Assignment 3

Name: 刘志远

SID: 11612532

因为要交代码所以报告不把所有代码贴上去了。贴关键代码然后阐述思路~

E1 Code & Result

- **Code**

整体思路：维护一个二维数组，记录每个点的可访问状况，初始状态都为0，若该点有子弹经过则变为1并记录子弹所覆盖点的数量，最后用总的点数减去子弹覆盖的点则为答案。

Data structure:我使用了一个结构体存放每个bullets的信息。

```
4  struct bullets{
5     int pos_x;
6     int pos_y;
7     int dir_x;
8     int dir_y;
9  };
```

N m k输出和初始化

```
29    int n,m,k,fir,sec,dir1,dir2,ans=0;
30    cout <<"plz enter n,m,k";
31    cin>>n>>m>>k;
32    //malloc
33    int **pointer=(int **)malloc(sizeof(int *)*n);
34    for (int  i = 0; i < n; i++)pointer[i]=(int *)malloc(sizeof(int)*m);
35
36    bullets *bullets_arr=(bullets*)malloc(sizeof(bullets)*k);
37    //init
38    for (int i = 0; i < n; i++)
39    {
40       for (int j = 0; j <m; j++)pointer[i][j]=0;
41    }
42
```

对每个bullets进行update函数

```
59    for (int i = 0; i < k; i++)
60    ans+=update(pointer,bullets_arr[i].pos_x,
61    bullets_arr[i].pos_y,bullets_arr[i].dir_x,bullets_arr[i].dir_y,n,m);
```

Update

从起点沿着向量方向遍历路径上的节点知道边界。在途中更新每个节点的值。

```
int update(int ** pointer,int pos_x,int pos_y,int dir_x,int dir_y,int n,int m){
    int cnt=0;
    int temp_x=pos_x;
    int temp_y=pos_y;
    while(temp_x>=0 && temp_x<n && temp_y>=0 &&temp_y<m){
        if(pointer[temp_x][temp_y]==0){
            cnt++;
            pointer[temp_x][temp_y]=1;
        }
        temp_x+=dir_x;
        temp_y+=dir_y;

    }
    return cnt;

 }
```

- **Result**

```
3
guludeMacBook-Pro:assig3 gulu$ g++ exer1.cpp -o exer1
guludeMacBook-Pro:assig3 gulu$ ./exer1
plz enter n,m,k3 4 5
plz enter next part~
1 1 1 -1
1 1 -1 1
0 3 1 0
0 2 1 0
0 0 -1 -1
3
```

E2 Code & Result

该题目只用从右上角开始逆时针遍历一遍数组即可。

- **Code**

Main

1.Main 函数首先完成对m，n的输入以及验证合法性，我的策略是不合法就继续输入直至合法为止。

2.数组的malloc以及初始化。

3.spiral函数将数组按要求赋值。

4.printf_arr打印数组

5.free

```
int main(){
    int m,n;
    do{
    cout <<"plz enter two integer m,n:";
    cin >> m;
    cin >> n;
    if(m>0 && n>0){
        int **arr=(int **)malloc(sizeof(int *)*m);
        for (int i = 0; i < m; i++)
        {
            arr[i]=(int *)malloc(sizeof(int)*n);
        }
        for (int i = 0; i < m; i++)
        {
            for (int j = 0; j < n; j++)
            {
                arr[i][j]=0;
            }
        }
        spiral(m,n,arr);
        printf_arr(arr,m,n);
        for (int i = 0; i < m; i++)
        {
            free(arr[i]);
        }
        free(arr);
    }
    }while(m<=0 || n<=0);
}
```

维护四个变量分别代表上下左右四个边界。每次移动检测到边界则退出循环并更新相应的边界范围。
由于该算法在最后一个的时候会被边界卡掉。所以最后一个单独赋值。

```
14  void spiral(int m,int n,int **arr){
15      int top=0;
16      int bottom=m-1;
17      int left=0;
18      int right=n-1;
19      int pos_x=0;
20      int pos_y=n-1;
21      int cnt=0;
22      while(cnt<m*n-1){
23          while(pos_y>left && arr[pos_x][pos_y]==0){
24              arr[pos_x][pos_y--]=++cnt;
25          }
26          top++;
27          while(pos_x<bottom && arr[pos_x][pos_y]==0){
28              arr[pos_x++][pos_y]=++cnt;
29          }
30          left++;
31          while(pos_y<right && arr[pos_x][pos_y]==0){
32              arr[pos_x][pos_y++]=++cnt;
33          }
34          bottom--;
35          while(pos_x>top && arr[pos_x][pos_y]==0){
36              arr[pos_x--][pos_y]=++cnt;
37          }
38          right--;
39      }
40      if(!arr[pos_x][pos_y])arr[pos_x][pos_y]=++cnt;
41  }
```

- **Result**

```
guludeMacBook-Pro:assig3 gulu$ g++ exer2.cpp -o exer2
guludeMacBook-Pro:assig3 gulu$ ./exer2
plz enter two integer m,n:4 4
4   3   2   1
5   14   13   12
6   15   16   11
7   8   9   10
guludeMacBook-Pro:assig3 gulu$ ./exer2
plz enter two integer m,n:5 6
6   5   4   3   2   1
7   22   21   20   19   18
8   23   30   29   28   17
9   24   25   26   27   16
10   11   12   13   14   15
guludeMacBook-Pro:assig3 gulu$ ./exer2
plz enter two integer m,n:10 10
10   9   8   7   6   5   4   3   2   1
11   44   43   42   41   40   39   38   37   36
12   45   70   69   68   67   66   65   64   35
13   46   71   88   87   86   85   84   63   34
14   47   72   89   98   97   96   83   62   33
15   48   73   90   99   100   95   82   61   32
16   49   74   91   92   93   94   81   60   31
17   50   75   76   77   78   79   80   59   30
18   51   52   53   54   55   56   57   58   29
19   20   21   22   23   24   25   26   27   28
guludeMacBook-Pro:assig3 gulu$ ./exer2
plz enter two integer m,n:-1 0
plz enter two integer m,n:
```

E3 Code & Result

首先将Block.txt读入内存，然后重定向文本到该进程，对文本进行utf-8 to unicode的解码，然后将unicode值进行比对统计处出现此时最多的unicode区间即block

- **Code**

  数据结构，使用map储存每个block出现次数，key为block的name，value为出现的次数 times

```
 8
 9  struct block{
10      string block_name;
11      int sta;
12      int end;
13  } block_list[300];
14  int cnt=0;
15
16  map<string,int> cnt_map;//<name,times>
17  int max_val=0;
18  string max_string="";
19
20
```

标准输入每次读入一行,将开头为’ #’ 和 ‘\0 ‘的过滤掉然后剩余的调用store函数，store函数可以把每一行的信息提炼并且储存在map中。

```
96  void store(string &s){
97      int sta=0;
98      int end=0;
99      string name="";
100     int pos=0;
101     while(s[pos]!='.'){
102         name+=s[pos++];
103     }
104     block_list[cnt].sta=hex_string_to_int(name);
105     pos+=2;
106     name="";
107     while(s[pos]!=';'){
108         name+=s[pos++];
109     }
110     block_list[cnt].end=hex_string_to_int(name);
111     pos+=2;
112     name="";
113
114     for (int i = pos; i < s.length(); i++)
115     {
116         name+=s[i];
117     }
118     block_list[cnt++].block_name=name;
119 }
```

重定向文本处理。每次读入一个char，对该char进行分析，使用utf-8的准则进行译码，分析出该character的长度len，一次读入len的char并将char数组和len传入utf8_2_unicode函数进行解码，返回值为unicode 的int 值。

```
133     while(cin>>temp_c){
134
135         if(temp_c<=127)loop_cnt=1;//0xxx xxxx
136         else if(temp_c<=223)loop_cnt=2;//110x xxxx
137         else if(temp_c<=239)loop_cnt=3;//1110 xxxx
138         else loop_cnt=4;//1111 0xxx
139         arr[0]=temp_c;
140         for (int i = 1; i < loop_cnt; i++)cin >>arr[i];
141
142         int val=utf8_2_unicode(arr,loop_cnt);
143         // cout << val << endl;
144         check_range(val);
145     }
```

根据长度进行相应的计算，格式为utf-8的标准格式。

```
int utf8_2_unicode(unsigned char *c,int len){
    int ans=0;
    if(len==1)//01xx xxxx
    {
        ans=int(c[0]);
    }
    else if (len==2)//110x xxxx  |  10xx xxxx
    {
        ans+=int((c[0]-128-64));
        ans<<=6;
        ans+=int(c[1]-128);
    }
    else if(len==3)//1110 xxxx | 10xx xxxx | 10xx xxxx
    {
        ans+=int(c[0]-128-64-32);
        ans<<=6;
        ans+=int(c[1]-128);
        ans<<=6;
        ans+=int(c[2]-128);
    }
    else //1111 0xxx | 10xx xxxx | 10xx xxxx| 10xx xxxx
    {
        ans+=int(c[0]-128-64-32-16);
        ans<<=6;
        ans+=int(c[1]-128);
        ans<<=6;
        ans+=int(c[2]-128);
        ans<<=6;
        ans+=int(c[3]-128);
    }
    return ans;
}
```

将上面函数的返回值传入check_range，统计出出现最多的block，并返回block_name。

```
void check_range(int c){
    bool bug=true;
    for (int i = 0; i < cnt; i++)
    {
        if(c>= block_list[i].sta && c<=block_list[i].end){//hit
            if(cnt_map.count(block_list[i].block_name)==0){
                cnt_map[block_list[i].block_name]=1;
            }
            else{
                cnt_map[block_list[i].block_name]+=1;
            }
            if(cnt_map[block_list[i].block_name]>max_val){
                max_val=cnt_map[block_list[i].block_name];
                max_string=block_list[i].block_name;
            }
            bug=false;
        }
    }
    if(bug){
        cout <<"!!!!!!!!!!!!!!!!"<<endl;
    }
}
```

- **Result**

```
For more details, please visit https://support.apple.com/kb/HT208050.
guludeMacBook-Pro:Assignment3_for_sutdents 2 gulu$ ./gulu < ./TEST\ DATA\ FOR\ LAB\ 4/sample.txt
Armenian
3230
guludeMacBook-Pro:Assignment3_for_sutdents 2 gulu$ ./gulu < ./TEST\ DATA\ FOR\ LAB\ 4/sample2.txt
Georgian
1127
guludeMacBook-Pro:Assignment3_for_sutdents 2 gulu$ ./gulu < ./TEST\ DATA\ FOR\ LAB\ 4/sample3.txt
Lao
454
guludeMacBook-Pro:Assignment3_for_sutdents 2 gulu$
```