

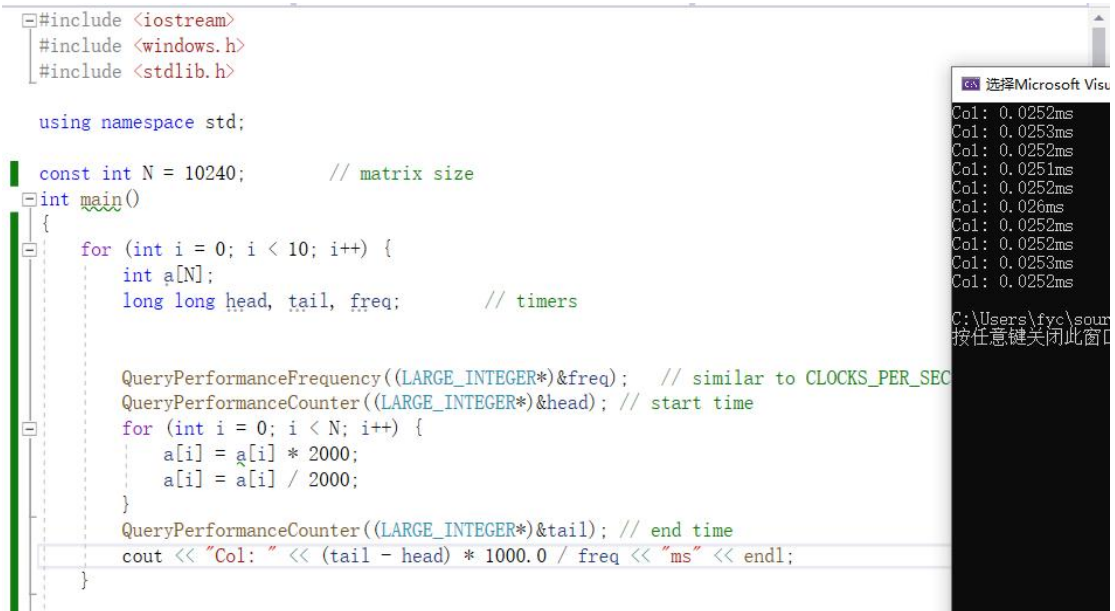
## 预习作业 1-4

### 2011696 付寅聪 物联网工程

#### 一 c 程序优化

1.实验平台: windows 平台下的 vs2022

2.实验方案:使用 windows 下的计时机制 query performance 分别对两端代码的运行速度进行计时,以此来判断运行效率.



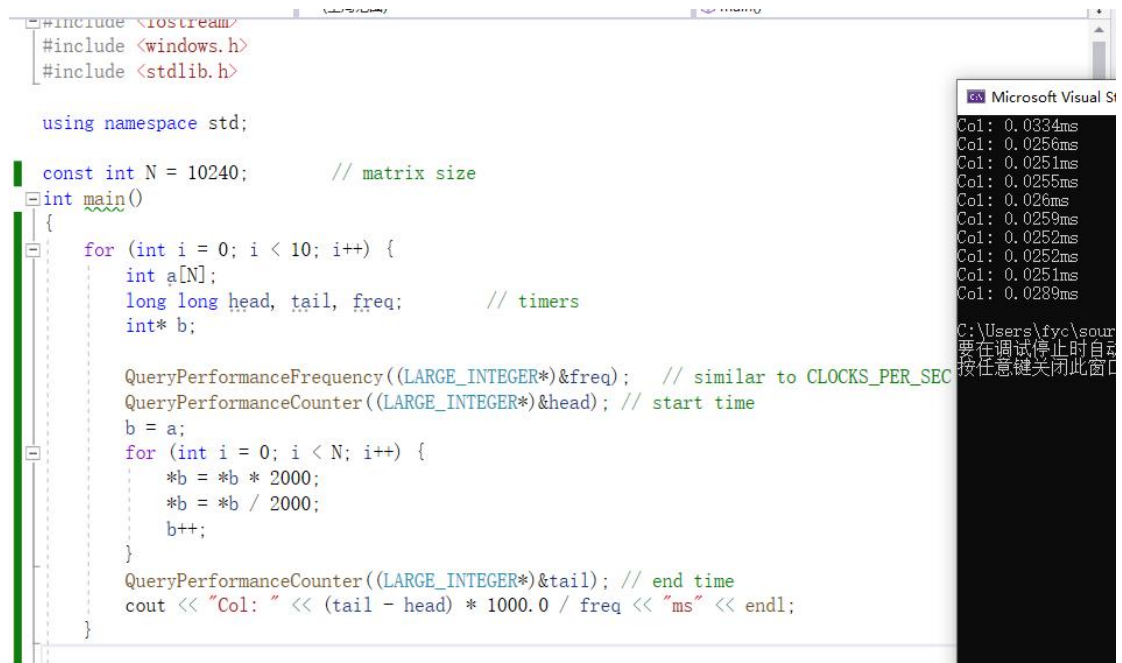
```
#include <iostream>
#include <windows.h>
#include <stdlib.h>

using namespace std;

const int N = 10240;          // matrix size
int main()
{
    for (int i = 0; i < 10; i++) {
        int a[N];
        long long head, tail, freq;    // timers

        QueryPerformanceFrequency((LARGE_INTEGER*)&freq); // similar to CLOCKS_PER_SEC
        QueryPerformanceCounter((LARGE_INTEGER*)&head); // start time
        for (int i = 0; i < N; i++) {
            a[i] = a[i] * 2000;
            a[i] = a[i] / 2000;
        }
        QueryPerformanceCounter((LARGE_INTEGER*)&tail); // end time
        cout << "Col: " << (tail - head) * 1000.0 / freq << "ms" << endl;
    }
}
```

Col: 0.0252ms  
Col: 0.0253ms  
Col: 0.0252ms  
Col: 0.0251ms  
Col: 0.0252ms  
Col: 0.026ms  
Col: 0.0252ms  
Col: 0.0252ms  
Col: 0.0253ms  
Col: 0.0252ms



```
#include <iostream>
#include <windows.h>
#include <stdlib.h>

using namespace std;

const int N = 10240;          // matrix size
int main()
{
    for (int i = 0; i < 10; i++) {
        int a[N];
        long long head, tail, freq;    // timers
        int* b;

        QueryPerformanceFrequency((LARGE_INTEGER*)&freq); // similar to CLOCKS_PER_SEC
        QueryPerformanceCounter((LARGE_INTEGER*)&head); // start time
        b = a;
        for (int i = 0; i < N; i++) {
            *b = *b * 2000;
            *b = *b / 2000;
            b++;
        }
        QueryPerformanceCounter((LARGE_INTEGER*)&tail); // end time
        cout << "Col: " << (tail - head) * 1000.0 / freq << "ms" << endl;
    }
}
```

Col: 0.0334ms  
Col: 0.0256ms  
Col: 0.0251ms  
Col: 0.0255ms  
Col: 0.026ms  
Col: 0.0259ms  
Col: 0.0252ms  
Col: 0.0252ms  
Col: 0.0251ms  
Col: 0.0289ms

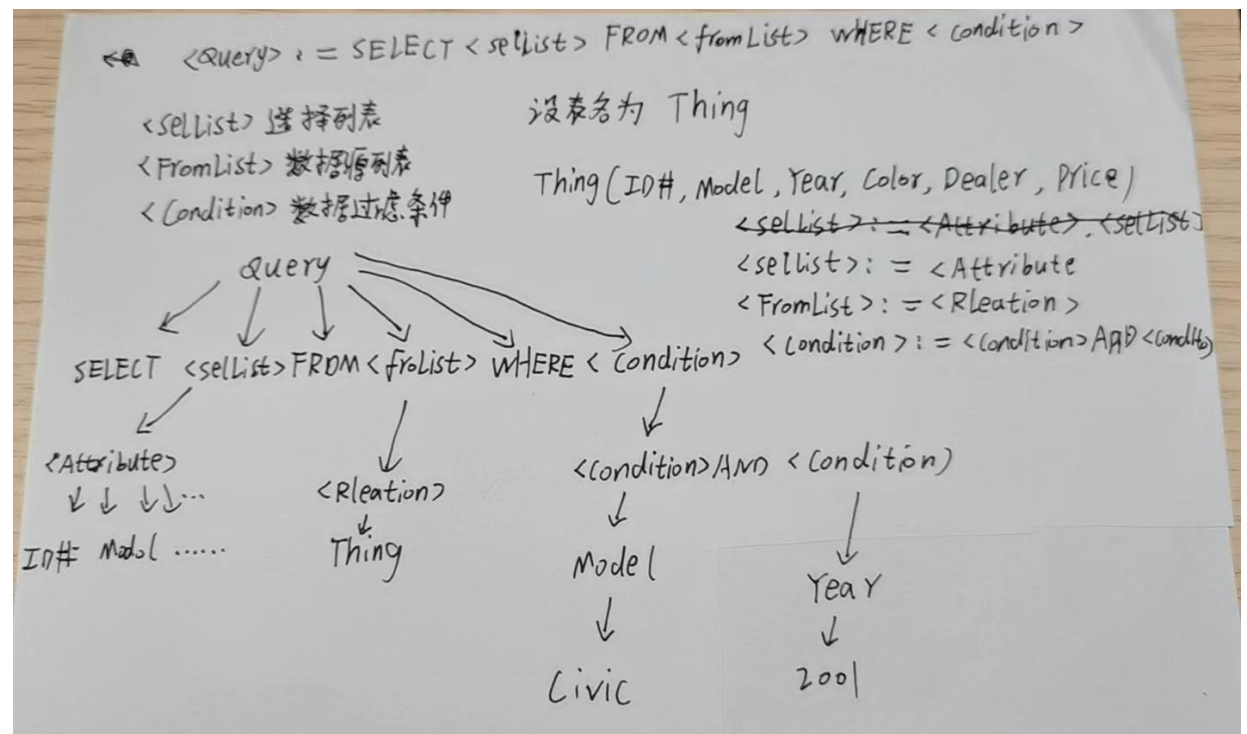
3. 通过此次实验,我选择下标法访问数组,指针是直接指向数据存储地址,访问时指针是直接访问,数组为间接访问,一般情况下指针应该比下标法访问速度快,但是也不安全.并且由此实

验结果可以看出,这个程序下使用下标法比指针法更快,也更稳定,所以我选择下标法.

#### 4. 思考

可以对比不同编程风格的代码写法,通过不同的编译器和编译优化参数,通过编译器生成汇编代码,静态分析所生成的汇编代码效率,也可以在不同平台上进行测试,得出较准确的结果.

## 二 分词、构造语法树



### 三 静态检查

```
char firstChar1 (char *s)
{
    return *s;
}
```

引用没有指向任何内存地址的指针,也就是使用了一个没有赋值的指针.

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

返回局部变量的地址

```
void h(void)
{ unsigned int i;
  if (i >= 0)
    printf(">=0\n");
  else printf("<0\n");
}
```

i 未初始化

## 四 语法描述

1. `a` 是标识符
2. `b` 是标识符
3. 若 `int a, b, ..., x` 是标识符列表,则 `x` 也是标识符