**Звіт з першої лабораторної з АОС Калюжний Денис К-23**

**Мета лабраторної**: розробити програму, яка вимірює кількість виконуваних базових операцій (команд) за секунду конкретною ОбСист (комп’ютер + ОС + Система програмування). Типи для тестів: *char*, *int*, *long*, *float* та *double*.

**Середа розробки**: Microsoft Visual Studio 2017.

**Опис розробки:** перш за все організуємо клас тестів. Зробимо його template, оскільки класи для різних типів відрізнятимуться лише типом даних, над якими виконуються операції. Клас має таке тіло:

template <class T> class Tests

{

T a1 = 111, a2 = 22, a3 = 333, a4 = 4444, a5 = 555, a6 = 666, a7 = 77, a8 = 888888, a9 = 9999999, a10 = 101;

double empty\_time\_div();

double empty\_time\_other(); //other = add + subtr + div

public:

double add\_time();

double subtr\_time();

double mult\_time();

double div\_time();

};

В ньому зберігаються дані, над якими проводяться операції, та функції підрахунку часу виконання кожної операції.

Реалізація таких функцій ідентична з точністю до знаку операції:

template <class T> double Tests<T>::add\_time()

{

clock\_t start = clock();

for (unsigned long long i = 1; i < ITERATIONS; i++)

{

a1 = 111 % i + 1; a2 = 22 % i + 1;

a3 = a1 + a2; a4 = a2 + a3; a5 = a3 + a4; a6 = a4 + a5;

a7 = a5 + a6; a8 = a6 + a7; a9 = a7 + a8; a10 = a8 + a9;

a1 = a9 + a10; a2 = a10 + a1;

}

clock\_t end = clock();

return end - start - empty\_time\_other();

}

В циклі виконується певна кількість операцій одного типу (було вирішено робити по 10 000 000 ітерацій та використовувати 10 змінних). При "нарощуванні" тіла циклу потрібно враховувати ряд моментів. Перш за все, вони пов’язані із оптимізацією об’єктного коду сучасними компіляторами, що фактично означає необов’язкову відповідність  дій у вхідній програмі діям у об’єкному коді. Зайвий код, що не призводить до зміни даних, компілятором просто викидається. Тому у циклі реалізована більш складна логіка обчислення. Час виконання циклу вимірюється. При повертанні отриманого часу також враховується витрати на виконання самого циклу та операцій присвоювання, отже даний час вираховується (його повертає empty\_time\_other()). Зазначимо ще те, що тіло циклу для операції ділення трішки відрізняється від тіл додавання, віднімання та множення (Це пов’язано із запобіганням отримання нуля під час цілочисельного ділення). Тому, відповідно, я окремо виділив empty\_time\_div().

Далі створимо у програмі змінні, яким присвоїмо результат виконання відповідних функцій з підрахунку часу.

const double ADD\_TIME\_INT = Tests<int>().add\_time();

const double SUBTR\_TIME\_INT = Tests<int>().subtr\_time();

const double MULT\_TIME\_INT = Tests<int>().mult\_time();

const double DIV\_TIME\_INT = Tests<int>().div\_time();

const double ADD\_TIME\_LONG = Tests<long>().add\_time();

. . . . . . . . . .

Виявилось, що при такому підході принциповою різницею є локалізація цих змінних. Не є бажаним їх розташування у global namespace, бо компіляція такого коду (статичних змінних) суттєво відрізняється (від нестатичних). Після того, як вищезазначені значення обчислені, залишається лише організувати зворотній зв’язок із користувачем – вивести отриману статистику.

Для підрахунку швидкості виконання у відсотках (відносно найкращого результату серед усіх типів) використовується така формула:

percent = BEST\_TIME\_EVER / duration\_sec \* 100;

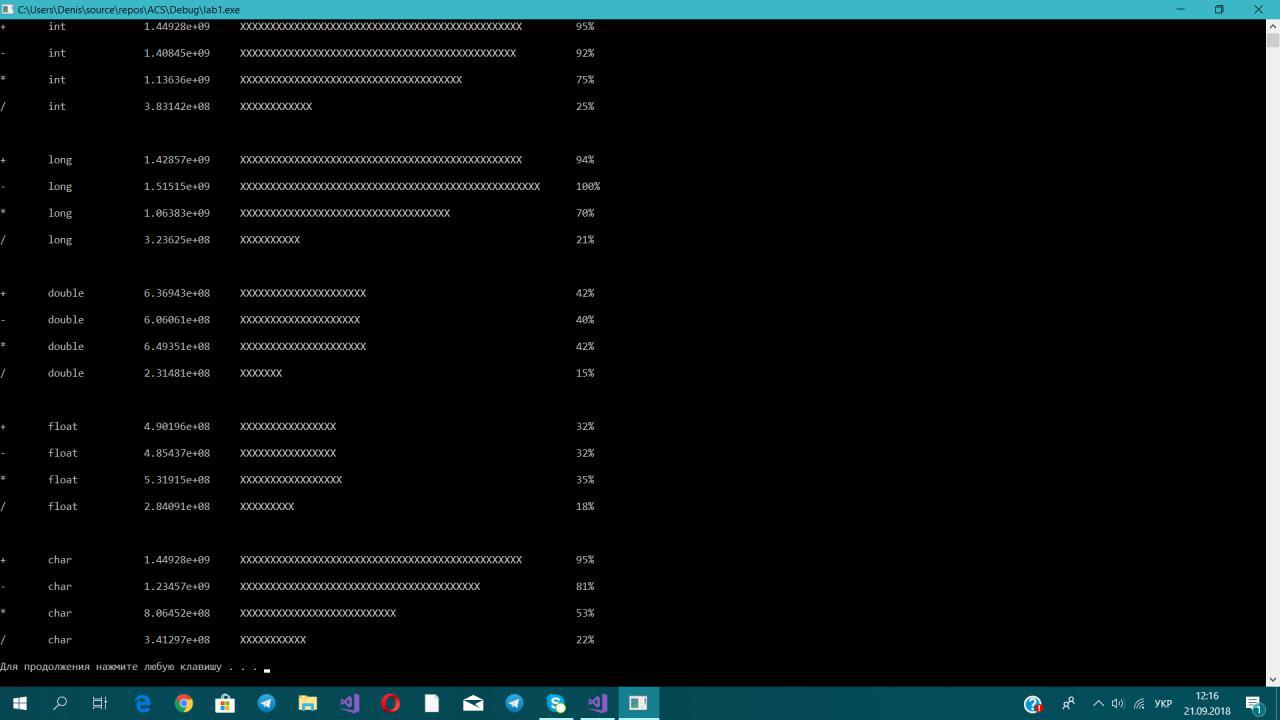
де BEST\_TIME\_EVER – найшвидший результат, duration\_sec – результат швидкості для поточного типу.

Для підрахунку кількості операцій одного типу, виконаних за секунду, використовується така формула:

oper\_per\_sec = ITERATIONS \* OPERATIONS \* TACT\_PER\_SECOND / duration\_sec;

Де ITERATIONS – кількість ітерацій в циклі, OPERATIONS – кількість операцій в циклі, TACT\_PER\_SECOND – кількість тактів за секунду, що виконує тестуючий комп’ютер (значення дістається із бібліотеки), duration\_sec – результат швидкості для поточного типу.

Отже, вивід програми має такий вигляд:



Де 1й стовпчик відповідає знаку операції, 2й – типу данних, 3й - oper\_per\_sec, 4й – діаграма, що відповідає percent, 5й – percent.

**Висновок:** Найшвидшими є операції int та long. При чому можна побачити, що операції «плюс» та «мінус» кожного з типів є найшвидшими та мають приблизно однаковий час виконання (серед одного типу). Операція множення завжди повільніша за останні, а операція ділення найповільніша серед усіх. Тип double та float показали себе як найповільніші. Також треба зазначити, що при даних тестуваннях є вірогідність отримання надвисокої похибки через велику складність сучасних комп’ютерів та можливість псування статистики через виконання комп’ютером інших задач паралельно під час проведення тестувань.