编译系统原理第一章预习作业

计科一班 2013536 汤清云

1. C 程序优化：

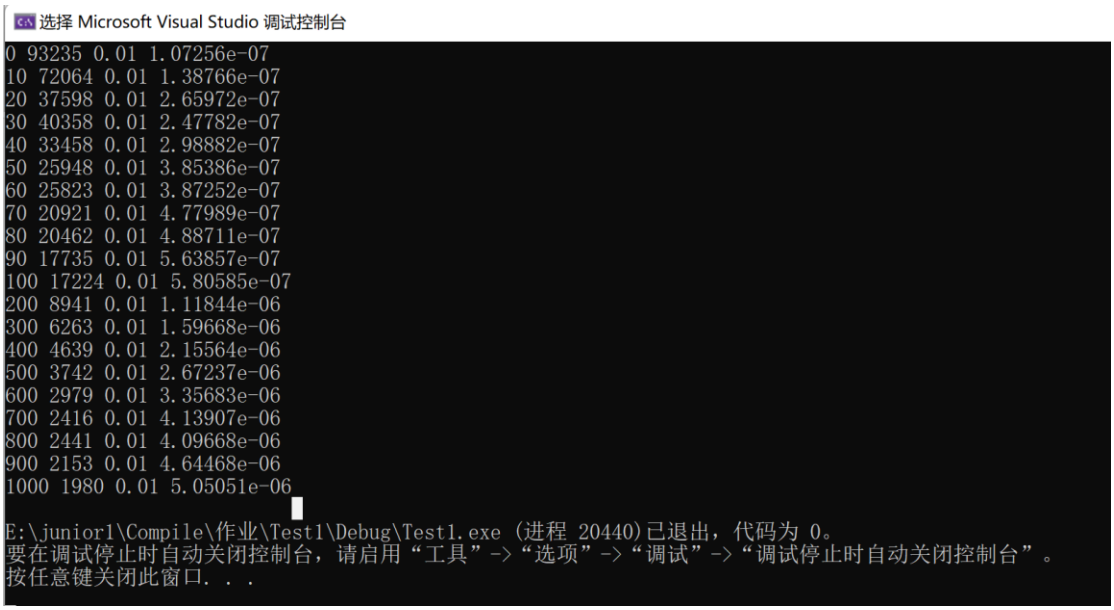
代码具体如下：

```
#include<iostream>
#include<windows.h>
#include<stdlib.h>
#include<time.h>
using namespace std;
int main()
{
    int N=1000;
    //cin >> N;
    int* a = new int[N];
    long counter;
    float seconds;
    int step = 10;
    clock_t start, finish;
    start = clock();
    counter = 0;
    //初始化
    for (int i = 0;i < N;i++)
    {
        a[i] = 100;
    }
    /*loop1
    for (N = 0; N <= 1000; N += step) {
        // get time for size n
        start = clock();
        counter = 0;
        while (clock() - start < 10) {
            counter++;
            for (int i = 0;i < N;i++)
            {
                a[i] = a[i] * 2000;
                a[i] = a[i] / 10000;
            }
        }
        finish = clock();
        seconds = (finish - start) / float(CLOCKS_PER_SEC);
        cout << N << ' ' << counter << ' ' << seconds<< ' ' << seconds /
```

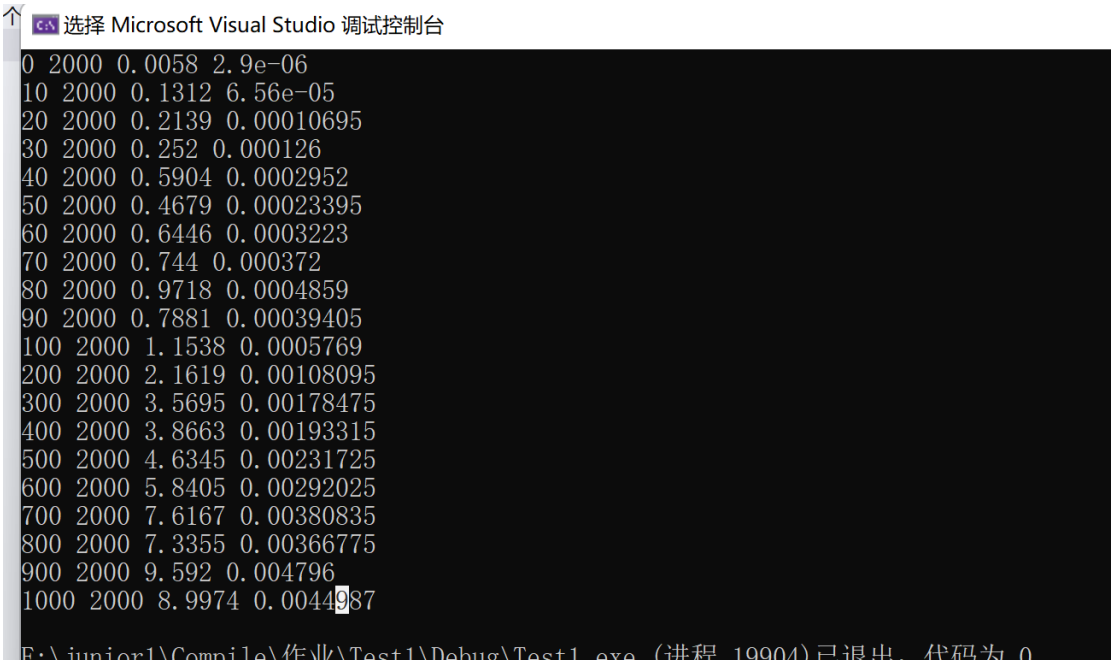
```
counter << endl;
        if (N == 100) step = 100;
    }
    */

    /*loop2*/
    int* b;
    b = a;
    for (N = 0; N <= 1000; N += step) {
        // get time for size n
        start = clock();
        counter = 0;
        while (clock() - start < 10) {
            counter++;
            for (int i = 0;i < N;i++)
            {
                *b = (*b) * 2000;
                *b = *b / 10000;
                b++;
            }
            b = a;
        }
        finish = clock();
        seconds = (finish - start) / float(CLOCKS_PER_SEC);
        cout << N << ' ' << counter << ' ' << seconds << ' ' << seconds /
counter << endl;
        if (N == 100) step = 100;
    }


}
```

测试结果如下：

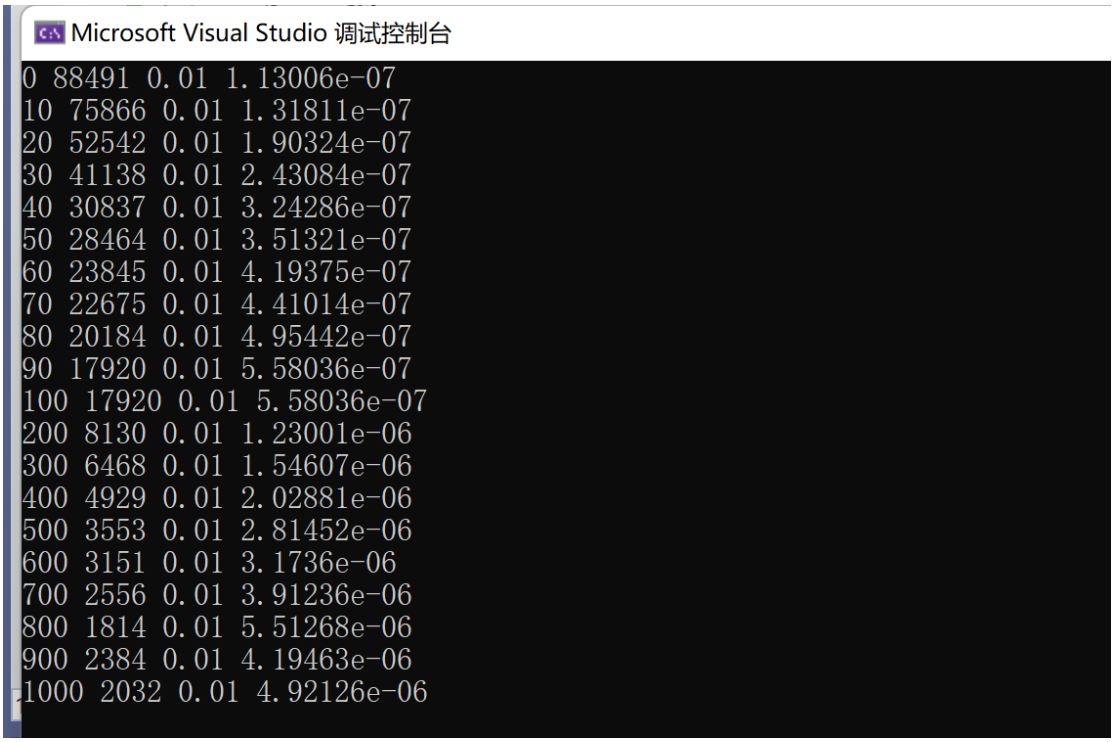
对于第一种程序，N 的取值、测试次数、总共运行时间/s、平均运行时间/s 如下：



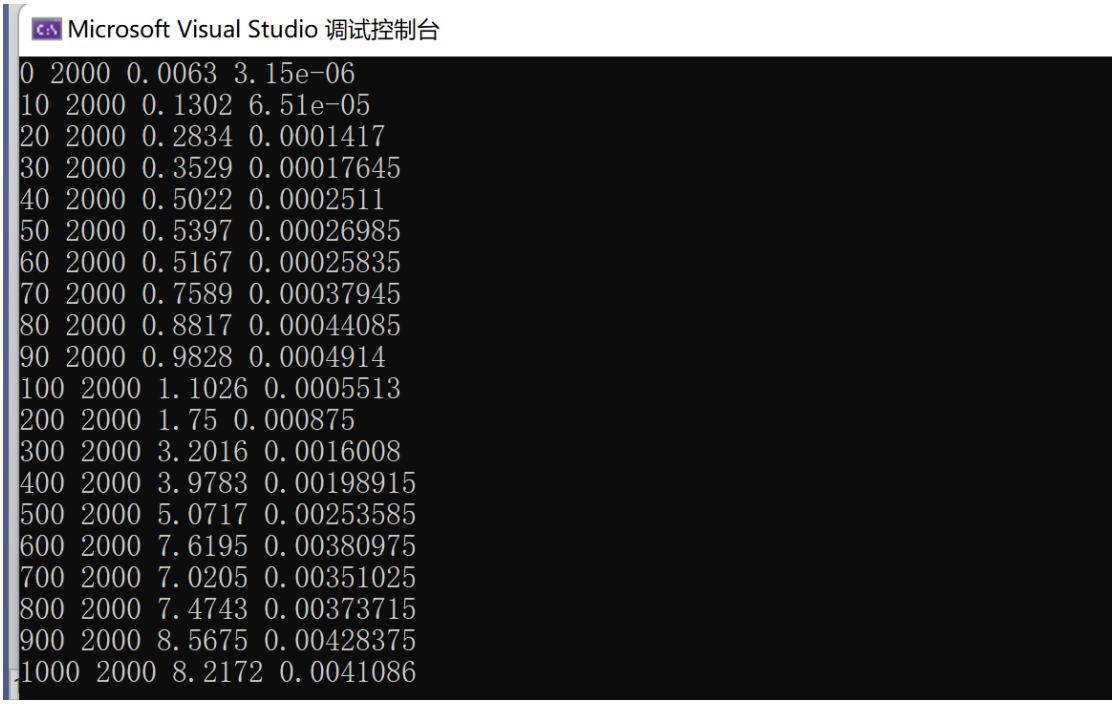
精确计时（单位毫秒）：



对于第二种程序，N 的取值、测试次数、总共运行时间/s、平均运行时间/s 如下：

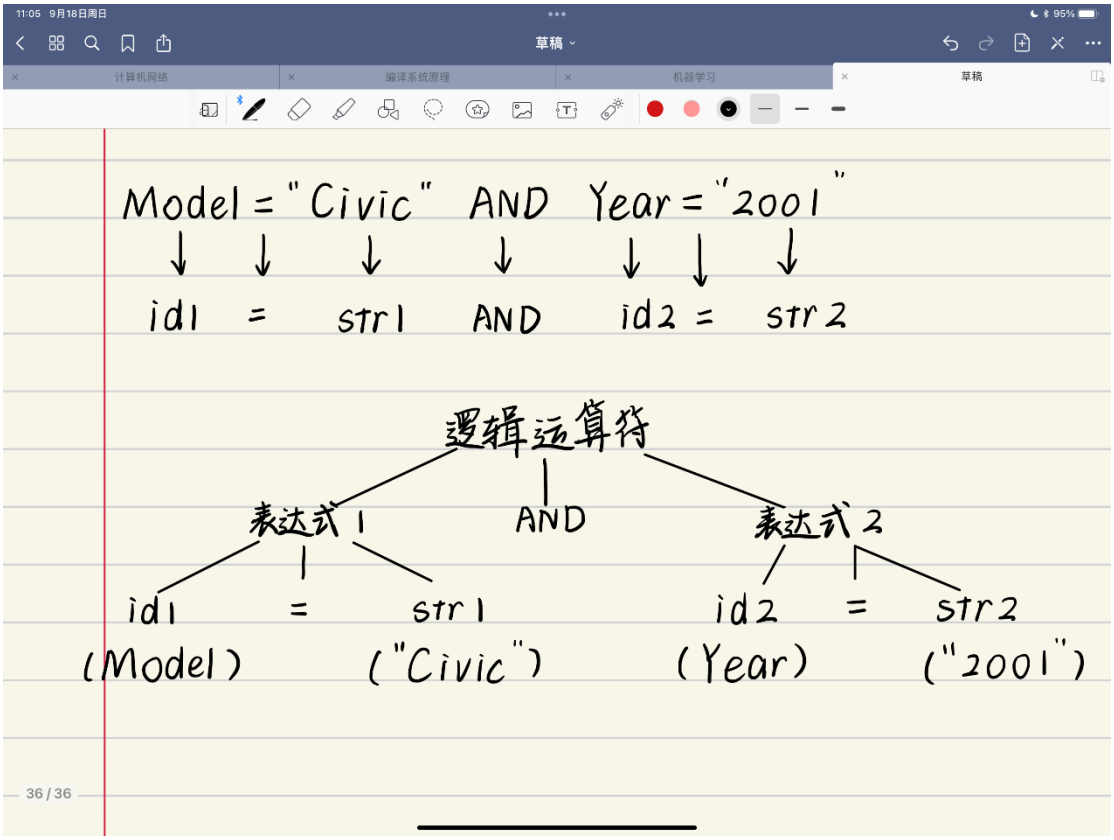


精确测试（单位毫秒）：



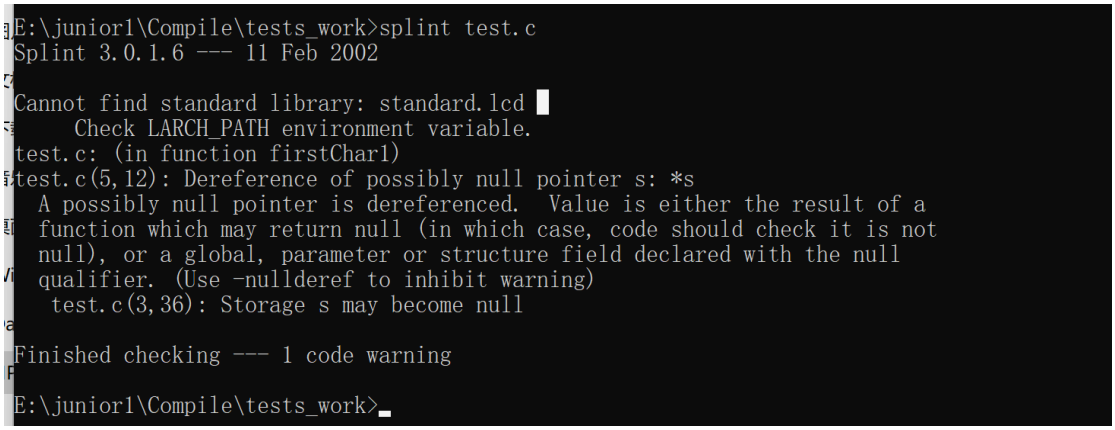
实践所得，第二种更快，但是测试结果并不稳定。

2. 分词、构造语法树：



3. 静态检查：

```
#include <stdio.h>

char firstChar1 ( /*@null@*/ char *s)
{
    return *s;#可能为空的指针，函数可能返回 null
}


int main()
{


    return 0;
}
```




```
#include <stdio.h>

int *glob;
int *f (int **x)
{    int sa[2] = { 0, 1 };
     int loc = 3;
     glob = &loc;
     *x = &sa[0];#三、可从参数 x 访问堆栈分配的存储*x 而存储*x 变成堆栈
     return &loc;# 一、当函数返回时，由外部引用指向堆栈引用。调用之后，堆栈分配的存储被销毁，留下一个悬空引用；二、直接地址&loc 仅隐式返回:&loc 与直接地址(& operator 的结果)传输不一致。
}
#四、函数返回引用已释放存储的全局 glob 在传递控制时，全局变量不满足其注释。
#五、声明被导出，但不在此模块之外使用。声明可以使用静态限定符。
int main()
{



     return 0;
}
```

```
E:\junior1\Compile\tests_work>splint test.c
Splint 3.0.1.6 --- 11 Feb 2002

Cannot find standard library: standard.lcd
     Check LARCH_PATH environment variable.
test.c: (in function f)
test.c(9,11): 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)
test.c(9,11): Immediate address &loc returned as implicitly only: &loc
  An immediate address (result of & operator) is transferred inconsistently.
  (Use -immediatetrans to inhibit warning)
test.c(9,16): Stack-allocated storage *x reachable from parameter x
   test.c(8,4): Storage *x becomes stack
test.c(9,16): Function returns with global glob referencing released storage
  A global variable does not satisfy its annotations when control is
  transferred. (Use -globstate to inhibit warning)
   test.c(9,11): Storage glob is released
test.c(3,6): Variable exported but not used outside test: glob
  A declaration is exported, but not used outside this module. Declaration can
  use static qualifier. (Use -exportlocal to inhibit warning)

Finished checking --- 5 code warnings
```

```
#include <stdio.h>
void h(void)
{   unsigned int i;
     if (i >= 0) #无符号值 i 与 0 不可以比较，且 i 未定义
        printf(">=0\n");
     else printf("<0");#不可识别的标识符
}
int main()
{
    return 0;
}
```

```
E:\junior1\Compile\tests_work>splint test.c
Splint 3.0.1.6 --- 11 Feb 2002

Cannot find standard library: standard.lcd
     Check LARCH_PATH environment variable.
test.c: (in function h)
test.c(4,9): 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)
test.c(4,9): Variable i used before definition
  An rvalue is used that may not be initialized to a value on some execution
  path. (Use -usedef to inhibit warning)
test.c(5,7): Unrecognized identifier: printf
  Identifier used in code has not been declared. (Use -unrecog to inhibit
  warning)

Finished checking --- 3 code warnings
```

4. 标识符列表的递归定义：

基本规则：不以数字开头的只包含字母、下划线和数字的变量名是标识符；单个标识符即是一个标志符列表。

递归定义：标志符列表+英文逗号+标志符列表的组合结果也是标志符列表