实验一 实验报告

王子龙

CIT1808

18281218

Mar. 5th, 2019

目　录

1 .实验 A **3**

1.1 在第一个 FOR 循环执行前，监控到的各个变量 3
1.2 回答问题 4
1.3 程序修改 6

2 .实验 B **7**

2.1 代码测试 7
2.2 ANS 记录的数据 7
2.3 错误原因 7
2.4 代码修改 8
2.5 CHECK 函数的作用 8

3 .实验报告 **9**

3.1 实验中遇到的问题及解决方法 9
3.2 IDE 调试监控变量 9
3.3 STEP IN, STEP OVER, STEP OUT 的作用 10
3.4 非法内存访问 10
3.4.1 为什么会非法内存访问 *10*
3.4.2 与现实之中相似之处 *10*
3.5 关于内存泄漏 10

1 .实验 A

1.1 在第一个 for 循环执行前，监控到的各个变量

变量/常量名	变量值	变量/常量名	变量值	变量/常量名	变量值	变量/常量名	变量值
a	0x00007ffee26fc990	p	0x00007ffee26fc990	*p	0x00007ffee26fc990	*a	tu
a[0]	0x00007ffee26fc990	p[0]	0x00007ffee26fc990	*(p+0)	0x00007ffee26fc990	*(a+0)	0x00007ffee26fc990
a[1]	0x00007ffee26fc99c	p[1]	0x00007ffee26fc99c	*(p+1)	0x00007ffee26fc99c	*(a+1)	0x00007ffee26fc99c
a[2]	0x00007ffee26fc9a8	p[2]	0x00007ffee26fc9a8	*(p+2)	0x00007ffee26fc9a8	*(a+2)	0x00007ffee26fc9a8
&a[0][0]	0x00007ffee26fc990	p[0]+0	0x00007ffee26fc990	*p+0	0x00007ffee26fc990	*a+0	0x00007ffee26fc990
&a[0][1]	0x00007ffee26fc994	p[0]+1	0x00007ffee26fc994	*p+1	0x00007ffee26fc994	*a+1	0x00007ffee26fc994
&a[0][2]	0x00007ffee26fc998	p[0]+2	0x00007ffee26fc998	*p+2	0x00007ffee26fc998	*a+2	0x00007ffee26fc998
&a[1][0]	0x00007ffee26fc99c	p[1]+0	0x00007ffee26fc99c	*p+3	0x00007ffee26fc99c	*a+3	0x00007ffee26fc99c
&a[1][1]	0x00007ffee26fc9a0	p[1]+1	0x00007ffee26fc9a0	*p+4	0x00007ffee26fc9a0	*a+4	0x00007ffee26fc9a0
&a[1][2]	0x00007ffee26fc9a4	p[1]+2	0x00007ffee26fc9a4	*p+5	0x00007ffee26fc9a4	*a+5	0x00007ffee26fc9a4
&a[2][0]	0x00007ffee26fc9a8	p[2]+0	0x00007ffee26fc9a8	*p+6	0x00007ffee26fc9a8	*a+6	0x00007ffee26fc9a8
&a[2][1]	0x00007ffee26fc9ac	p[2]+1	0x00007ffee26fc9ac	*p+7	0x00007ffee26fc9ac	*a+7	0x00007ffee26fc9ac
&a[2][2]	0x00007ffee26fc9b0	p[2]+2	0x00007ffee26fc9b0	*p+8	0x00007ffee26fc9b0	*a+8	0x00007ffee26fc9b0

a[0][0]	1	*p[0]	1	**p	1	p[0][0]	1
a[0][1]	2	*(p[0]+1)	2	*(*p+1)	2	p[0][1]	2
a[0][2]	3	*(p[0]+2)	3	*(*p+2)	3	p[0][2]	3
a[1][0]	4	*p[1]	4	*(*p+3)	4	p[1][0]	4
a[1][1]	5	*(p[1]+1)	5	*(*p+4)	5	p[1][1]	5
a[1][2]	6	*(p[1]+2)	6	*(*p+5)	6	p[1][2]	6
a[2][0]	7	*p[2]	7	*(*p+6)	7	p[2][0]	7
a[2][1]	8	*(p[2]+1)	8	*(*p+7)	8	p[2][1]	8
a[2][2]	9	*(p[2]+2)	9	*(*p+8)	9	p[2][2]	9

1.2 回答问题

a) 编译不能通过，出现错误：“array type 'int [3][3]' is not assignable”。

b) 程序可以正常编译。

c) a 与 q 的相同点：a 与 q 的值都是数组 a[3][3]的首地址的值，但是 q 作为指针变量可以进行运算，然而 a 不行。

变量名	变量值	变量名	变量值	常量名	变量值
a	0x00007ffee26fc990	p	0x00007ffee26fc990	q	0x00007ffee10b5990
a[0]	0x00007ffee26fc990	*p	0x00007ffee26fc990	*q	1
a[1]	0x00007ffee26fc99c	p[0]	0x00007ffee26fc990	q[0]	1
a[2]	0x00007ffee26fc9a8	p[1]	0x00007ffee26fc99c	q[1]	2
&a[0][0]	0x00007ffee26fc990	p[2]	0x00007ffee26fc9a8	q[2]	3
&a[0][1]	0x00007ffee26fc994	p[3]	0x00007ffee10b59b4	q[3]	4
&a[0][2]	0x00007ffee26fc998	*p[0]	1	q[4]	5
&a[1][0]	0x00007ffee26fc99c	*p[1]	4	q[5]	6
&a[1][1]	0x00007ffee26fc9a0	*p[2]	7	q[6]	7
&a[1][2]	0x00007ffee26fc9a4	*p[3]	32766	q[7]	8
&a[2][0]	0x00007ffee26fc9a8	*p[9]	32766	q[8]	9
&a[2][1]	0x00007ffee26fc9ac	p[9]	0x00007ffee10b59fc	q[9]	32766
&a[2][2]	0x00007ffee26fc9b0	p[0][0]	1	q[0][0]	error: subscripted value is not an array,

					pointer, or vector
a[0][0]	1	p+1	0x00007ffee10b599c	q+1	0x00007ffee10b5994
a[0][1]	2	p+2	0x00007ffee10b59a8	q+2	0x00007ffee10b5998
a[0][2]	3	p+3	0x00007ffee10b59b4	q+3	0x00007ffee10b599c
a[1][0]	4	p+4	0x00007ffee10b59c0	q+4	0x00007ffee10b59a0
a[1][1]	5	p+9	0x00007ffee10b59fc	q+9	0x00007ffee10b59b4
a[1][2]	6	*(p+1)	{4,5,6}	*(q+1)	2
a[2][0]	7	*(p+9)	{32766,–519349304,32766}	*(q+9)	32766
a[2][1]	8	*p+1	0x00007ffee10b5994	*q+1	2
a[2][2]	9	*p+9	0x00007ffee10b59b4	*q+9	10
		*(*p+1)	2	*(*q+1)	error: indirection requires pointer operand ('int' invalid)
		*(*p+9)	32766	*(*q+1)	error: indirection requires pointer operand ('int' invalid)

	交换前矩阵值	**i**	**j**	**p**	**q**	**p+j**	***(p+j)**	***(*(p+j)+i**	***(*(p+j)+i)**	**q+j**	***(q+j)**	**交换后矩阵值**
1	$\begin{bmatrix}1&2&3\\4&5&6\\7&8&9\end{bmatrix}$	0	1	0x00007ffeeced9990	0x00007ffeeced9990	0x00007ffeeced999c	{2,5,6}	0x00007ffeeced999c	2	0x00007ffeeced9994	4	$\begin{bmatrix}1&4&3\\2&5&6\\7&8&9\end{bmatrix}$
2	$\begin{bmatrix}1&4&3\\2&5&6\\7&8&9\end{bmatrix}$	0	2	0x00007ffeeb846990	0x00007ffeeb846990	0x00007ffeeb8469a8	{3,8,9}	0x00007ffeeb8469a8	3	0x00007ffeeb846998	3	$\begin{bmatrix}1&4&7\\2&5&6\\3&8&9\end{bmatrix}$
3	$\begin{bmatrix}1&4&7\\2&5&6\\3&8&9\end{bmatrix}$	1	2	0x00007ffeeb846990	0x00007ffeeb846990	0x00007ffeeb8469a8	{3,7,9}	0x00007ffeeb8469ac	7	0x00007ffeeb846998	8	$\begin{bmatrix}1&4&8\\2&5&6\\3&7&9\end{bmatrix}$

1.3 程序修改

代码出错的原因为：在交换了矩阵的元素的值指挥才改变了指针 q 的值，所以实际上交换的是矩阵第一行第三个和第三行第二个元素，导致了转置错误。

应该先改变 q 的值再进行转置，即先修改指针 q 的值，再转置：

```
for (i = 0; i < 3; i++)    //遍历每一行
{
    q = q + i * 3;
    for(j = i + 1; j < 3; j++)
    {
        temp = *(*(p + j) + i);
        *(*(p + j) + i) = *(q + j);
        *(q + j) = temp;
    }
}
```

改正后的程序

```
1 4 7

2 5 8

3 6 9



Process finished with exit code 0
```

输出结果

2 .实验 B

2.1 代码测试

代码测试时，使用 IDE CLion 时，发现 control + D 在 IDE 中无法使用，因此采用了使用 Terminal 进行测试，得到以下结果：

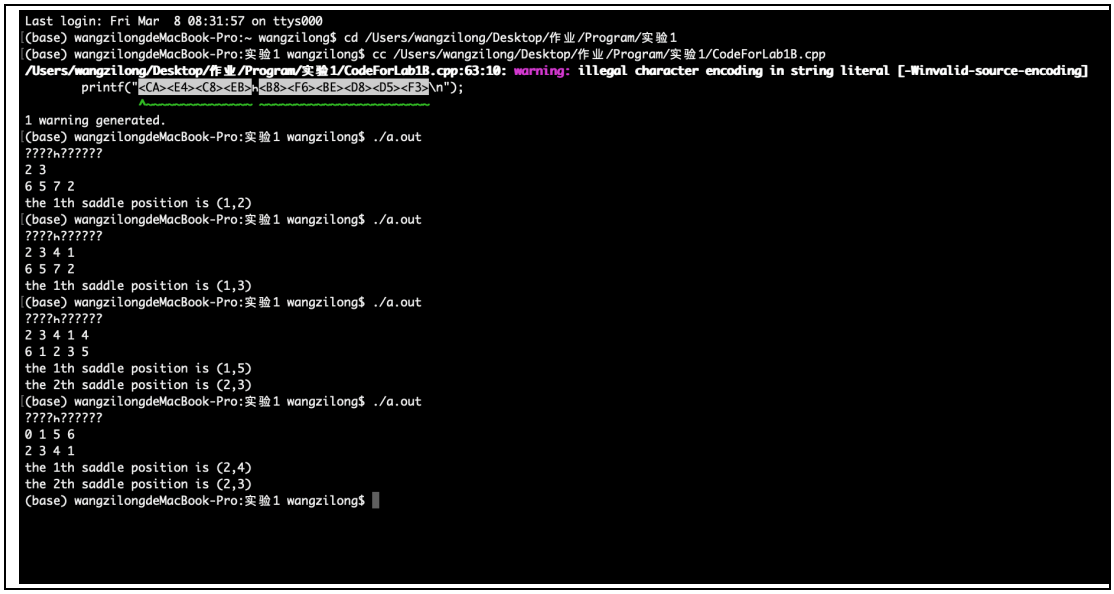

图 2-1 测试结果

	ColIndex	Non-zero elements
Data1	{2,1,2,0,0,···}	2,1,2
Data2	{2,2,2,0,0,···}	2,2,2
Data3:	{2,3,4,0,0,···}	2,3,4
Data4:	{2,3,2,0,0,···}	2,3,2

表 2-1 Experimental data

2.2 ans 记录的数据

Variables	Using
ans[0].x	记录一共有多少个鞍点
ans[x].x	记录第 x 个鞍点的行数
ans[y].x	记录第 x 个鞍点的列数

表 2-2

2.3 错误原因

在每一次判断后并没有清除 ColIndex 数组里的值，所以导致下一次判断时会把上一次

判断的最大值误认为是下一次判断的最大值。这就会导致寻找鞍点的值的过程会发生错误。

2.4 代码修改

```
void work()
{
    int i,j;
    int tem;
    int ColIndex[100] = {0};
    int k=1;
    for(i = 0; i < lenx; i = i + 1)
    {
        tem = 0;
        for(j = 1; j < leny; j = j + 1)
        {
            if(matrix[i][j] > matrix[i][tem])
            {
                tem = j;
            }
        }
        for(j = 0; j < leny; j = j + 1)
        {
            if(matrix[i][j] == matrix[i][tem])
            {
                ColIndex[0] = ColIndex[0] + 1;
                ColIndex[ColIndex[0]] = j;
            }
        }

        for(;k<=ColIndex[0];k=k+1)
        {
            if(check(i,ColIndex[k]) == 1)
            {
                remember(i,ColIndex[k]);
            }
        }
    }
}
```

代码修改：增加变量 k

2.5 check 函数的作用

Fucntion Check 用来检查 work 是否找出来的点是一列中最小的点。

3 .实验报告

3.1 实验中遇到的问题及解决方法

我使用的 IDE 是 JetBrains 公司开发的 CLion，然而 CLion 的 Console 存在着一个巨大的 BUG，即无法传入 EOF，故实验 B 一开始我只能在 Terminal 中测试结果，但是并不能进行调试，在疯狂 Google 之后，决定换一个 IDE，放弃研究，因此转到了 Xcode。

3.2 IDE 调试监控变量

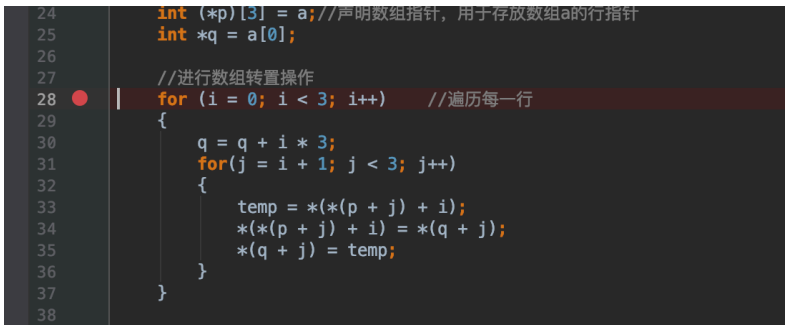

图 3-1 设置断点

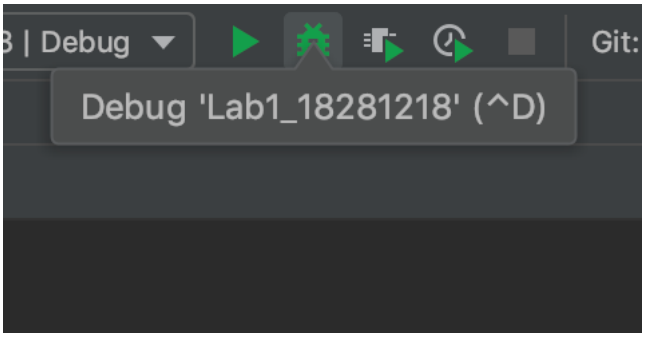

图 3-2 进行调试

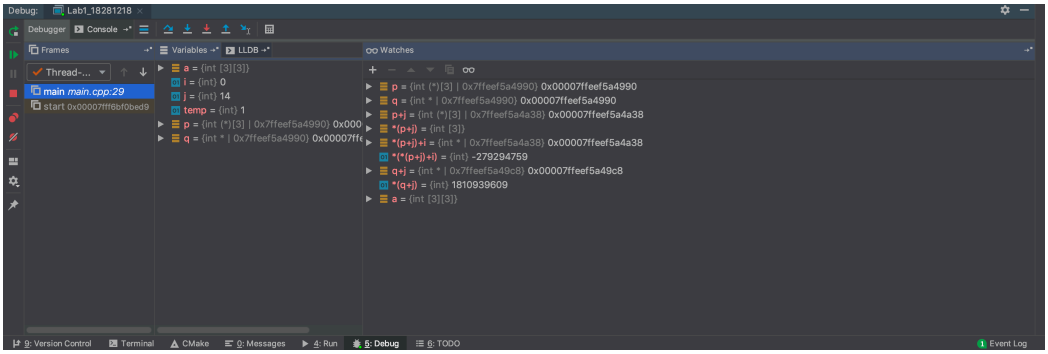

图 3-3 监控变量的值

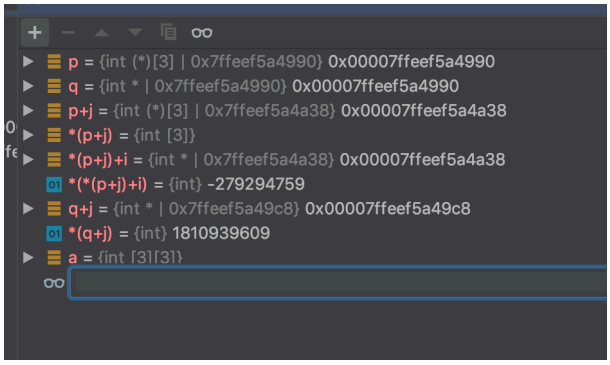

图 3-4 增加监控变量

3.3 step in, step over, step out 的作用

step-in	单步执行
step-over	单步执行（不进入子程序）
step-out	执行完子程序剩余部分，回到主函数

表 3-1

3.4 非法内存访问

3.4.1 为什么会非法内存访问

原因一、访问越界。
原因二、访问无效地址

3.4.2 与现实之中相似之处

一、我家很大，面积足够，但我非得访问不属于我家的邻居家
二、北京很大，面积足够，但我非得访问北京七环，八道口这种不存在的地方

3.5 关于内存泄漏

开发人员进行程序开发的过程使用动态存储变量时，不可避免地面对内存管理的问题。程序中动态分配的存储空间，在程序执行完毕后需要进行释放。没有释放动态分配的存储空间而造成内存泄漏，是使用动态存储变量的主要问题。一般情况下，开发人员使用系统提供的内存管理基本函数，如 malloc、recalloc、calloc、free 等，完成动态存储变量存储空间的分配和释放。但是，当开发程序中使用动态存储变量较多和频繁使用函数调用时，就会经常发生内存管理错误，例如：
分配一个内存块并使用其中未经初始化的内容；
释放一个内存块，但继续引用其中的内容；

子函数中分配的内存空间在主函数出现异常中断时、或主函数对子函数返回的信息使用结束时，没有对分配的内存进行释放；

程序实现过程中分配的临时内存在程序结束时，没有释放临时内存。内存错误一般是不可再现的，开发人员不易在程序调试和测试阶段发现，即使花费了很多精力和时间，也无法彻底消除。