REPORT

Etienne PAPEGNIES
Remy SAKSIK
Hamed SARGAZI
Naima EL MISSOURI
Avignon University
France

31st March 2012

Contents

Introduction

In this paper we describe the test procedures. We use Valgring for memory test and gprof for profiling.

To achieve these we proceed to test two test two and test memories profiling. In a first test we realized that a loop is usually a 10 to * any operations on numbers randomly
(created thanks to the function 'rand ()'), which are each 4096 bytes. In a second test we realized that a loop is usually a 50 to * any operations on random numbers.

nota:Every time you spend in a loop we create objects of class LongInt.
EX: main.cpp
for(int i=0; i¡ 10; i++)
LongInt a; LongInt b;

LongInt i(32), j(32), k(32), l(32), m(32);

i = a; j = a; k = a; l = a; m = a;

i += b; j -= b; k *= b; l /= b; m %= b;

for(int i=0; i¡ 50; i++)
LongInt a; LongInt b;

LongInt i(32), j(32), k(32), l(32), m(32);

i = a; j = a; k = a; l = a; m = a;

i += b; j -= b; k *= b; l /= b; m %= b;

When an object is created it is assigned a random number with the

function 'rand ()' cf : default constructor
    char buffer[4096];
srand (time(NULL));
generate secret number:
for(k=0; k¡4095; k++)
buffer[k] = rand()%10 + 48;
buffer[4095]='\0';

Plan de test

1 coreAdd(LongInt)

	YES	NO
The sum is she corectement performed?	X	

2 coreSub(LongInt)

	YES	NO
The substraction is she corectement performed?	X	

3 coreMul(LongInt)

	YES	NO
The multiplication is she corectement performed?	X	

4 coreDiv(LongInt)

	YES	NO
The divison is she corectement performed?	X	

5 mod(LongInt)

	YES	NO
The modulo is it corectement performed?	X	

Valgring

6 First memory test

==2129== HEAP SUMMARY:
==2129== in use at exit: 12,520 bytes in 1 blocks
==2129== total heap usage: 211 allocs, 210 frees, 633,038 bytes allocated
==2129==
==2129== LEAK SUMMARY:
==2129== definitely lost: 0 bytes in 0 blocks
==2129== indirectly lost: 0 bytes in 0 blocks
==2129== possibly lost: 0 bytes in 0 blocks
==2129== still reachable: 12,520 bytes in 1 blocks
==2129== suppressed: 0 bytes in 0 blocks
==2129==
==2129== ERROR SUMMARY: 450 errors from 46 contexts (suppressed: 19 from 8)

7 Second memory test

==2809== HEAP SUMMARY:
==2809== in use at exit: 12,520 bytes in 1 blocks
==2809== total heap usage: 1,051 allocs, 1,050 frees, 3,115,023 bytes allocated
==2809==
==2809== LEAK SUMMARY:
==2809== definitely lost: 0 bytes in 0 blocks
==2809== indirectly lost: 0 bytes in 0 blocks
==2809== possibly lost: 0 bytes in 0 blocks
==2809== still reachable: 12,520 bytes in 1 blocks

==2809== suppressed: 0 bytes in 0 blocks
==2809== Rerun with –leak-check=full to see details of leaked memory
==2809==
==2809== ERROR SUMMARY: 2400 errors from 38 contexts (suppressed: 19 from 8)

Profiling

8 First profiling test

Flat profile:

time	*culmutative*	*seconds*	*selfseconds*	*calls*	*self*(*ms*/*call*)	*total*(*ms*/*call*)	*name*
64.11	6.09		6.09	36847	0.17	0.17	*coreMul*(...
35.89	9.50		3.41	36867	0.00	0.09	*coreAdd*(...)
0.00	9.50		0.00	80	0.00	0.00	*coreSub*(...)
0.00	9.50		0.00	20	0.00	0.09	*coreDiv*(...)
0.00	9.50		0.00	10	0.00	0.09	*mod*(...)

9 Second profiling test

Flat profile:

time	*culmutative*	*seconds*	*selfseconds*	*calls*	*self*(*ms*/*call*)	*total*(*ms*/*call*)	*name*
64.62	31.14		31.14	184480	0.17	0.17	*coreMul*(..
35.29	48.15		17.00	184580	0.09	0.09	*coreAdd*(...
0.05	48.17		0.03	400	0.06	0.06	*coreSub*(...
0.00	48.19		0.00	100	0.00	0.22	*coreDiv*(...
0.00	48.19		0.00	50	0.00	0.22	*mod*(...)

time&: program used by this function.
culmutative second:a running sum of the number of seconds accounted for by this function and those listed above it.
selfsecond:the number of seconds accounted for by this function alone. This is the major sort for this listing.
call: the number of times this function was invoked, if this function is profiled, else blank.
self ms/call: the average number of milliseconds spent in this function per call, if this function is profiled, else blank.
total: the average number of milliseconds spent in this function and its descendents

per call, if this function is profiled, else blank.
name: the name of the function. This is the minor sort for this listing. The index shows the location of the function in the gprof listing.
If the index is in parenthesis it shows
where it would appear in the gprof listing if it were to be printed.