实验七、八 数组实验

计算机学院 熊明 20305055

一、实验目的

1. 理解一维数组和二维数组的概念，理解字符数组和字符串的概念。
2. 掌握一维数组和二维数组的定义、初始、数组元素引用。
3. 掌握字符数组的处理，掌握常用字符串处理函数。
4. 掌握数组和循环配合编程的方法。

二、实验原理

数组（Array）是C++的一种数据结构，是相同类型的一些对象的一个集合，这些对象依次存储在连续的内存空间中。每个独立的对象被称为数组的元素（element），所包含的数据的个数称为数组长度（Length）。用数组来表示相同类型的数据，编写计算和操作这些数据变得非常容易。我们通过数组的名称和数组中特定元素的位置号来引用数组中的特定位置的元素。例如一个名为c的整数数组，它包含12个元素。通过数组名c和在方括号（[ ]）中给出数组位置来引用数组中任意一个元素。位置号更正式地称为索引或下标。第一个元素有下标0，有时称为第零个元素。因此，数组c的元素是c[0]（发音为“c sub zero”）、c[1]、c[2]等等。数组c中的最高下标是11，比数组中的元素数少1。数组名遵循与其他变量名相同的约定。

数组的定义决定了数组名称、元素类型以及元素个数。没有显式初始化操作的数组定义。

在数组定义中，可以将元素数量指定为一个常量表达式，或者在特定情况下，指定为涉及变量的表达式。采用这两种方式定义的数组分别被称为固定长度数组（fixed-length）和长度可变（variable length）数组。

当数组中每个元素都只带有一个下标时，称这样的数组为一维数组。一维数组用于表示值的列表或序列中的项。数组中的每个元素都有一个序号，这个序号从0开始，而不是从我们熟悉的1开始，称为下标（Index），也称为索引变量或下标变量。使用数组元素时，指明下标即可。

在C++中，一维数组的引用格式为：数组名[下标]

例如：

```
int a[10];
```

其中，a是一维数组的数组名，该数组有10个元素，依次表示为a[0] a[1] a[2] a[3] a[4] a[5] a[6] a[7] a[8] a[9]。

数组的初始化：上面的代码是先定义数组再给数组赋值，我们也可以在定义数组的同时赋值，例如：

```
int a[4] = {20, 345, 66, 39};
```

数组元素的值由{ }括起来，各个值之间以,分隔。

对于数组程序设计需要注意以下几点：

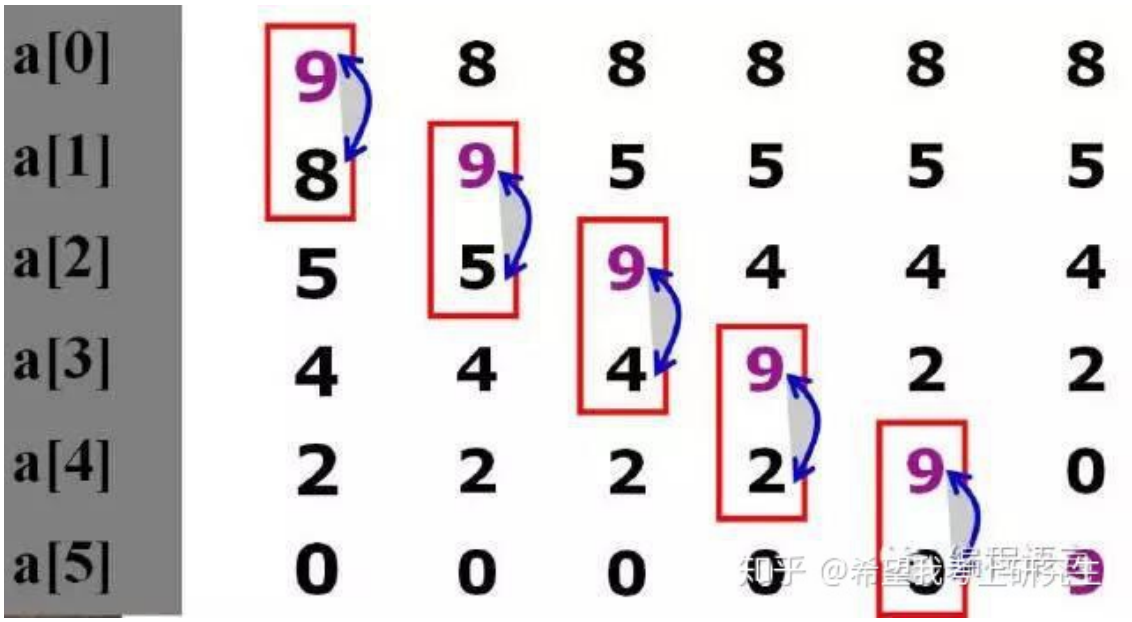
1. 可以只给部分元素赋值。
2. 只能给元素逐个赋值，不能给数组整体赋值（除了定义的时候）
3. 使用数组最常见的错误就是引用不存在的数组下标，引起数组索引越界。

三、实验内容

1. 用C++编写冒泡排序改进程序：当一次冒泡过程中发现没有交换操作时，表明序列已经排好序了，便终止冒泡操作。将10个整数按照从小到大的顺序排序，分析算法复杂度。排序功能分别用非函数和函数实现。

   冒泡排序：

   1. 其中i=0时： j从0开始a[0],a[1]比较大小，把其中的较大者给a[1],然后j++,a[1]和a[2]再比较，再把两者中的较大者给a[2],这样a[0],a[1],a[2]中的最大者已经交换到a[2]中，这个过程继续，直到j=10-i-1=9,这样 a[9]中的为10个数中的最大数。
   2. 然后i=1时: 由于最大数已找到并放到a[9]中，所以这一次循环j最大只需到10-i-1=8，即a[8]即可，再次从j=0开始a[j]和a[j+1]两两比较交换，最后次大数放到a[8]中
   3. 然后i++,继续...
   4. 当i=9时已经过9次两两比较完成所有排序，i<9不再成立退出比较。



对于n个数，只需要进行n-1次外循环的两两比较就完成排序。

至于按降序排列只需将if(a[j]>a[j+1])改为if(a[j]<a[j+1])即可。

```
#include<iostream>
using namespace std;
void print_arr(int *a,int n){
    for(int i=0;i<n;i++){
        cout<<a[i]<<" ";
    }
    cout<<endl;
}
void swap(int &a,int &b){
    int temp = a;
    a = b;
    b = temp;
```

```
}
void sorted(int *a,int n){//从小到大
    for (int i = 0; i < n-1;i++){
        for (int j = 0; j < n - i-1;j++){
            if (a[j] > a[j + 1]){
                swap(a[j],a[j+1]);
            }
        }
    }
    print_arr(a,n);
}
int main(){
    int a[100];
    int n;
    cout<<"输入数组规模："<<endl;
    cin>>n;
    for(int i=0;i<n;i++){
        cout<<"输入第"<<i+1<<"个数字："<<endl;
        cin>>a[i];
    }
    sorted(a,n);
    for (int i = 0; i < n-1;i++){    //从大到小
        for (int j = 0; j < n - i-1;j++){
            if (a[j] < a[j + 1]){
                swap(a[j],a[j+1]);
            }
        }
    }
    print_arr(a,n);
}
```

sorted函数是将数组a从小到大排序，而main函数里面的是将a从大到小排序。由此可以完成函数和非函数的实现。

运行结果如下：

```
输入数组规模:
5
输入第1个数字:
4
输入第2个数字:
3
输入第3个数字:
2
输入第4个数字:
1
输入第5个数字:
6
1 2 3 4 6
6 4 3 2 1
```

2. 输入一个字符串（长度小于100），分别统计大写字母A~Z、小写字母a~z、数字0~9的出现次数和频率。利用数组的理论知识，分别用非函数和函数版本实现

```cpp
#include<string>
#include<iostream>
using namespace std;
int main(){
    string str;
    cout<<"输入字符串："<<endl;
    cin>>str;
    int count[70] = {0};
    for(int i=0;i<str.size();i++){
        if(str[i]>=48 && str[i]<=57)//0~9 0~9
            count[int(str[i])-48]++;
        else if(str[i]>=65 && str[i]<=90)//A~Z 10~35
            count[int(str[i])-55]++;
        else if(str[i]>=97 && str[i]<=122)//a~z 36~61
            count[int(str[i])-61]++;
    }
    for(int i=0;i<70;i++){
        if(i>=0 && i<=9){
            if(count[i]!=0)
                cout<<i<<"出现的次数："<<count[i]<<endl;
        }else if(i>=10 && i<=35){
            if(count[i]!=0)
                cout<<char(i+55)<<"出现的次数："<<count[i]<<endl;
        }else if(i>=36 && i<=61){
            if(count[i]!=0)
                cout<<char(i+61)<<"出现的次数："<<count[i]<<endl;
        }
    }
}
```

运行结果：

```
输入字符串：
sU2xUAlA8LSzv1z2y8eA24E3j
1出现的次数：1
2出现的次数：3
3出现的次数：1
4出现的次数：1
8出现的次数：2
A出现的次数：3
E出现的次数：1
L出现的次数：1
S出现的次数：1
U出现的次数：2
e出现的次数：1
j出现的次数：1
l出现的次数：1
s出现的次数：1
v出现的次数：1
x出现的次数：1
y出现的次数：1
z出现的次数：2
```

3. 贪吃蛇游戏

源代码：

```
#include <graphics.h>
#include <conio.h>
#include <stdio.h>
#include <time.h>
#define BLOCK_SIZE 20 // 每个小格子的长宽大小
#define HEIGHT 30 // 高度上一共30个小格子
#define WIDTH 40 //  宽度上一共40个小格子
// 全局变量定义
int Blocks[HEIGHT][WIDTH] = { 0 }; //  二维数组，用于记录所有的游戏数据
char moveDirection;  //  小蛇移动方向
int food_i, food_j; //  食物的位置
int isFailure = 0; //  是否游戏失败

void moveSnake() //  移动小蛇及相关处理函数
{
    int i, j;
    for (i = 0; i < HEIGHT; i++) // 对行遍历
        for (j = 0; j < WIDTH; j++) // 对列遍历
            if (Blocks[i][j] > 0) // 大于0的为小蛇元素
                Blocks[i][j]++; //  让其+1
    int oldTail_i, oldTail_j, oldHead_i, oldHead_j; // 定义变量，存储旧蛇尾、旧蛇头坐标
    int max = 0; // 用于记录最大值
    for (i = 0; i < HEIGHT; i++) //  对行列遍历
    {
        for (j = 0; j < WIDTH; j++)
        {
            if (max < Blocks[i][j]) //  如果当前元素值比max大
            {
                max = Blocks[i][j]; // 更新max的值
                oldTail_i = i; //  记录最大值的坐标，就是旧蛇尾的位置
                oldTail_j = j; //
            }
            if (Blocks[i][j] == 2) // 找到数值为2
            {
                oldHead_i = i; //  数值为2恰好是旧蛇头的位置
                oldHead_j = j; //
            }
        }
    }
    int newHead_i = oldHead_i; //  设定变量存储新蛇头的位置
    int newHead_j = oldHead_j;

    //  根据用户按键，设定新蛇头的位置
    if (moveDirection == 'w') // 向上移动
        newHead_i = oldHead_i - 1;
    else if (moveDirection == 's') // 向下移动
        newHead_i = oldHead_i + 1;
    else if (moveDirection == 'a') // 向左移动
        newHead_j = oldHead_j - 1;
    else if (moveDirection == 'd') // 向右移动
        newHead_j = oldHead_j + 1;
```

```
52
53      //  如果蛇头超出边界，或者蛇头碰到蛇身，游戏失败
54      if (newHead_i >= HEIGHT || newHead_i < 0 || newHead_j >= WIDTH ||
    newHead_j < 0
55          || Blocks[newHead_i][newHead_j]>0)
56      {
57          isFailure = 1; //  游戏失败
58          return; // 函数返回
59      }
60
61      Blocks[newHead_i][newHead_j] = 1;  // 新蛇头位置数值为1
62      if (newHead_i == food_i && newHead_j == food_j) //  如果新蛇头正好碰
    到食物
63      {
64          food_i = rand() % (HEIGHT - 5) + 2; //  食物重新随机位置
65          food_j = rand() % (WIDTH - 5) + 2; //
66          // 不对旧蛇尾处理，相当于蛇的长度+1
67      }
68      else // 新蛇头没有碰到食物
69          Blocks[oldTail_i][oldTail_j] = 0; // 旧蛇尾变成空白，不吃食物时保持
    蛇的长度不变
70  }
71
72  void startup()  //  初始化函数
73  {
74      int i;
75      Blocks[HEIGHT / 2][WIDTH / 2] = 1; // 画面中间画蛇头，数字为1
76      for (i = 1; i <= 4; i++) //  向左依次4个蛇身，数值依次为2,3,4,5
77          Blocks[HEIGHT / 2][WIDTH / 2 - i] = i + 1;
78      moveDirection = 'd';     //  初始向右移动
79      food_i = rand() % (HEIGHT - 5) + 2; //  初始化随机食物位置
80      food_j = rand() % (WIDTH - 5) + 2; //
81      initgraph(WIDTH * BLOCK_SIZE, HEIGHT * BLOCK_SIZE); //  新开画面
82      setlinecolor(RGB(200, 200, 200)); // 设置线条颜色
83      BeginBatchDraw(); // 开始批量绘制
84  }
85
86  void show()  // 绘制函数
87  {
88      cleardevice(); // 清屏
89      int i, j;
90      for (i = 0; i < HEIGHT; i++) //  对二维数组所有元素遍历
91      {
92          for (j = 0; j < WIDTH; j++)
93          {
94              if (Blocks[i][j] > 0) // 元素大于0表示是蛇，这里让蛇的身体颜色色
    调渐变
95                  setfillcolor(HSVtoRGB(Blocks[i][j] * 10, 0.9, 1));
96              else
97                  setfillcolor(RGB(150, 150, 150)); // 元素为0表示为空，颜色
    为灰色
98              // 在对应位置处，以对应颜色绘制小方格
99              fillrectangle(j * BLOCK_SIZE, i * BLOCK_SIZE,
100                 (j + 1) * BLOCK_SIZE, (i + 1) * BLOCK_SIZE);
101         }
```

```
    }
    setfillcolor(RGB(0, 255, 0)); //  食物为绿色
    //  绘制食物小方块
    fillrectangle(food_j * BLOCK_SIZE, food_i * BLOCK_SIZE,
        (food_j + 1) * BLOCK_SIZE, (food_i + 1) * BLOCK_SIZE);
    if (isFailure) //  如果游戏失败
    {
        setbkmode(TRANSPARENT); // 文字字体透明
        settextcolor(RGB(255, 0, 0));// 设定文字颜色
        settextstyle(80, 0, _T("宋体")); //  设定文字大小、样式
        outtextxy(240, 220, _T("游戏失败")); //  输出文字内容
    }
    FlushBatchDraw(); // 批量绘制
}

void updateWithoutInput() // 与输入无关的更新函数
{
    if (isFailure) //  如果游戏失败，函数返回
        return;
    static int waitIndex = 1; // 静态局部变量，初始化时为1
    waitIndex++; // 每一帧+1
    if (waitIndex == 10) // 如果等于10才执行，这样小蛇每隔10帧移动一次
    {
        moveSnake(); //  调用小蛇移动函数
        waitIndex = 1; // 再变成1
    }
}

void updateWithInput()  // 和输入有关的更新函数
{
    if (_kbhit() && isFailure == 0)  //  如果有按键输入，并且不失败
    {
        char input = _getch(); //  获得按键输入
        if (input == 'a' || input == 's' || input == 'd' || input ==
'w') // 如果是asdw
        {
            moveDirection = input;  // 设定移动方向
            moveSnake(); // 调用小蛇移动函数
        }
    }
}

int main() //  主函数
{
    srand((unsigned)time(NULL));
    startup();  // 初始化函数，仅执行一次
    while (1)   // 一直循环
    {
        show();  // 进行绘制
        updateWithoutInput(); // 和输入无关的更新
        updateWithInput();    // 和输入有关的更新
    }
    return 0;
}
```

出自童晶老师的博客

四、实验心得体会

开发了一个小游戏，学会了自顶向下变成。