数据结构第2次作业

评分

满分—3分

学号：XXXXXXXXXX 姓名：张某某 专业：XXXXXXXXX

知识范畴：堆栈、队列、串

提交截止日期：20XX年XX月XX日

作业内容：

♦ 栈与队列

3.3 3.4 3.13

3.9 3.15 3.25

上机作业

1. 输入一个非零正整数，输出其各位数字。要求数字之间间隔至少一个空格。

例如：输入12085, 输出为1 2 0 8 5。

要求：采用递归和非递归(用堆栈)两种算法；输入的整数不允许按单个字符输入。

1. 编写程序，实现中缀表达式化为后缀式输出。已知中缀表达式中操作数全部用小写英文字母表示，运算符只含有+, -, \*, /四种，定界符只有( )以及结束符#。

♦ 字符串

4.7 4.11

上机作业

1. 从键盘输入主串s以及子串t1, t2，试将主串s中所有t1子串替换为t2子串，输出替换后得到的串以及t1被替换的次数。要求子串查找采用改进KMP算法。

作业解答：

3.3

输出结果: stack

3.4

algo1: 将栈内的元素逆置

algo2: 删除栈内与元素e相等的元素

3.13

algo3: 将队列内的元素逆置

3.9

void digui(int n)

{

    int i;

    i = n;

    if (i > 1)

        return printf("%d", i), digui(n - 1);

}

3.15

void inistack(twS &s) //初始化

{

    s.elem = (elemT \*)malloc(InitSize \* sizeof(elemT));

    if (!s.elem) //内存分配失败

    {

        printf("Bad alloc!\n");

        exit(-1);

    }

    s.size = InitSize;

    s.topl = 0, s.topr = s.size;

}

void push(twS &s, int i, const elemT &x)

{

    if (s.topr == s.topl) //如果发生上溢则扩容

        resize(s);

    if (i)

        s.elem[s.topr--] = x;

    else

        s.elem[s.topl++] = x;

}

void pop(twS &s, int i, elemT &x)

{

    if (i && s.topr == s.size || !i && !s.topl) //发生下溢报错

    {

        printf("Out of Range!\n");

        exit(-1);

    }

    if (i)

        x = s.elem[++s.topr];

    else

        x = s.elem[--s.topl];

}

按过程设计算法过程简练, 但可读性差.

按函数设计算法可以高效复用代码, 并且可读性高, 但过程繁琐.

3.25

int f(int n) //递归法

{

    return n ? n \* f(n / 2) : 1;

}

以下是消除递归的算法:

stk s;

int a[10000];

initStk(s), push(s, n);

memset(a, -1, sizeof a);

a[0] = 1;

while (!isEmpty(s))

{

int tmp = pop(s);

    if (a[tmp] != -1)

        continue;

    push(s, tmp);

    if (a[tmp / 2] == -1)

      push(s, tmp / 2);

    else

        a[tmp] = tmp \* a[tmp / 2];

}

经过测试结果正确

第三章上机作业1:

关键代码(源文件为exp-3-1.cpp):

stk s;

initStk(s);

int tmp = n;

while (tmp)

{

    push(s, tmp % 10);

    tmp /= 10;

}

printf("\n消除递归结果:\n");

while (!isEmpty(s))

    printf("%5d", pop(s));

第三章上机作业2:

关键代码(源文件为exp-3-2.cpp):

int main()

{

    printf("请输入中缀表达式:\n");

    stk optr;

    initStk(optr);

    char ch, fml[100], tmp;

    for (int i = 0;;)

    {

        ch = getchar();

        if (ch == ' ' || ch == '\n')

            continue;

        if (ch == '#')

            break;

        fml[i++] = ch;

    }

    int n = strlen(fml);

    for (int i = 0; i < n; ++i)

    {

        ch = fml[i];

        if (isalpha(ch))

            printf("%c", ch);

        else

        {

            if (ch == '(')

                push(optr, ch);

            else if (ch == ')')

            {

                ch = pop(optr);

                while (ch != '(')

                {

                    printf("%c", ch);

                    ch = pop(optr);

                }

            }

            else

            {

                if (isEmpty(optr))

                    push(optr, ch);

                else

                {

                    if (compare(tmp = pop(optr), ch))

                    {

                        printf("%c", tmp);

                        push(optr, ch);

                    }

                    else

                    {

                        push(optr, tmp);

                        push(optr, ch);

                    }

                }

            }

        }

    }

    while (!isEmpty(optr))

        printf("%c", pop(optr));

}

4.7

s:’a, a, a, b’

next:’-1, 0, 1, 2’

nextval:’-1, -1, -1, 2’

t:‘a, b, c, a, b, a, a’

next:’-1, 0, 0, 0, 1, 2, 1’

nextval:’-1, 0, 0, -1, 0, 2, 1’

u: ‘a, b, c, a, a, b, b, a, b, c, a, b, a, a, c, b, a, c, b, a’

next:’-1, 0, 0, 0, 1, 1, 2, 0, 1, 2, 3, 4, 2, 1, 1, 0, 0, 1, 0, 0’

nextval:’-1, 0, 0, -1, 1, 0, 2, -1, 0, 0, -1, 4, 2, 1, 1, 0, -1, 1, 0, -1’

4.11

关键代码:

int main()

{

    StringType s, t, r;

    StrInit(s), StrInit(t), StrInit(r);

    StrInput(s), StrInput(t);

    int alpha[26];

    char tmp[26];

    memset(alpha, -1, sizeof alpha);

    for (int i = 0; i < StrLength(s); ++i)

        if (alpha[s.idx[i] - 'a'] == -1)

            alpha[s.idx[i] - 'a'] = i;

    for (int i = 0; i < StrLength(t); ++i)

        alpha[t.idx[i] - 'a'] = -1;

    for (int i = 0, cnt = 0; i < 26; ++i)

        if (alpha[i] != -1)

            printf("字符%c在s中第一次出现的位置为:%d\n", tmp[cnt++] = i + 'a', alpha[i] + 1);

    StrAssignwithChars(r, tmp);

    StrTraverse(r);

    return 0;

}

第四章上机作业:

关键代码(源文件为exp-4-1.cpp):

void getNext(const str &s, int nextval[])

{

    int j = 0, k = -1;

    nextval[0] = -1;

    while (s.idx[j])

    {

        if (k == -1 || s.idx[j] == s.idx[k])

        {

            ++j, ++k;

            if (s.idx[j] != s.idx[k])

                nextval[j] = k;

            else

                nextval[j] = nextval[k];

        }

        else

            k = nextval[k];

    }

}

int kmp(str &s, const str &t1, const str &t2)

{

    int n = s.len, nextval[t1.len], i = 0, j = 0, cnt = 0;

    getNext(t1, nextval); //获取nextval数组

    while (i < s.len)

    {

        if (j == -1 || s.idx[i] == t1.idx[j])

            ++i, ++j;

        else

            j = nextval[j];

        if (j == t1.len)

        {

            j = 0, ++cnt;

            i -= t1.len;

            trimStr(s, i, t1.len); //删除t1

            insertStr(s, i, t2); //插入t2, 完成替换

            i += t2.len;

        }

    }

    return cnt;

}

int main()

{

    str s, t1, t2;

    initStr(s), initStr(t1), initStr(t2);

    inputStr(s), inputStr(t1), inputStr(t2);

    traverseStr(s);

    printf("本次kmp一共进行了%d次匹配\n", kmp(s, t1, t2));

    printf("匹配替换后的主串为:\n");

    traverseStr(s);

    printf("\n");

    return 0;

}