢诜 学号： 2193712613

#### 一、实验内容和结果

* **题目1**

读入一个后缀表达式，仅含有单个正整数和+、\*运算。中间用空格分割，以#号表示结束。输出计算表达式结果。例如：

6 5 2 3 + 8 \* + 3 + \* #

Result = 288

【源程序】

#include<stdio.h>

struct Sqstack //定义结构体：栈，包含栈内元素和栈长两个数据

{

int data[100];

int top;

};

void Push(struct Sqstack \*stack,int x) //声明函数：入栈

{

if(stack->top==99) //栈满，输出提示并禁止输入。

{

printf("ERROR,CAN'T PUT IT IN");

}

else //栈不满，则栈长自增，栈顶指针处填入新元素

{

stack->top++;

stack->data[stack->top]=x;

}

}

int Pop(struct Sqstack \*stack) //声明函数：出栈

{

if(stack->top==-1) //若栈空，输出提示并禁止输出

{

printf("ERROR,CAN'T PUT IT IN");

return 0;

}

else //若栈不空，调出栈顶元素并使栈长自减。

{

stack->top--;

return stack->data[stack->top+1];

}

}

int main() //主函数开始

{

struct Sqstack stack; //定义栈和栈指针

struct Sqstack \*pstack;

pstack=&stack;

pstack->top=-1; //置空栈

char note; //定义字符，并输入第一个字符。

scanf("%c",&note);

while(note!='#') //若上一个输入的字符不是终止符，则一直持续循环：

{

if(note>='0'&&note<='9') //若输入的元素是数字，将数字压入栈

{

Push(pstack,(int)(note-'0'));

}

else if(note=='+') //若读取的元素是算符+，将栈顶两个数字弹出并执行加和操作

{

int i,j;

i=Pop(pstack);

j=Pop(pstack);

Push(pstack,i+j); //将加和运算结果压入栈

}

else if(note=='\*') //若读取的元素是算符\*，将栈顶两个数字弹出并执行乘积操作

{

int i,j;

i=Pop(pstack);

j=Pop(pstack);

Push(pstack,i\*j); //将乘积运算结果压入栈

}

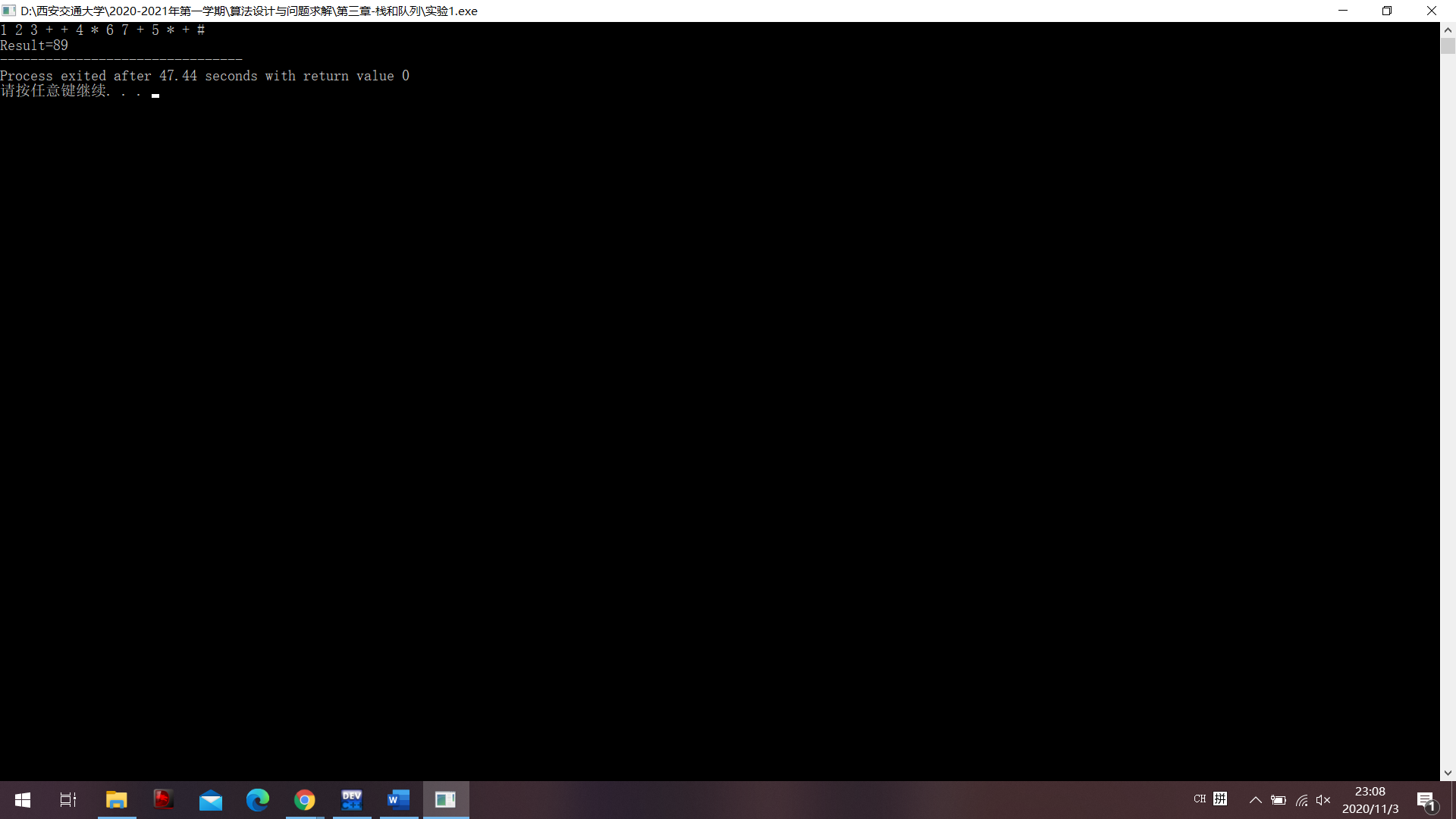
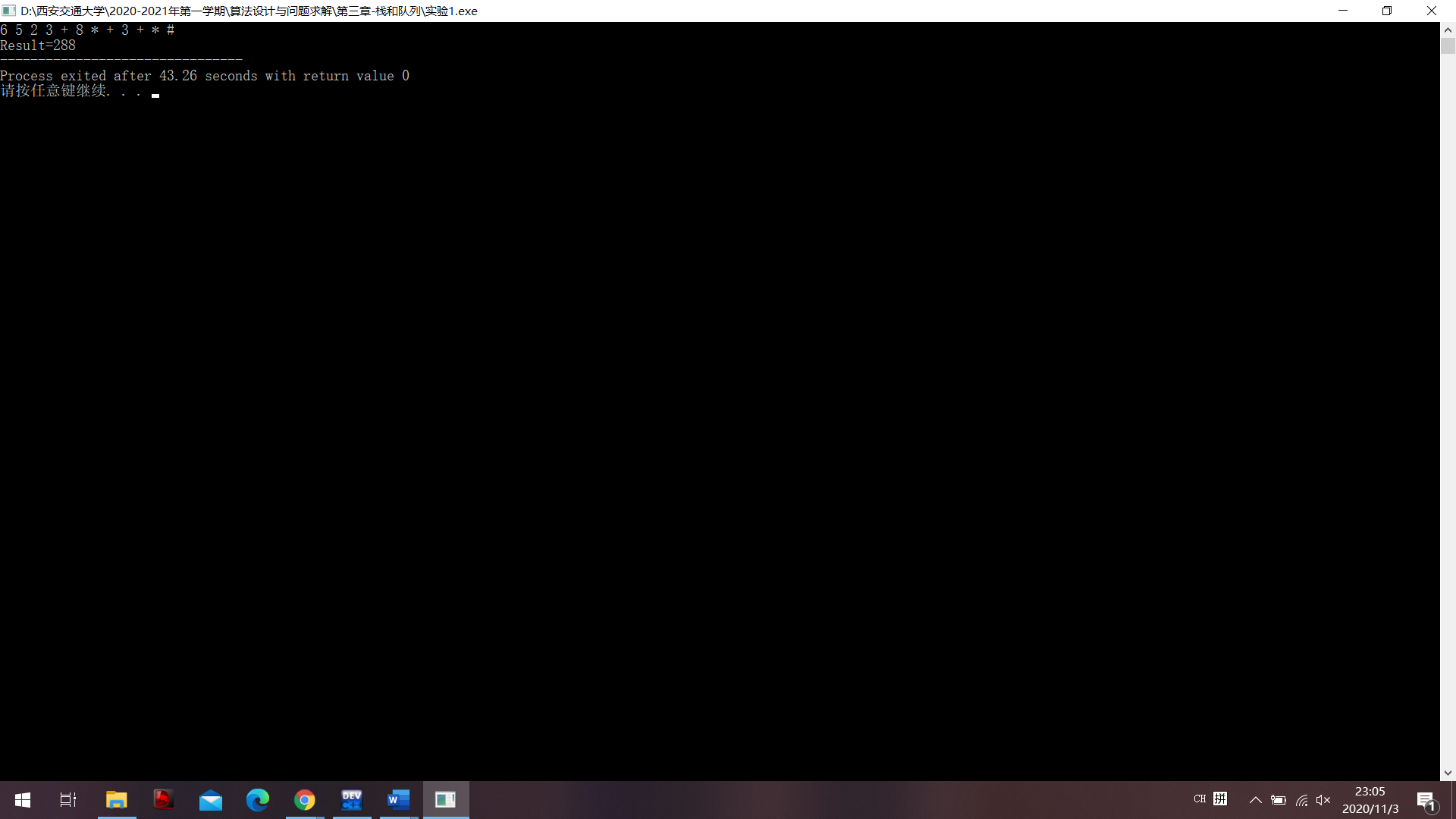
scanf("%c",&note); //继续读取下一个位点的字符

}

printf("Result=%d",Pop(pstack)); //将栈中剩余的最后一个元素弹出并输出，主函数结束。

}

【运行结果】



* **题目2**

用char数组建立顺序队列，实现并验证下面算法

1）入队

【算法】

void EnQueue(SqQueue \*Q , char x)

{

if(队尾加一取模MAX等于队头)

输出"队列已满"的信息；

else {

队尾后移一位； // 队尾=队尾加1取模MAX

队尾的数据设置为x；

}

}

2）出队

【算法】

char DeQueue(SqQueue \*Q )

{

if(队尾等于队头) {

输出"队列已空"信息；

返回 ’#’； // 假定返回 # 代表无数据

} else {

队头后移一位； // 加一取模MAX

返回队头数据；

}

}

3）运行下面测试主函数

// 主函数

int main()

{

SqQueue \*queue;

queue = new SqQueue; //创建队列

queue->rear=queue->front=0;

EnQueue(queue, 'A');

EnQueue(queue, 'B');

EnQueue(queue, 'C');

//一个元素出队,并显示

printf("出队：%c\n", (char)DeQueue(queue));

//再入队3个后打印队列

EnQueue(queue, 'D');

EnQueue(queue, 'E');

EnQueue(queue, 'F');

//所有元素依次出队直到队空

while (queue->front != queue->rear)

printf("出队：%c\n", (char)DeQueue(queue));

return 0;

}

【源程序】

#include<stdio.h>

#include<iostream>

int MAX=100; //定义队列最大长度

struct SqQueue //定义结构体：队列

{

char alphabets[100];

int rear; //定义队头指针和队尾指针

int front;

};

struct SqQueue\* CreateQueue() //定义函数：创建队列

{

struct SqQueue \*Q=new SqQueue; //为队列安排内存空间并将队头、队尾指针置空，返回地址

Q->rear=0;

Q->front=0;

return Q;

}

void EnQueue(struct SqQueue \*Q,char c) //声明函数：入队

{

if((Q->rear+1)%MAX==Q->front%MAX) //若队满（队头和队尾指针取余后仅仅差1）则输出提示并禁止输入。

{

printf("QUEUE FULL,CAN'T GET IN");

}

else //若队不满，则将元素安排入队列中，队尾指针自增。

{

Q->rear++;

Q->alphabets[(Q->rear-1)%MAX]=c;

}

}

char DeQueue(struct SqQueue \*Q) //声明函数：出队

{

if((Q->front)%MAX==(Q->rear)%MAX)

{

printf("QUEUE EMPTY,CAN'T PUT OUT"); //若队空（队头和队尾指针取余后相等）则输出提示并禁止输出。

return 0;

}

else //若队不空，则将队头元素弹出，队头指针自增。

{

Q->front++;

return Q->alphabets[(Q->front-1)%MAX];

}

}

int main() //主函数开始

{

struct SqQueue \*queue;

queue = new SqQueue; //创建队列

queue->rear=queue->front=0;

EnQueue(queue, 'A');

EnQueue(queue, 'B');

EnQueue(queue, 'C'); //一个元素出队,并显示

printf("出队：%c\n", (char)DeQueue(queue)); //再入队3个后打印队列

EnQueue(queue, 'D');

EnQueue(queue, 'E');

EnQueue(queue, 'F'); //所有元素依次出队直到队空，同时输出这些元素

while (queue->front != queue->rear)

printf("出队：%c\n", (char)DeQueue(queue));

return 0; //主函数结束

}

【运行结果】

