Download the following zipped file. It contains code for a sample program with a number of simple functions.

[[<https://markbowman.org/231/Lab05.zip>](https://www.msu.edu/~bowmanm/230/Lab12.winzip)](https://www.msu.edu/~bowmanm/230/Lab11.winzip)

Throughout this lab you will add counters to the function to approximate the number of lines of code that are executed. This isn’t the exact count of CPU operations, but will give an idea of how complex the functions are.

1. Run the program to verify that it works. You will be prompted to enter an integer N, then the linear() function will be called.

Enter N: ***10***

0 1

1 2.718

2 7.38752

3 20.0793

4 54.5755

5 148.336

6 403.178

7 1095.84

8 2978.49

9 8095.52

Linear count = 0

2. Add the following lines to count the number of lines of code that are executed inside the loop.

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* linear()

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

int linear(int n)

int counter;

{ int i;

float p;

counter = 0;

for(i=0;i<n;i++)

{ cout << i << " ";

p = pow(2.718F,i);

counter += 3;

cout << p << endl;

};

Change this line to return the value of counter.

return 0;

}

***Code Snippet – Lab05.cpp:***

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* linear()

\* Safford, Twymun - 08-Sep-2021

\*

\* Added code additions for exection

\* inside linear loop

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

int linear(int n)

{

int i;

//counter - defined outside of for loop

int counter;

float p;

//initialize counter to zero

//cout << "Number of Lines | Values" << endl; //debugging

//cout << "---------------------------" << endl; //debugging

counter = 0;

for (i = 0; i < n; i++)

//n - user defined

//i will increment until it is

//no longer less than n

{

//cout i and p

cout << i << " ";

p = pow(2.718F, i);

//increment counter by 3

cout << p << endl;

counter += 3;

};

return counter;

}

3. Run the program again, and write down the counts for the following values of N:

|  |  |  |  |
| --- | --- | --- | --- |
| N | 10 | 100 | 1000 |
| Count | 30 | 300 | 3000 |

***Sample runs:***

*N=10*

Graphical user interface, text, website

Description automatically generated

*N = 100*

*Shape, rectangle

Description automatically generated*

*N=1000*

*Text

Description automatically generated*

4. For each run, compare the results by dividing the Count by N. What is the ratio?

***The ratio for each of these runs is 3. This reinforces that the function is linear.***

This ratio should be the same for each run, telling us that the function is linear. If the loop had more calculation and output lines, we could increase the counter increment and get the same results.

5. Comment out the section in main() for steps 1-4. Uncomment the section in main() for steps 5-8. Run the program to make sure the logarithmic function doubles up to your input N value.

***Sample Output:***

Text

Description automatically generated

6. Like you did in the previous function, add an integer counter to add up the total number of lines executed while the loop runs. Have your function return this value to the main() function.

***Code Snippet (Logarithmic) – Lab05.cpp:***

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* logarithmic()

\* Safford, Twymun - 08-Sep-2021

\*

\* Added code additions for exection

\* inside loop for logarithmic

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

int logarithmic(int n)

{

int sum;

//implement counter

//int counter;

//set counter equal to zero

int counter = 0;

sum = 1;

while (sum < n)

{

//how does the sum scale logarithmicly?

sum = sum \* 2;

//counter need to increment by intervals of 3

counter += 3;

cout << sum << endl;

};

//need to return counter

return counter;

}

7. Run the program again, and write down the counts for the following values of N:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| N | 10 | 100 | 1000 | 10,000 |
| Log2N | 3.322 | 6.644 | 9.966 | 13.287 |
| Count | 12 | 21 | 30 | 42 |
| Count/ Log2N | 3.612 | 3.161 | 3.010 | 3.161 |

***Sample runs:***

*N = 100:*

***Text

Description automatically generated***

*N = 10000:*

Text

Description automatically generated

8. For each run, compare the results by dividing the Count by Log2N. What is the ratio?

* 12/3.322 = **3.612**
* 21/6.644= **3.161**
* 30/9.666 = **3.010**
* 42/13.288 = **3.161**

*The ratio appears to be 3.612, 3.161, 3.010, and 3.161 respectively for the values of N (number of lines to run). This in part based on the enclosed chart when the novel idea crossed my mind to visualize different functions on a graph that this approximates (aside from being in the table) Log2N.*

*Chart

Description automatically generated*

9. Comment out the section in main() for steps 5-8. Uncomment the section in main() for steps 9-12. Run the program to make sure the quadratic function prompts for a value for M, then displays a rectangle of #’s.

*Sample Output:*

Text

Description automatically generated

10. Use the variables inner and outer to add counters to add up the total lines of code executed by the inner loop and the outer loop. Have the function display the value of the outer counter, and return the value of the inner counter. Examples:

|  |  |  |
| --- | --- | --- |
| Enter N: ***10***  Enter M: ***5***  ##########  ##########  ##########  ##########  ##########  Outer count = 5  Quadratic count = 50 |  | Enter N: ***20***  Enter M: ***10***  ####################  ####################  ####################  ####################  ####################  ####################  ####################  ####################  ####################  ####################  Outer count = 10  Quadratic count = 200 |

***Code Snippet (Quadratic) – Lab05.cpp:***

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* quadratic()

\* Safford, Twymun - 08-Sep-2021

\*

\* Added code additions for exection

\* inside linear loop

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

//approximate number of lines of code

//for inner and outer loops

/// n - number of operations

int quadratic(int n)

{

//i - outer loop for condition

//j - inner loop for condition

//m - number of rows for #'s

int i, j, m;

//inner - inner operations counter

//outer - outer operations counter

int inner, outer;

// Initialize

inner = outer = 0;

// Get m

cout << "Enter M: ";

cin >> m;

// Outer loop

for (i = 0; i < m; i++)

{

// Inner loop

for (j = 0; j < n; j++)

{

cout << '#';

//counter - increment every time inner loop runs

inner++;

};

//increment every time condition for outer loop completes

outer++;

//endl for each row of #'s

cout << endl;

};

//cout statement - convey number of outer operations

cout << "The outer count is = " << outer << endl;

return inner;

}

**Example Output:**

**A screen shot of a computer

Description automatically generated with low confidence**

11. Run the program again, using the same value for M as N and write down the total counts for the following values of N:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| N | 10 | 20 | 100 | 200 |
| Count | 100 | 400 | 10000 | 40000 |

12. How do the counts relate to the values of N?

***The counts are equal to N2 – which is to be expected for this quadratic function.***

Lab05.cpp – Final code

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* CPSC 231 MW - Lab 05

\* Created by Safford, Twymun

\* Date: 08-Sep-2021

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#include <iostream>

using namespace std;

//for the counter

int linear(int n);

int logarithmic(int n);

int quadratic(int n);

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* main()

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void main()

{

int a, n;

// Get N

cout << "Enter N: ";

cin >> n;

// Steps 1-4

//a = linear(n);

//cout << "Linear count = " << a << endl;

// Steps 5-8

//logarithmic count

//a = logarithmic(n);

//cout << "Logarithmic count = " << a << endl;

// Steps 9-12

//quadratic count

a = quadratic(n);

cout << "Quadratic count = " << a << endl;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* linear()

\* Safford, Twymun - 08-Sep-2021

\*

\* Added code additions for exection

\* inside linear loop

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

int linear(int n)

{

int i;

//counter - defined outside of for loop

int counter;

float p;

//initialize counter to zero

//cout << "Number of Lines | Values" << endl; //debugging

//cout << "---------------------------" << endl; //debugging

counter = 0;

for (i = 0; i < n; i++)

//n - user defined

//i will increment until it is

//no longer less than n

{

//cout i and p

cout << i << " ";

p = pow(2.718F, i);

//increment counter by 3

cout << p << endl;

counter += 3;

};

return counter;

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* logarithmic()

\* Safford, Twymun - 08-Sep-2021

\*

\* Added code additions for exection

\* inside loop for logarithmic

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

int logarithmic(int n)

{

int sum;

//implement oounter

int counter;

//set counter equal to zero

//sum is equal to 1

counter = 0;

sum = 1;

//integers for i

//int i;

//i = 0;

while (sum < n)

{

//sum should scale logarithmically

//cout << i << " ";

sum = sum \* 2;

//i++;

//counter - increment by 3

cout << sum << endl;

counter += 3;

};

return counter;

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* quadratic()

\* Safford, Twymun - 08-Sep-2021

\*

\* Added code additions for exection

\* inside linear loop

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

//approximate number of lines of code

//for inner and outer loops

/// n - number of operations

int quadratic(int n)

{

//i - outer loop for condition

//j - inner loop for condition

//m - number of rows for #'s

int i, j, m;

//inner - inner operations counter

//outer - outer operations counter

int inner, outer;

// Initialize

inner = 0;

outer = 0;

// Get m

cout << "Enter M: ";

cin >> m;

// Outer loop

for (i = 0; i < m; i++)

{

// Inner loop

for (j = 0; j < n; j++)

{

cout << '#';

//counter - increment every time inner loop runs

inner++;

};

//increment every time condition for outer loop completes

outer++;

//endl for each row of #'s

cout << endl;

};

//cout statement - convey number of outer operations

cout << "The outer count is = " << outer << endl;

return inner;

}