**南京师范大学中北学院**

**《数据结构》**

**实**

**验**

**报**

**告**

**姓 名: 钟玮**

**学 号: 85213241**

**班 级: 计算机二班**

**日 期: 2024年3月26日**

**顺序栈的基本操作及简单应用**

1. **实验目的**

**1、掌握顺序栈的定义；**

**2、掌握顺序栈的基本操作，如初始化、判空、获得栈顶元素、入栈、出栈、遍历等，及栈的简单应用：进制转化。**

1. **实验内容**

**1、初始化一个顺序栈；**

**2、判断顺序栈是否为空；**

**3、获得栈顶元素，存放在e中；**

**4、压一个元素进栈；**

**5、弹出栈顶元素,存放到e中；**

**6、遍历栈，从栈底到栈顶对每个元素进行打印；**

**7、用顺序栈实现：对于输入的任意一个非负十进制整数，打印输出与其等值的八进制数**

1. **实验主要代码**

**可以截图并做简要解释，如：**

* 1. **初始化一个顺序栈**

1. // 定义栈的结构体
2. typedef struct
3. {
4. SElemType \*base; // 栈底指针
5. SElemType \*top; // 栈顶指针
6. int stacksize; // 当前已分配的存储空间，以元素为单位
7. } *SqStack*;
8. // 初始化栈
9. *Status* InitStack(*SqStack* &*S*)
10. {
11. // 为栈分配初始空间
12. S.base = (SElemType \*)malloc(STACK\_INIT\_SIZE \* sizeof(SElemType));
13. // 如果分配失败，程序退出
14. if (!S.base)
15. exit(OVERFLOW);
16. // 初始化栈顶指针和栈底指针
17. S.top = S.base;
18. // 初始化栈的大小
19. S.stacksize = STACK\_INIT\_SIZE;
20. return OK;
21. }

**2、判断顺序栈是否为空**

// 判断栈是否为空

*Status* StackEmpty(*SqStack* *S*)

{

// 如果栈顶指针等于栈底指针，说明栈为空

if (S.top == S.base)

return TRUE;

else

return FALSE;

}

**3、获得栈顶元素，存放在e中**

*Status* GetTop(*SqStack* *S*, *SElemType* &*e*)

{

// 如果栈为空，返回错误

if (S.top == S.base)

return ERROR;

// 否则，将栈顶元素赋值给e

e = \*(S.top - 1);

return OK;

}

**4、压一个元素进栈**

*Status* Push(*SqStack* &*S*, *SElemType* *e*)

{

// 如果栈满，需要重新分配空间

if (S.top - S.base >= S.stacksize)

{

S.base = (SElemType \*)realloc(S.base, (S.stacksize + STACKINCREMENT) \* sizeof(SElemType));

if (!S.base)

exit(OVERFLOW);

S.top = S.base + S.stacksize;

S.stacksize += STACKINCREMENT;

}

// 将元素e入栈

\*S.top++ = e;

return OK;

}

**5、弹出栈顶元素,存放到e中**

*Status* Pop(*SqStack* &*S*, *SElemType* &*e*)

{

// 如果栈为空，返回错误

if (S.top == S.base)

return ERROR;

// 否则，将栈顶元素出栈

e = \*--S.top;

return OK;

}

**6、遍历栈，从栈底到栈顶对每个元素进行打印**

void StackTraverse(*SqStack* *S*)

{

// 如果栈为空，打印提示信息

if (S.base == S.top)

{

printf("当前栈为空栈\n");

}

else

{

// 如果栈不为空，打印出从栈底到栈顶的所有元素

printf("当前从栈底到栈顶的元素为：\n");

// 使用while循环遍历栈，直到栈顶指针等于栈底指针

while (S.top > S.base)

// 打印出当前元素，并将栈底指针向上移动一位

printf("%d ", \*S.base++);

// 打印换行符，使输出更美观

printf("\n");

}

}

**7、用顺序栈实现：对于输入的任意一个非负十进制整数，打印输出与其等值的八进制数**

void conversion(int *Num*)

{

// 定义一个元素e用于存储出栈的元素

SElemType e;

// 定义一个栈S

SqStack S;

// 初始化栈S

InitStack(S);

// 当Num不为0时，执行循环

while (Num)

{

// 将Num除以8的余数入栈

Push(S, Num % 8);

// 更新Num为Num除以8的商

Num = Num / 8;

}

// 当栈不为空时，执行循环

while (!StackEmpty(S))

{

// 将栈顶元素出栈，并存储到e中

Pop(S, e);

// 打印出栈的元素

printf("%d", e);

}

// 打印换行符，使输出更美观

printf("\n");

}

**8、main函数**

int main()

{

*SqStack* s;

*SElemType* e1, e2;

int i;

InitStack(s);

for (i = 1; i <= 5; i++)

Push(s, rand() % 40);

StackTraverse(s);

GetTop(s, e1);

printf("当前栈顶元素为：%d\n", e1);

printf("\n");

printf("元素逐个出栈：\n");

while (!StackEmpty(s))

{

Pop(s, e2);

printf("当前出栈元素为：%d\n", e2);

StackTraverse(s);

}

printf("\n");

int num;

printf("请输入需要转化的非负十进制整数：\n");

scanf("%d", &num);

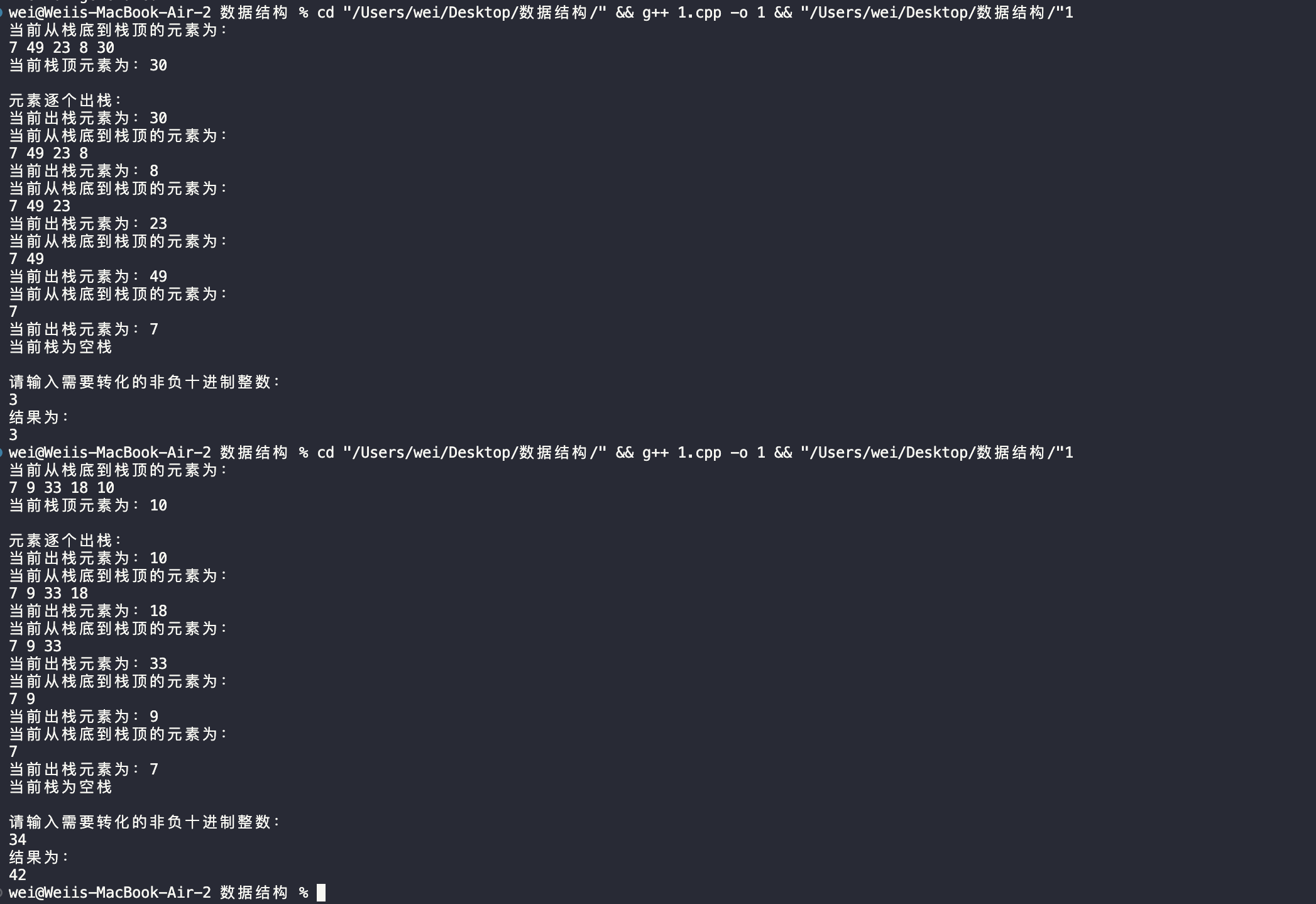
printf("结果为：\n");

conversion(num);

}

1. **实验结果**

**实验运行结果的截图，如main函数运行结果截图：**

****