汇编实验四

完成实验后，需用实验报告纸撰写实验报告。

一、实验报告包含以下内容

- 实验序号
- 实验内容
- 算法描述
- 汇编程序
- 运行结果

二、实验目的

1. 掌握分支与循环结构的汇编表示。
2. 掌握子以栈传递参数方式编写程序或函数
3. 掌握递归编程
4. 熟悉调试器 ollydbg 的使用。

三、实验内容

1.编程输出九九乘法表

2.编程实现解决整数拆分问题

3.编程实现解迷宫问题

4.编程实现解素数环问题

提示：

2.问题描述

输入一个 N，输出所有拆分的方式。

如 input: 3

output:

1+1+1

1+2

3

算法思想:

用一个数组 res[]存放拆分的解，用全局变量存放拆分的方法数。

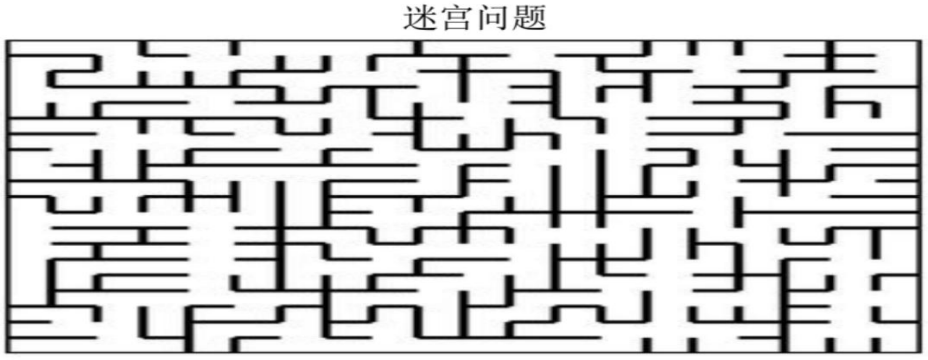
divN（n，k）使用 n 表示要分解的整数，k 表示 res 数组下标，即第 k 次拆分。先从 divN（n，1）开始，用 num 表示第 k 个拆分的数，即 res[k]=num，让 num 在[1，n]内遍历。用 rest=n-num 表示拆分后剩下的整数值。若 rest 等于零，代表本次拆分结束，输出拆分解。否则处理第 k+1 个数组元素，即 divN（rest，k+1），依次类推，直到 rest 为 0 输出结果。

源代码

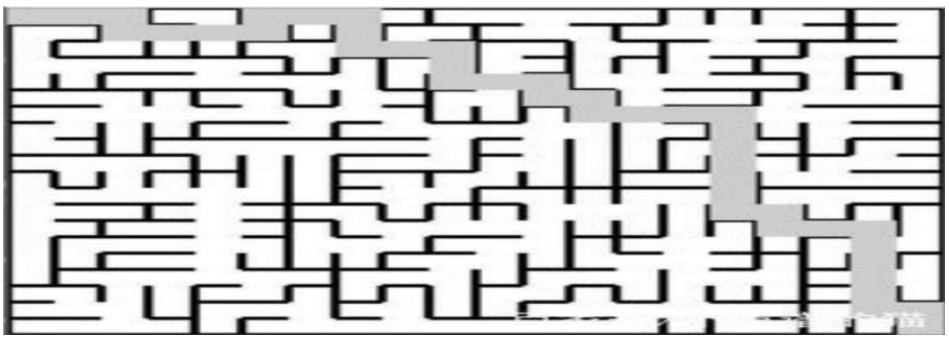
```
#include "stdafx.h"
#include<stdio.h>
#include<stdlib.h>
int res[10000] = { 0 }; //res 数组存放解
int times = 0; //times 计算拆分的次数
void divN(int n, int k) { //n 是需要拆分的整数，k 是指 res 数组的下标
	int rest;		//存放拆分后剩余的整数
	for (int num = 1;num <= n; num++) {	//从 1 开始尝试拆分
```

```
            if (num >= res[k - 1] ) { //拆分的解要大于或等于前一个解保证不重复
                res[k] = num;          //将这次拆分存放在 res 数组中
                rest = n - num;        //剩下的是 n-num
                if (rest == 0) {       //如果没有剩下的，说明本次拆分结束
                    times++;           //拆分次数加 1
                    printf("%3d：", times);
                    for (int j = 1; j < k; j++) {    //输出解
                        printf("%d+", res[j]);
                    }
                    printf("%d\n", res[k]);
                }
                else divN(rest, k + 1);   //如果有剩下的，继续求出 res[k+1]
            }
        }
}
int main() {
    int n;
    printf("Please enter a integer N:");
    scanf_s("%d", &n);
    divN(n, 1);
    printf("there are %d ways to divide the integer %d.", times,n);
    system("pause");
    return 0;
}
```

3. 解迷宫问题



解迷宫问题的一个解如下：



有一个 7 x 7 的迷宫，起点是'S'，终点是'E'，墙是'o'，道路是空格。
请找出从起点到终点的通路，通路用符号'.'表示。

用二维数组表示迷宫场景。其中用 2 代表迷宫的墙壁，0 代表可行通道。
走的路径记作 1，也就是数组中的 0 被改为 1

```
#include <stdio.h>
#include <stdlib.h>
#define M 9
//把 7*7 迷宫加大成 9*9 格局
int maze[M][M] ={
        {2,2,2,2,2,2,2,2,2},
        {2,0,0,0,0,0,0,0,2},
        {2,0,2,2,0,2,2,0,2},
        {2,0,2,0,0,2,0,0,2},
        {2,0,2,0,2,0,2,0,2},
        {2,0,0,0,0,0,2,0,2},
        {2,2,0,2,2,0,2,2,2},
        {2,0,0,0,0,0,0,0,2},
        {2,2,2,2,2,2,2,2,2}
};

int start1=1,start2=1;              //假定[1][1]是入口
int end1=7,   end2=7;                  //假定[7][7]是出口

void visit(int i,int j){
        int m,n;
    maze[i][j] = 1;
        if(i==end1 && j==end2) { //判断是否到达出口位置，到达直接输出
                printf("\n 显示路径：\n");
                for(m=0;m<M;m++){
                    for(n=0;n<M;n++){
                        if(maze[m][n] == 2)    printf("o");
                        else if(maze[m][n] == 1)    printf(".");
                        else         printf(" ");
                      }
                      printf("\n");
                 }//end for
      }//end if
    //不再判定是否到达出口，只分析老鼠可以在迷宫移动的方向，
    //并递归求下一步.
    if(maze[i][j+1] == 0)       visit(i,j+1);
    if(maze[i+1][j] == 0)       visit(i+1,j);
    if(maze[i][j-1] == 0)       visit(i,j-1);
```

```
    if(maze[i-1][j] == 0)       visit(i-1,j);
    //若代码运行到这一步，则证明前面走的路径并不能到达出口，
    //则返回，把走过的位置重新写作 0
     maze[i][j] = 0;
}

int main (){
        int i,j;
        printf("显示迷宫：\n");
        for(i=0;i<M;i++)    {   //对摆放的数组迷宫进行打印
            for(j=0;j<M;j++)
                if(maze[i][j] == 2)     printf("o");
                else        printf(" ");
                printf("\n");
        }
        visit(start1,start2);   //直接调用 visit 函数，把输出内容放在 visit 函数中，
        //好让所有路径进行遍历
        return 0;
}
```

4.素数环

题目：输入正整数 n，把整数 1，2，3，...，n 组成一个环。

使得相邻两个整数之和均为素数。

输出时从整数 1 開始逆时针排列。

同一个环应该恰好输出一次。

```
#include<iostream>
#include<cstdio>
#include<cstring>
using namespace std;

const int maxn =1000;
int vis[maxn];
int A[maxn];
int isp[maxn];
int n;
int ans=0;

int is_prime(int x){
    for( int i=2; i*i<=x; i++ ){
        if(x%i==0) return 0;
    }
    return 1;
}

void dfs(int cur){
```

```
    if(cur==n&&isp[A[0]+A[n-1]]){
        ans++;
        for( int i=0; i<n; i++ ) cout<<A[i]<<" ";
        cout<<endl;
    }
    else{
        for(int i=2; i<=n; i++ ){
            if(!vis[i]&&isp[i+A[cur-1]]){
            /*i 这个数没被用过，并且符合前后两个数相加为素数的要求*/
                A[cur]=i;/*采用这个数*/
                vis[i]=1;/*设置使用标志*/
                dfs(cur+1);
                vis[i]=0;/*消除标志*//*回溯的本质*/
            }
        }
    }
}

int main(int argc, char const *argv[])
{
    cin>>n;
    memset(vis,0,sizeof(vis));
    for( int i=2; i<=n*2; i++ ) isp[i]=is_prime(i);
    A[0]=1;/*题目中规定从 1 开始*/
    dfs(1);
    cout<<ans<<endl;

    return 0;
}
```