Master Plan – C++ Project

Our goal

Build a spreadsheet with different kind of simulation of loans.

Options of our project

1. Build a spreadsheet with fixed rate and installment

(3 out 4 entries)

2. Build a spreadsheet whith fixed principal payments, and

- fixed interest rate

- margin + parametrised, normally distributed floating interest rate

Blocks

1. main block

open the exit file

call the menu block procedure

close the exit file

2. menu block procedure

ask the user which option he/she wants until option 3

if option is 1. Build a spreadsheet with fixed rate and installment

then call data entry procedure to fixe rate and installment

if option is 2. margin

then call data entry procedure to margin, normally distributed floating interest rate

if option 3 exit the program

3. fixedrateinstall\_dataentry procedure – started when option 1 of menu block is selected

loop asking the option among 3 out 4 variables which desire until option 5

cout>> “ 1. PMT PV r” , call secant procedure

cout>> “ 2. PMT r n” , call secant procedure

cout>> “ 3. PV r n” , call secant procedure

cout>> “ 4. PMT PV n”, call secant procedure

n= number of periods- you have asking in which base – annual, quarterly, semi-annual

r= interest rate

cout>>”5. Back to the menu block Procedure”

4. margin\_dataentry procedure

cout>>” what is the present value? “

cout>>”what is the margin?”

cout>>”how many periods?” yearly,quarterly,semi-annual, monthly

starting date is always the current month and year

call the margin\_bench procedure - box\_muller\_random

with the return of margin procedure print in csv file the results

double box\_muller\_random(double mu, double sigma)

{

static const double epsilon = std::numeric\_limits<double>::min();

static const double two\_pi = 2.0\*3.14159265358979323846;

thread\_local double z1;

thread\_local bool generate;

generate = !generate;

if (!generate)

return z1 \* sigma + mu;

double u1, u2;

do

{

u1 = rand() \* (1.0 / RAND\_MAX);

u2 = rand() \* (1.0 / RAND\_MAX);

}

while ( u1 <= epsilon );

double z0;

z0 = sqrt(-2.0 \* log(u1)) \* cos(two\_pi \* u2);

z1 = sqrt(-2.0 \* log(u1)) \* sin(two\_pi \* u2);

return z0 \* sigma + mu;

}

float secant(int choice,float input1, float input2, float input3)

{

float n = 0, f,xm, x0, c,x,subtrat;

float x1 = 0, x2 = 1, E = 0.0001;

.. plug in the options which does not depend to calcule secant, just print out in the file

if (f(choice,input1,input2,input3,x1) \* f(choice,input1,input2.input3,x2) < 0) {

do {

// calculate the intermediate value

subtrat = f(choice,input1,input2,input3,x2) - f(choice,input1,input2.input3,x1)

x0 = (x1 \* f(choice,input1,input2,input3,x2) - x2 \* f(choice,input1,input2,input3,x1)) / (subtrat);

// check if x0 is root of equation or not

c = f(choice,input1,input2,input3,x1) \* f(choice,input1,input2,input3,x0);

// update the value of interval

x1 = x2;

x2 = x0;

// update number of iteration

n++;

// if x0 is the root of equation then break the loop

if (c == 0)

break;

xm = (x1 \* f(x2) - x2 \* f(x1)) / (f(x2) - f(x1));

} while (fabs(xm - x0) >= E); // repeat the loop

// until the convergence

secant = x0;

} else

cout << "Can not find a root in the given inteval";

}

call print the file