预习作业 1

杨锐 2011189

1.

实验平台：Visual Studio 2022

实验方案：针对两种不同的寻址方式编写了计时代码进行测试，同时为了避免偶然性利用循环进行了 20 轮测试，具体代码如下：

```
#include<iostream>
#include<windows.h>
using namespace std;
const int N = 10240;       // matrix size
double a[N];

void start(int n) {
    for (int i = 0; i < n; i++) {
        a[i] = i * 100 ;
    }
}

int main() {
    int i = 0;
    long long head, tail, freq;        // timers
    for (int j = 0; j < 20; j++) {
        cout << "测试次数：" << j + 1 << endl;
        start(N);
        QueryPerformanceFrequency((LARGE_INTEGER*)&freq);    // similar to
CLOCKS_PER_SEC
        QueryPerformanceCounter((LARGE_INTEGER*)&head); // start time
        for (i = 0; i < N; i++) {
            a[i] = a[i] * 2000;
            a[i] = a[i] / 10000;
        }
        QueryPerformanceCounter((LARGE_INTEGER*)&tail); // end time
        cout << "Loop1: " << (tail - head) * 1000.0 / freq << "ms" << endl;

        QueryPerformanceFrequency((LARGE_INTEGER*)&freq);    // similar to
CLOCKS_PER_SEC
        QueryPerformanceCounter((LARGE_INTEGER*)&head); // start time
        double* b = a;
        for (i = 0; i < N; i++) {
            *b = *b * 2000;
            *b = *b / 10000;
            b++;
        }
        QueryPerformanceCounter((LARGE_INTEGER*)&tail); // end time
```

```
        cout << "Loop2: " << (tail - head) * 1000.0 / freq << "ms" << endl;
    }//测试 20 轮数据
}
```

首先创建了一个大小为 10240 的 double 数组，并给每个值赋值为 i*100，然后调用计时函数分别对直接使用数组下标进行寻址和使用指针寻址进行测试。

实验结果：程序输出的结果如下所示

测试次数：1	测试次数：11
Loop1: 0.0133ms	Loop1: 0.0111ms
Loop2: 0.0114ms	Loop2: 0.0105ms
测试次数：2	测试次数：12
Loop1: 0.0112ms	Loop1: 0.0113ms
Loop2: 0.0194ms	Loop2: 0.0104ms
测试次数：3	测试次数：13
Loop1: 0.0358ms	Loop1: 0.0111ms
Loop2: 0.0106ms	Loop2: 0.0105ms
测试次数：4	测试次数：14
Loop1: 0.0114ms	Loop1: 0.0112ms
Loop2: 0.0107ms	Loop2: 0.0104ms
测试次数：5	测试次数：15
Loop1: 0.012ms	Loop1: 0.0111ms
Loop2: 0.0109ms	Loop2: 0.0132ms
测试次数：6	测试次数：16
Loop1: 0.0112ms	Loop1: 0.0111ms
Loop2: 0.0108ms	Loop2: 0.0104ms
测试次数：7	测试次数：17
Loop1: 0.0113ms	Loop1: 0.0111ms
Loop2: 0.0103ms	Loop2: 0.0123ms
测试次数：8	测试次数：18
Loop1: 0.0113ms	Loop1: 0.0113ms
Loop2: 0.0106ms	Loop2: 0.0104ms
测试次数：9	测试次数：19
Loop1: 0.0112ms	Loop1: 0.0112ms
Loop2: 0.0108ms	Loop2: 0.0104ms
测试次数：10	测试次数：20
Loop1: 0.0112ms	Loop1: 0.0111ms
Loop2: 0.0105ms	Loop2: 0.0105ms

可以看出，在大多数情形下，Loop1 的耗时都将比 Loop2 长，但也有少数情况结果相反。

结果分析：在第一种方式，直接使用 a[i]寻址时，在每次计算结束后的循环都要对数组进行寻址，而第二种方式每次计算结束后直接将指针指向下一位，效率会更高一些。

更完整的测试方案：

（1）可以分析数组规模对结果造成的影响，如下方将数组大小进一步扩大到50000，得到如下的结果

测试次数：1 Loop1: 0.0598ms Loop2: 0.0831ms 测试次数：2 Loop1: 0.0531ms Loop2: 0.0502ms 测试次数：3 Loop1: 0.0531ms Loop2: 0.0721ms 测试次数：4 Loop1: 0.0645ms Loop2: 0.0753ms 测试次数：5 Loop1: 0.053ms Loop2: 0.0507ms 测试次数：6 Loop1: 0.0679ms Loop2: 0.0504ms 测试次数：7 Loop1: 0.053ms Loop2: 0.0864ms 测试次数：8 Loop1: 0.0651ms Loop2: 0.0505ms 测试次数：9 Loop1: 0.0529ms Loop2: 0.0776ms 测试次数：10 Loop1: 0.053ms Loop2: 0.0506ms	测试次数：11 Loop1: 0.053ms Loop2: 0.0508ms 测试次数：12 Loop1: 0.1195ms Loop2: 0.0504ms 测试次数：13 Loop1: 0.1798ms Loop2: 0.0525ms 测试次数：14 Loop1: 0.053ms Loop2: 0.0788ms 测试次数：15 Loop1: 0.0647ms Loop2: 0.1744ms 测试次数：16 Loop1: 0.0579ms Loop2: 0.0505ms 测试次数：17 Loop1: 0.0531ms Loop2: 0.1745ms 测试次数：18 Loop1: 0.0741ms Loop2: 0.0507ms 测试次数：19 Loop1: 0.0652ms Loop2: 0.0507ms 测试次数：20 Loop1: 0.0532ms Loop2: 0.0506ms

在多数情形下，Loop2 的优势更为明显，不过依然存在部分特殊情况。

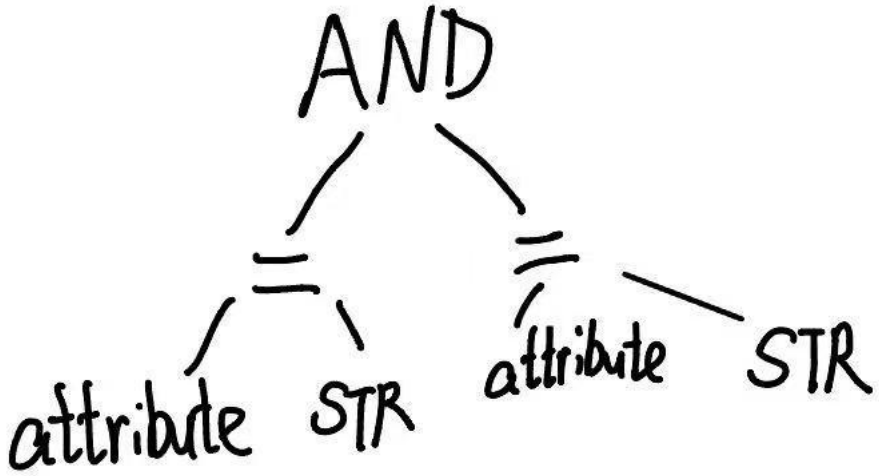
此外，还可以考虑利用多次调用数组内的值来实现提高算法的复杂度，向$\theta(n^2)$等拓展进行测试。

（2）调整编译器优化级别，如测试-O0、-O1、-O2、-O3、-Og、-Os、-Ofast 等方式，此外还可以更换不同的编译器、机器进行测试。

2.

分词：attribute=STR AND attribute=STR

语法树：



3.

使用 splint 检查及代码分析得到的部分主要问题如下：

（1）

```
cpp(11,11): Stack-allocated storage &loc reachable from return value: &loc
A stack reference is pointed to by an external reference when the function
returns. The stack-allocated storage is destroyed after the call, leaving a
dangling reference. (Use -stackref to inhibit warning)
```

```
{  int sa[2] = { 0, 1 };
   int loc = 3;
   glob = &loc;
   *x = &sa[0];
   return &loc;
```

这一部分的代码 return 的是堆栈分配的变量，属于是局部变量。

（2）

```
cpp(15,9): Comparison of unsigned value involving zero: i >= 0
An unsigned value is used in a comparison with zero in a way that is either a
bug or confusing. (Use -unsignedcompare to inhibit warning)
```

```
unsigned int i;
if (i >= 0)
```

这里的 i 是无符号整形 i，但是将其进行了 i>=0 的比较。

4.

（1）若 a,b 是标识符，则 a,b 是标识符列表

（2）若 int a 是变量声明语句，则 int a,b,...,x 也是变量声明语句