实验一 C 语言上机实验

实验目的：

1. 掌握 C 语言中二维数组的基本操作
2. 掌握函数的声明和调用
3. 熟练使用指针数组和数组指针

实验环境：

编译器为 Dev-C++，操作系统为 Windows 10

实验内容：

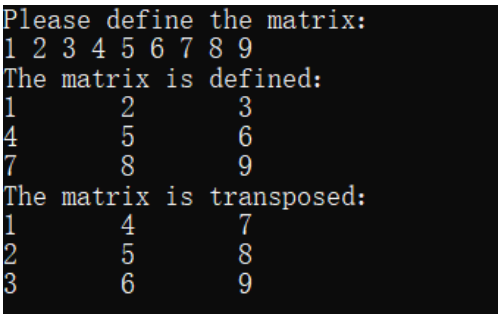
1. 编写一个函数 void convert(int array[3][3])，对给定的一个二维数组（3×3）转置，即行列互换。
2. 编写一个函数，由实参传来一个字符串，统计此字符串中字母、数字、空格和其他字符的个数，在主函数中输入字符串并输出结果。
3. 在主函数中输入 10 个等长的字符串，用另一个函数对它们排序。然后在主函数中输出这 10 个已排好序的字符串。（使用指向数组的指针来完成）
4. 用指针数组处理上一题，字符串不等长。

实验过程：

1. 在 Dev-C++上编译并运行 exer1.1.cpp

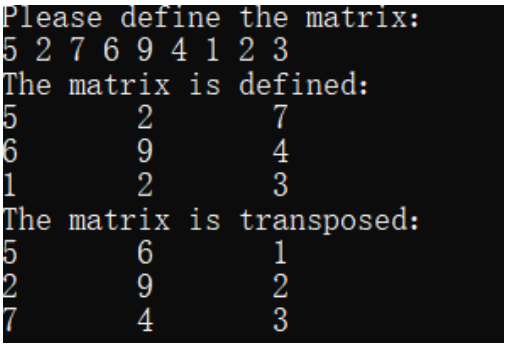
   测试用例 1：1 2 3 4 5 6 7 8 9

   运行结果：



   测试用例 2：5 2 7 6 9 4 1 2 3

   运行结果：



测试用例 3：1 5 9 6 4 2 7 3 8

运行结果：

```
Please define the matrix:
1 5 9 6 4 2 7 3 8
The matrix is defined:
1       5       9
6       4       2
7       3       8
The matrix is transposed:
1       6       7
5       4       3
9       2       8
```

2. 在 Dev-C++上编译并运行 exer2.1.cpp

测试用例 1：ABCDhij012139..　　// ?

运行结果：

```
letter:7, digit:6, space:4, others:5
```

测试用例 2：12hdjdahGDJS..　　adnj///,.";

运行结果：

```
letter:14, digit:2, space:3, others:10
```

测试用例 3：12hsd　...//　OHG

```
letter:6, digit:2, space:4, others:5
```

3. 在 Dev-C++上编译并运行 exer3.1.cpp

测试用例 1：aa

bb

cc

dd

ee

ff

gg

hh

ii

jj

运行结果：

```
Please input 10 strings:
aa
bb
cc
dd
ee
ff
gg
hh
ii
jj
The squence is:
aa
bb
cc
dd
ee
ff
gg
hh
ii
jj
```

测试用例 2：ABC
BCD
CDE
DEF
EFG
FGH
GHI
HIJ
IJK
JKL

运行结果：

```
Please input 10 strings:
ABC
BCD
CDE
DEF
EFG
FGH
GHI
HIJ
IJK
JKL
The squence is:
ABC
BCD
CDE
DEF
EFG
FGH
GHI
HIJ
IJK
JKL
```

4. 在 Dev-C++上编译并运行 exer4.1.cpp

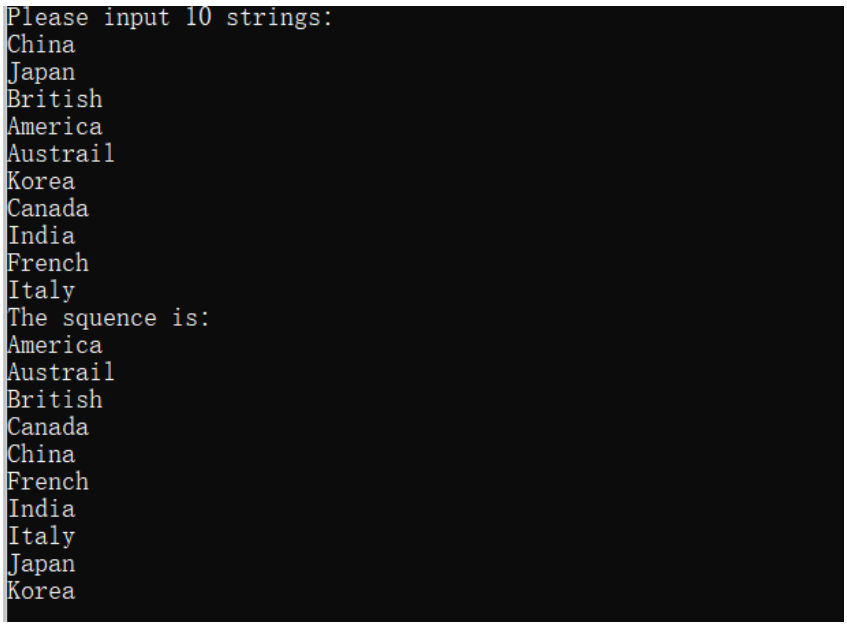
测试用例 1：abcdf
abdfg
adf
hsg
kshla
hgjs
mhsjs
hjl
hyi
okgf

运行结果：

```
Please input 10 strings:
abcdf
abdfg
adf
hsg
kshla
hgjs
mhsjs
hjl
hyi
okgf
The squence is:
abcdf
abdfg
adf
hgjs
hjl
hsg
hyi
kshla
mhsjs
okgf
```

测试用例 2：China
Japan
British
America
Austrail
Korea
Canada
India
French
Italy

运行结果：



实验总结

1. 指针数组与数组指针较易混淆
2. 实验过程中，涉及到输入多个字符串时需要用到空格隔开。

实验代码：

```
//exer01.cpp
#include <stdio.h>
#define N 3
int main(){
    void printArray(int a[N][N]);
    void conver(int a[N][N]);
    void scanfArray(int a[N][N]);

    int a[N][N];

    scanfArray(a);

    printf("The matrix is defined: \n");
    printArray(a);

    conver(a);

    printf("The matrix is transposed: \n");
    printArray(a);
    return 0;
}

void conver(int a[N][N]){
```

```
    for(int i = 0; i < N; i++){
        for(int j = 0; j < N; j++){
            int temp;
            a[i][i] = a[i][i];
            if(j > i){
                temp = a[i][j];
                a[i][j] = a[j][i];
                a[j][i] = temp;
            }
        }
    }

}

void printArray(int a[N][N]){
    for(int i = 0; i < N; i++){
        for(int j = 0; j < N; j++){
            printf("%d\t",a[i][j]);
        }
        printf("\n");
    }
}

void scanfArray(int a[N][N]){
    printf("Please define the matrix: \n");
    for(int i = 0; i < N; i++){
        for(int j = 0; j < N; j++){
            scanf("%d",&a[i][j]);
        }
    }
}
```

```
//exer02.cpp
#include <stdio.h>
#include <string.h>
int count(char a[]){
    int letterCount = 0,numberCount = 0,emptyCounter = 0,otherCount
= 0;
    int length = strlen(a);
    for(int i = 0; i < length; i++){
        if(a[i] >= 'A' && a[i] <= 'Z' || a[i] >= 'a' && a[i] <= 'z'){
            letterCount++;
        }
        else if(a[i] >= '0' && a[i] <= '9'){
            numberCount++;
```

```
        }
        else if(a[i] == ' '){
            emptyCounter++;
        }
        else{
            otherCount++;
        }
    }
    printf("letter:%d,digit:%d,space:%d,others:%d",letterCount,numb
erCount,emptyCounter,otherCount);
}

int main(){
    char str[100];
    printf("Please input string:\n");
    gets(str);
    printf("The string is inputed is:\n");
    puts(str);
    count(str);
    return 0;
}
```

```
 //exer03.cpp
#include <stdio.h>
#include <string.h>

#define N 10

int main(){

    void sortString(char (*p)[N]);
    void scanfString(char str[N][N]);
    void printString(char str[N][N]);

    char str[N][N];
    char (*p)[N];

    scanfString(str);
    p = str;

    sortString(p);

    printf("The squence is:\n");
    printString(p);
```

```
}

void sortString(char (*p)[N]){
    char temp[N];

    for(int i = 0; i < N - 1; i++){
        for(int j = 0; j < N - 1 - i; j++){
            if(strcmp(*(p + j), *(p + j +1)) > 0){
                strcpy(temp, *(p + j));
                strcpy(*(p + j), *(p + j + 1));
                strcpy(*(p + j + 1), temp);
            }
        }
    }
}

void scanfString(char str[N][N]){
    printf("Please input %d strings:\n",N);
    for(int i = 0; i < N; i++){
        scanf("%s",&str[i]);
    }
}

void printString(char str[N][N]){
    for(int i = 0; i < N; i++){
        printf("%s\n",str[i]);
    }
}
```

//exer04.cpp

```
#include <stdio.h>
#include <string.h>

#define N 10
#define M 15

int main(){
    void sortString(char *[]);
    void scanfString(char str[N][M]);
    void printString(char *p[N]);
    void cmpString(char *p[], char str[N][M]);

    char *p[N],str[N][M];

    scanfString(str);
```

```
    cmpString(p, str);
    sortString(p);

    printf("The squence is:\n");
    printString(p);

}

void sortString(char *p[]){
    char *temp;

    for(int i = 0; i < N - 1; i++){
        for(int j = 0; j < N - 1 - i; j++){
            if(strcmp(*(p + j), *(p + j + 1)) > 0){
                strcpy(temp, *(p + j));
                strcpy(*(p + j), *(p + j + 1));
                strcpy(*(p + j + 1), temp);
            }
        }
    }
}

void scanfString(char str[N][M]){
    printf("Please input %d strings:\n", N);
    for(int i = 0; i < N; i++){
        scanf("%s",&str[i]);
    }
}

void printString(char *p[N]){
    for(int i = 0; i < N; i++){
        printf("%s\n",p[i]);
    }
}

void cmpString(char *p[], char str[N][M]){
    for(int i = 0; i < N; i++){
        p[i] = str[i];
    }
}
```

实验二 数据结构上机实验

实验目的：

1. 运用顺序表来实现合并等基本操作
2. 利用栈结构的特性来实现进制的转换
3. 二叉树的先序次序构造
4. 递归的三种方式遍历二叉树，用非递归的中序次序遍历二叉树

实验环境：

编译器为 Dev-C++，操作系统为 Windows 10

实验内容：

1. 编写一个程序实现两个有序（从小到大）顺序表合并成为一个顺序表， 合并后的结果放在第一个顺序表中。

2. 利用栈结构具有先进后出的特性，编程实现：输入一个任意十进制数，转换为八进制数和二进制数进行输出。

3. 用先序次序的方法构造一棵二叉树：

1）三种递归的先序、中序、后续遍历方式遍历此二叉树。

2）以非递归的中序遍历方法遍历此二叉树。

实验过程：

1. 在 Dev-C++上编译并运行 2.1.cpp

   测试用例 1：

   12 23 45 46 48 51 0

   5 15 41 46 48 50 51 59 0

   运行结果：

```
请从小到大输入顺序表L1(以0结束)：12 23 45 46 48 51 0
请从小到大输入顺序表L2(以0结束)：5 15 41 46 48 50 51 59 0
合并处理后，顺序表L1为：5 12 15 23 41 45 46 46 48 48 50 51 51 59
--------------------------------
Process exited after 23.23 seconds with return value 0
请按任意键继续. . .
```

   测试用例 2：

   1 2 6 8 13 42 0

   6 7 13 19 41 42 50 0

   运行结果：

```
请从小到大输入顺序表L1(以0结束)：1 2 6 8 13 42 0
请从小到大输入顺序表L2(以0结束)：6 7 13 19 41 42 50 0
合并处理后，顺序表L1为：1 2 6 6 7 8 13 13 19 41 42 42 50
--------------------------------
Process exited after 25.65 seconds with return value 0
请按任意键继续. . .
```

2. 在 Dev-C++上编译并运行 2.2.cpp

测试用例 1:12

运行结果：

```
请输入要转换的十进制数：12
请输入要转换的数制：2
转换后的2进制数为：1100
--------------------------------
Process exited after 9.442 seconds with return value 0
请按任意键继续. . .
```

```
请输入要转换的十进制数：12
请输入要转换的数制：8
转换后的8进制数为：14
--------------------------------
Process exited after 2.585 seconds with return value 0
请按任意键继续. . .
```

测试用例 2：181

运行结果：

```
请输入要转换的十进制数：181
请输入要转换的数制：2
转换后的2进制数为：10110101
--------------------------------
Process exited after 4.074 seconds with return value 0
请按任意键继续. . .
```

```
请输入要转换的十进制数：181
请输入要转换的数制：8
转换后的8进制数为：265
--------------------------------
Process exited after 2.876 seconds with return value 0
请按任意键继续. . .
```

3. 在 Dev-C++上编译并运行 2.3.1.cpp

测试用例 1：AB..C.DE..F..

运行结果：

```
请按照先序方式依次输入结点的值(空结点为'.'):
AB..C.DE..F..
先序遍历结果为：ABCDEF
中序遍历结果为：BACEDF
后序遍历结果为：BEFDCA
--------------------------------
Process exited after 24.91 seconds with return value 0
请按任意键继续. . .
```

测试用例 2：ABC.D...EF..GH..I..

运行结果：

```
请按照先序方式依次输入结点的值(空结点为'.'):
ABC.D...EF..GH..I..
先序遍历结果为: ABCDEFGHI
中序遍历结果为: CDBAFEHGI
后序遍历结果为: DCBFHIGEA
--------------------------------
Process exited after 24.71 seconds with return value 0
```

4. 在 Dev-C++上编译并运行 2.3.2.cpp

   测试用例 1：ABC.D..E..FG..H..

   运行结果：

```
请按照先序方式依次输入结点的值(空结点为'.'):
ABC.D..E..FG..H..
非递归中序遍历结果为: CDBEAGFH
--------------------------------
Process exited after 34.19 seconds with return value 0
请按任意键继续. . .
```

   测试用例 2：ABC..D.E..F.G..H.I..

   运行结果：

```
请按照先序方式依次输入结点的值(空结点为'.'):
ABC..D.E..F.G..H.I..
非递归中序遍历结果为: CBDEAFG
--------------------------------
Process exited after 39.99 seconds with return value 0
请按任意键继续. . .
```

实验总结：

1. 创建顺序表时输入“0”表示结束创建。
2. 构建二叉树时空结点要使用‘.’，不然不能够形成一棵完整的二叉树。
3. 二叉树遍历的结果并非所有结点的值都会输出。

实验代码：

```
//2.1.cpp
/*****************************************
2.1 编写一个程序实现两个有序（从小到大）顺序
表合并成为一个顺序表， 合并后的结果放在第一
个顺序表中。
*****************************************/

#include <stdio.h>
#define M 100
typedef int datatype;

//制表
typedef struct{
    datatype a[M];
    int size;
}sequenceList;

//空表
```

```
void InitList(sequenceList *L){
    L->size = 0;
}


void Func(sequenceList *L1, sequenceList *L2){
    int i,j;
    for(i = 0; i < L2->size; i++ ){
        for(j = L1->size - 1; j >= 0; j--){
            if(L2->a[i] >= L1->a[j]){
                L1->a[j+1] = L2->a[i];
                break;
            }else{
                L1->a[j+1] = L1->a[j];
                if(j == 0)
                    L1->a[0] = L2->a[i];
            }
        }
        ++L1->size;
    }
}
void List(datatype a, sequenceList &L){
    scanf("%d",&a);
    while(a){
        L.a[L.size] = a;
        L.size++;
        scanf("%d",&a);
    }
}
int main(){
    sequenceList L1,L2;
    datatype a;
    int i;
    InitList(&L1);
    InitList(&L2);
    printf("请从小到大输入顺序表 L1(以 0 结束)：");
    List(a, L1);
//  scanf("%d",&a);
//  while(a){
//      L1.a[L1.size] = a;
//      L1.size++;
//      scanf("%d",&a);
//  }
    printf("请从小到大输入顺序表 L2(以 0 结束)：");
```

```
    List(a, L2);
//  scanf("%d",&a);
//  while(a){
//      L2.a[L2.size] = a;
//      L2.size++;
//      scanf("%d",&a);
//  }
    Func(&L1, &L2);
    printf("合并处理后，顺序表 L1 为：");
    for(i = 0; i < L1.size ; i++){
        printf("%d ",L1.a[i]);
    }
    return 0;
}
```

```
//2.2.cpp
/************************************************
2.2 利用栈结构具有先进后出的特性，编程实现：输入一个
任意十进制数，转换为八进制数和二进制数进行输出。
************************************************/

#include <stdio.h>
#include <stdlib.h>

#define INITSIZE 10
#define ERROR 0
#define OK 1
#define INCREMENT 2
typedef int Elemtype;
typedef int Status;
typedef struct{
    Elemtype *base;
    Elemtype *top;
    int StackSize;
}SqStack;

Status InitStack(SqStack *S){
    S->base = (Elemtype*)malloc(sizeof(Elemtype)*INITSIZE);
    if(!S->base){
        return ERROR;
    }
    S->top = S->base;
    S->StackSize = INITSIZE;
    return OK;
}
```

```
Status PushStack(SqStack *S, Elemtype e){
    if(S->top - S->base >= S->StackSize){
        S->base     =     (Elemtype*)realloc(S->base,(S->StackSize     +
INCREMENT)*sizeof(Elemtype));
        if(!S->base){
            return ERROR;
        }
        S->top = S->base + S->StackSize;
        S->StackSize += INCREMENT;
    }
    *S->top = e;
    S->top++;
    return OK;
}

Status StackEmpty(SqStack *S){
    if(S->top == S->base){
        return OK;
    }
    return ERROR;
}

Status PopStack(SqStack *S, Elemtype *e){
    if (S->top == S->base){
        return ERROR;
    }
    *e = *--S->top;
    return OK;
}

void Func_10_8_2(int number, int cet){
    SqStack S;
    Elemtype e;
    InitStack(&S);
    if(number == 0){
        printf("转换后的%d 进制数为：0", cet);
        return ;
    }
    printf("转换后的%d 进制数为：", cet);
    while(number){
    PushStack(&S, number%cet);
    number = number/cet;
    }
```

```
    while(!StackEmpty(&S)){
        PopStack(&S, &e);
        printf("%d", e);
    }
}
int main(){
    int number,cet;
    printf("请输入要转换的十进制数：");
    scanf("%d",&number);
    printf("请输入要转换的数制：");
    scanf("%d",&cet);
    Func_10_8_2(number,cet);
    return 0;
}
```

```
 //2.3.1.cpp
/*********************************************
3.用先序次序的方法构造一棵二叉树：
1）三种递归的先序、中序、后续遍历方式遍历此二叉树。
*********************************************/
#include <stdio.h>
#include <stdlib.h>
//树结点的数据类型
typedef char TElemType;
//创建二叉链表结点结构
typedef struct BiTNode{
    TElemType data;   //结点数据
    struct BiTNode *lchild, *rchild; //左右孩子指针
}BiTNode, *BiTree;
//结构体初始化
void Init(BiTree *T){
    *T = (BiTree)malloc(sizeof(BiTNode));
    (*T)->lchild = NULL;
    (*T)->rchild = NULL;
}
//建立二叉树
void CreateBiTree(BiTree *T){
    TElemType ch;
    scanf("%c",&ch);
    if(ch == '.'){
        *T = NULL;
    }else{
        *T= (BiTree)malloc(sizeof(BiTNode));
        (*T)->data = ch;
        CreateBiTree(&((*T)->lchild));
```

```
        CreateBiTree(&((*T)->rchild));
    }
}
//先序遍历（前序遍历）
void PreOrderTraverse(BiTree T){
    if(T == NULL){
        return;
    }
    printf("%c",T->data);
    PreOrderTraverse(T->lchild);
    PreOrderTraverse(T->rchild);

}

void InOrderTraverse(BiTree T){
    if(T == NULL){
        return;
    }
    InOrderTraverse(T->lchild);
    printf("%c",T->data);
    InOrderTraverse(T->rchild);
}

void PostOrderTraverse(BiTree T){
    if(T == NULL){
        return;
    }
    PostOrderTraverse(T->lchild);
    PostOrderTraverse(T->rchild);
    printf("%c",T->data);
}
int main()
{
    BiTree T;
    Init(&T);
    printf("请按照先序方式依次输入结点的值(空结点为'.'):\n");
    CreateBiTree(&T);
    printf("先序遍历结果为：");
    PreOrderTraverse(T);
    printf("\n 中序遍历结果为：");
    InOrderTraverse(T);
    printf("\n 后序遍历结果为：");
    PostOrderTraverse(T);
    return 0;
```

```
}
```

```
//2.3.2.cpp
/*********************************************
3.用先序次序的方法构造一棵二叉树：
2）以非递归的中序遍历方法遍历此二叉树。
*********************************************/
#include <stdio.h>
#include <stdlib.h>
//树结点的数据类型
typedef char TElemType;
//创建二叉链表结点结构
typedef struct BiTNode{
    TElemType data;    //结点数据
    struct BiTNode *lchild, *rchild; //左右孩子指针
}BiTNode, *BiTree;

//定义栈
typedef struct{
    BiTNode *stack;
    int base;
    int top;
    int stacksize;
}Stack;
//初始化栈
void InitStack(Stack* &s){
    s=(Stack*)malloc(sizeof(Stack));
    s->top = s->base = 0;
    s->stacksize = 20;
}
int StackEmpty(Stack* &s){
    if(s->top == s->base){
        return 1;
    }else{
        return 0;
    }
}
int StackFull(Stack* &s){
    if(s->top - s->base == s->stacksize){
        return 1;
    }else{
        return 0;
    }
}
```

```
void StackPush(Stack* &s, BiTNode* &T){
    if(StackFull(s) == 1){
        return ;
    }
    s->stack[s->top].data = T->data;
    s->stack[s->top].lchild = T->lchild;
    s->stack[s->top].rchild = T->rchild;
    s->top++;
}

BiTNode* StackPop(Stack* &s){
    if(StackEmpty(s) == 1){
        return NULL;
    }
    s->top--;
    return &(s->stack[s->top]);
}
void CreateBiTree(BiTNode* &T){
    char ch;
    scanf("%c",&ch);
    if(ch != '.'){
        T = (BiTNode*)malloc(sizeof(BiTNode));
        T->data = ch;
        CreateBiTree(T->lchild);
        CreateBiTree(T->rchild);
    }else{
        T = NULL;
    }
}
void InOrderTraverse(Stack* &s, BiTNode* &T){
    InitStack(s);
    BiTNode* p = T;
    BiTNode* q;
    while(p || !StackEmpty(s)){
        if(p){
            StackPush(s, p);
            p = p->lchild;
        }else{
            q = StackPop(s);
            putchar(q->data);
            p = q->rchild;
        }
    }
}
```

```
int main(){
    Stack* s;
    BiTNode* T;
    printf("请按照先序方式依次输入结点的值(空结点为'.'):\n");
    CreateBiTree(T);
    printf("非递归中序遍历结果为：");
    InOrderTraverse(s, T);
    return 0;

}
```