**Министерство науки и высшего образования Российской Федерации Федеральное государственное бюджетное образовательное учреждение высшего образования** **«Новгородский государственный университет имени Ярослава Мудрого»**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Кафедра ИТИС

«**Организация рекурсивных вычислений**»

Лабораторная работа №4

По дисциплине: Основы программирования: Алгоритмические языки и программирование

Отчет

Выполнил: студент гр. 2091

\_\_\_\_\_\_\_\_\_\_\_Бутовский М.С.

Проверил: Преподаватель

\_\_\_\_\_\_\_\_\_\_\_ Кулаков И. Ю.

Великий Новгород

2023

**1.Постановка задачи (внешние спецификации)**

1. Наименование задачи:

Решить две задачи:

1. Разработать программу, которая находит все трехзначные числа, представимые в виде сумм факториалов своих цифр. Использовать рекурсивную функцию вычисления n!.

2. Разработать программу, которая по введённым вещественным данным X и E, вычисляет через две функции формулу, сравнивает с формулой из оригинальной библиотеки и выводит время, затраченное на проделывание функции.

1. Словесное описание:

Ввод:

Xнач - Некая переменная для вычисления значений функции

Хкон - Некая переменная для вычисления значений функции

hx - Некая переменная для вычисления значений функции

Yнач - Некая переменная для вычисления значений функции

Yкон - Некая переменная для вычисления значений функции

hy - Некая переменная для вычисления значений функции

Вывод:

X – Переменная для вычисления f(X,Y)

Y – Переменная для вычисления f(X,Y)

Z – Функция f(X,Y)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| *Объект программы* | *Имя этого объекта в программе* | *Тип данных* | *Диапазоны представления* | *Простая/структура* | *Вход/выход/константа* |
| x | x | double | От 2.22507e-308 до 1.79769e+308 | Простая | Вход |
| e | e | double | От 2.22507e-308 до 1.79769e+308 | Простая | Вход |
| Логарифм (обычный) | ln\_std | double | От 2.22507e-308 до 1.79769e+308 | Простая | Вход |
| Логарифм (рекурсия) | ln\_rec | double | От 2.22507e-308 до 1.79769e+308 | Простая | Вход |
|  | n | int | От -2147483648 до 2147483647. | Простая | Вход |
| Логарифм(итеррационный) | ln\_iter | double | От 2.22507e-308 до 1.79769e+308 | Простая | Вход |
| Длительность рекурсии | duration\_rec.count() | float | От 1.17549e-38 до 3.40282e+38 | Простая | Выход |
| Длительность итеррационного | duration\_iter.count() | float | От 1.17549e-38 до 3.40282e+38 | Простая | Выход |

1. Внешние спецификации функции программы:

Получить на вход с помощью меню данные о выборе пользователя, если пользователь выбрал 1 задание, то выводим данные о расчетах и условии задания в окна, если он выбрал второе, то получаем на вход вещественные данные с проверкой корректности ввода, рассчитываем по формулам данные и также выводим в отдельные окна с помощью библиотеки curses.

1. Математическая формулировка задачи:

Сложение, умножение, деление, возведение в степень, рассчитывать квадратный корень и сравниваем данные:

pow((x - 1), n) / (n \* pow(x, n))

1. Спецификации интерфейса:

Вывод на консоль с использованием библиотеки curses:

Создаются два окна для ввода, меню для выбора, для вывода (input, output и menu), в первом программа получает входные данные и при ошибке пользователя просит ввести ещё раз, во втором окне предоставляются все выходные данные, в третьем пользователь с помощью кнопок выбирает в меню функционал.

Я выбрал метод решения через функции т.к. это более лаконично и эффективно, чем писать всё в функции main(), также я выбрал реализацию ввода/вывода через псевдографику т.к. это выглядит красивее и пользователь воспринимает информацию лучше.

1. Внешние данные тестирования:

*Ошибки при вводе данных:*

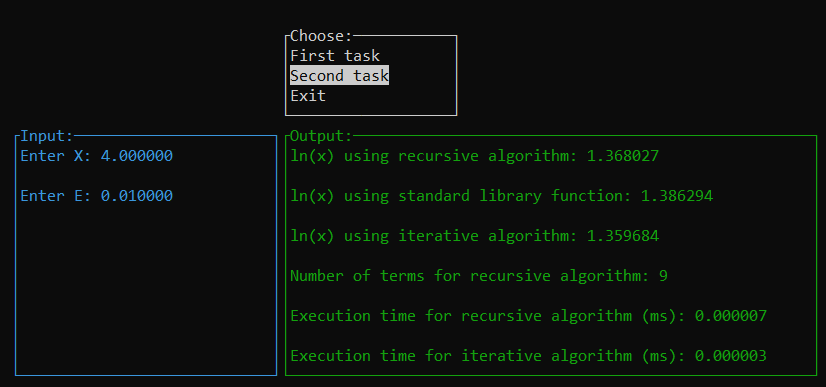
При вводе нечисловых данных, на экран будет выведено соответствующее сообщение и будет запрошен повторный ввод данных.

*Ошибки, связанные с вычислениями:*

Таковых не выявлено.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *Номер теста* | *Назначение теста* | *Значения исходных данных* | *Ожидаемый результат* | *Реакция программы* |
| 1 | Проверить программу на правильность выполнения | Menu = “First task” | Input = “task”  Output = 145 | Input = “task”  Output = 145 |
| 2 | Проверить программу на правильность выполнения | Menu = “Second task”  X = 4  E = 0.001 | ln(x) using recursive algorithm: 1.368027  ln(x) using standard library function: 1.386294  ln(x) using iterative algorithm: 1.359684  Number of terms for recursive algorithm: 9  Execution time for recursive algorithm (ms): 0.000004  Execution time for iterative algorithm (ms): 0.000002 | ln(x) using recursive algorithm: 1.368027  ln(x) using standard library function: 1.386294  ln(x) using iterative algorithm: 1.359684  Number of terms for recursive algorithm: 9  Execution time for recursive algorithm (ms): 0.000004  Execution time for iterative algorithm (ms): 0.000002 |
| 3 | Проверить программу на правильность выполнения | Menu = “Second task”  X = E | Invalid X, try again: | Invalid X, try again: |
| 4 | Проверить программу на правильность выполнения | Menu = “Second task”  X = 4  E = H | Invalid E, try again: | Invalid E, try again: |
| 5 | Проверить программу на правильность выполнения | Menu = “Exit” | Have a good day! | Have a good day! |

1. Пример работающей программы:



**2. Проектирование задачи**

1.Наименование программы

1\_try.sln

2.Уточнённое словесное описание

Программа создает два окна input, output и menu, на вход получает данные внутри программы выполняются функции:

1 задание: вычисление факториала числа, определение чисел, которые соответствуют условиям, (все трехзначные числа, представимые в виде сумм факториалов своих цифр), выходных данных.

2 задание: программа получает на вход два числа (x и e), ввод данных проверяется на верность и в противном случае выводится ошибка, запрашивается повторный ввод данных, после получения данных выполняются итерационный, рекурсивный и классический способ вычисления логарифма, также вычисляется время, затраченное на выполнения итерационного и рекурсивного алгоритмов. После данные о результате вычислений логарифма различными способами и время, затраченное на них выводится в окна.

3.Выбор метода решения поставленной задачи

Такой метод решения задачи т.к. этот метод единственно возможный в моём случае (использование формул для вычисления фиксированной формулы логарифма).

4.Уточнённые глобальные данные программы и пользовательские типы

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| *Объект программы* | *Имя этого объекта в программе* | *Характеристики* | | | *Как используется в программе* |
| *Тип данных* | *Диапазоны представления* | *Прост/структура/константа* |
| term | term | double | От 2.22507e-308 до 1.79769e+308 | простая | Переменная для хранения рекурсивной формулы |
| result | result | double | От 2.22507e-308 до 1.79769e+308 | простая | Переменная для хранения значения логарифма |
| Временная переменная окна | win | WINDOW\* | - | Структура | Используется для корректного отображения сообщений об ошибках и вводе |
| Окно для ввода | inputwin | WINDOW\* | - | Структура | Используется для корректного отображения сообщений об ошибках и вводе |
| Окно для вывода | outputwin | WINDOW\* | - | Структура | Используется для корректного отображения сообщений об ошибках и вводе |
| Окно для меню | menuwin | WINDOW\* | - | Структура | Используется для корректного отображения сообщений об ошибках и вводе |
| x | x | double | От 2.22507e-308 до 1.79769e+308 | простая | Переменная для вычисления выходных данных |
| e | e | double | От 2.22507e-308 до 1.79769e+308 | простая | Переменная для вычисления выходных данных |
| ln\_std | ln\_std | double | От 2.22507e-308 до 1.79769e+308 | простая | Переменная для вычисления стандартного логарифма |
| start\_rec | start\_rec | float | От 1.17549e-38 до 3.40282e+38 | простая | Переменная для начала вычисления времени |
| ln\_rec | ln\_rec | double | От 2.22507e-308 до 1.79769e+308 | простая | Переменная для вычисления рекурсивного логарифма |
| end\_rec | end\_rec | float | От 1.17549e-38 до 3.40282e+38 | простая | Переменная для конца вычисления времени |
| start\_iter | start\_iter | float | От 1.17549e-38 до 3.40282e+38 | простая | Переменная для начала вычисления времени |
| ln\_iter | ln\_iter | double | От 2.22507e-308 до 1.79769e+308 | простая | Переменная для вычисления стандартного логарифма |
| duration\_iter | duration\_iter | float | От 1.17549e-38 до 3.40282e+38 | простая | Переменная для конечного вычисления времени |

5.Декомпозиция функций

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Назначение | Имя | Параметры | | Процедура/функция | Тестируется/не тестируется |
| Вход: тип | Выход: тип |
| Функция для вычисления факториала | factorial | int | int | Функция | Тестируется |
| Функция для проверки корректности ввода пользователя | check | float | float | Функция | Тестируется |
| Найти цифры, подходящие под условия | findNumbers | - | - | процедура | Тестируется |
| Функция для вычисления рекурсивного логарифма | ln\_recursive | double | double | функция | Тестируется |
| Функция для вычисления итерационного логарифма | ln\_iterative | double | double | функция | Тестируется |

6.Алгоритмизация

#include <curses.h>

#include <iostream>

#include <iomanip>

#include <cmath>

#include <ctime>

#include <chrono>

#include <string>

using namespace std;

using namespace std::chrono;

double check(WINDOW\* win, int x, int y, const char\* prompt)

{

double num;

bool is\_valid = false;

while (!is\_valid)

{

char\* input\_str = new char[256];

echo();

wgetstr(win, input\_str);

noecho();

try

{

num = stof(input\_str);

if (num > 0) // Добавляем проверку на значение больше 0.5

{

is\_valid = true;

}

else

{

throw invalid\_argument(""); // Если значение меньше или равно 0.5, выбрасываем исключение

}

}

catch (const invalid\_argument&) // Исправляем тип исключения на const invalid\_argument&

{

mvwprintw(win, x, y, " ");

mvwprintw(win, x, y, prompt);

wrefresh(win);

}

}

return num;

}

double checkit(WINDOW\* win, int x, int y, const char\* prompt)

{

double num;

bool is\_valid = false;

while (!is\_valid)

{

char\* input\_str = new char[256];

echo();

wgetstr(win, input\_str);

noecho();

try

{

num = stof(input\_str);

if (num > 0.5) // Добавляем проверку на значение больше 0.5

{

is\_valid = true;

}

else

{

throw invalid\_argument(""); //Если значение меньше или равно 0.5, выбрасываем исключение

}

}

catch (const invalid\_argument&) //Исправляем тип исключения на const invalid\_argument&

{

mvwprintw(win, x, y, " ");

mvwprintw(win, x, y, prompt);

wrefresh(win);

}

}

return num;

}

int factorial(int n)//рекурсивно находим факториал

{

if (n == 0 || n == 1)

return 1;

return n \* factorial(n - 1);

}

void findNumbers(WINDOW\* win)//находим все числа, у которых сумма факториалов их чисел равна этому числу

{

for (int i = 100; i <= 999; i++)

{

int k = 1;

int number = i;

int sum = 0;

while (number > 0)

{

sum += factorial(number % 10);

number /= 10;

}

if (sum == i)

{

mvwprintw(win, k, 1, "%d", i);

k++;

}

}

wrefresh(win);

}

double ln\_recursive(double x, double e, int n = 1, double result = 0.0)//рекурсивно находим с точностью E

{

double term = pow((x - 1), n) / (n \* pow(x, n));//рекурсивная формула

result += term;//итоговое значение рекурсивной

if (fabs(term) < e)//если точность, какая нужна, то возвращаем итоговое значение

{

return result;

}

else

{

return ln\_recursive(x, e, n + 1, result);//если точность не та, то продолжаем цикл

}

}

double ln\_iterative(double x, double e)//итеррационно находим с точностью E

{

double result = 0;//ответ

double term = (x - 1) / x;//перввый прогон формулы

int n = 1;//для степеней

while (fabs(term) >= e)//пока по модулю точность не будет близка к E

{

result += term;//добавляем в конечный ответ точность

n++;//следующий проход

term = pow((x - 1) / x, n) / n;//формула

}

return result;

}

int main()

{

initscr();

//начало curses

noecho();

//не отоброжаем ввод пользователя

WINDOW\* menuwin = newwin(5, 20, 7, 40);

//создаем окно для меню

box(menuwin, 0, 0);

//выводим рамки окон

mvwprintw(menuwin, 0, 1, "Choose:");

//заголовок окон

start\_color();

//делаем окна цветными

/\*

COLOR\_BLACK

COLOR\_RED

COLOR\_GREEN

COLOR\_YELLOW

COLOR\_BLUE

COLOR\_MAGENTA

COLOR\_CYAN

COLOR\_WHITE

\*/

init\_pair(1, COLOR\_CYAN, COLOR\_BLACK);

init\_pair(2, COLOR\_GREEN, COLOR\_BLACK);

//создаем окно для ввода данных

WINDOW\* inputwin = newwin(13, 30, 12, 10);

//создаем окно для вывода данных

WINDOW\* outputwin = newwin(13, 60, 12, 40);

box(inputwin, 0, 0);

box(outputwin, 0, 0);

mvwprintw(inputwin, 0, 1, "Input:");

mvwprintw(outputwin, 0, 1, "Output:");

wbkgd(inputwin, COLOR\_PAIR(1));

wbkgd(outputwin, COLOR\_PAIR(2));

//делаем данные видимыми

wrefresh(inputwin);

wrefresh(outputwin);

wrefresh(menuwin);

//начало менюшки

keypad(menuwin, true);

//управление стрелочками

curs\_set(0);

//убираем курсор

string choices[3] = { "First task", "Second task", "Exit" };

//выбор в менюшке

int choice;

int highlight = 0;

bool work = 1;

//переменные для работы с меню

while (work)

{

for (int i = 0; i < 3; i++)

{

if (i == highlight)

wattron(menuwin, A\_REVERSE);

mvwprintw(menuwin, i + 1, 1, choices[i].c\_str());

wattroff(menuwin, A\_REVERSE);

}

choice = wgetch(menuwin);

switch (choice) {

case KEY\_UP:

highlight--;

if (highlight == -1)

highlight = 0;

break;

case KEY\_DOWN:

highlight++;

if (highlight == 3)

highlight = 2;

break;

default:

break;

}

if (choice == 10 && highlight == 0)

{

//очищаем окно от прошлых данных

wclear(outputwin);

box(outputwin, 0, 0);

mvwprintw(outputwin, 0, 1, "Output:");

wclear(inputwin);

box(inputwin, 0, 0);

mvwprintw(inputwin, 0, 1, "Input:");

//выводим условия в окно ввода

mvwprintw(inputwin, 1, 1, "Find all three-digit numbers");

mvwprintw(inputwin, 2, 1, "representable as sums of ");

mvwprintw(inputwin, 3, 1, "factorials of their digits.");

mvwprintw(inputwin, 4, 1, "Use a recursive function to");

mvwprintw(inputwin, 5, 1, "calculate n!");

wrefresh(inputwin);

wrefresh(outputwin);

//выводим результат

findNumbers(outputwin);

}

if (choice == 10 && highlight == 1)

{

wclear(outputwin);

box(outputwin, 0, 0);

mvwprintw(outputwin, 0, 1, "Output:");

wclear(inputwin);

box(inputwin, 0, 0);

mvwprintw(inputwin, 0, 1, "Input:");

wrefresh(inputwin);

wrefresh(outputwin);

echo();

curs\_set(1);

//вводим данные

double x, e;

mvwprintw(inputwin, 1, 1, "Enter X: ");

wrefresh(inputwin);

x = checkit(inputwin, 1, 1, "Invalid X, try again: ");

wclear(inputwin);

box(inputwin, 0, 0);

mvwprintw(inputwin, 0, 1, "Input:");

mvwprintw(inputwin, 1, 1, "Enter X: %f", x);

wrefresh(inputwin);

mvwprintw(inputwin, 3, 1, "Enter E: ");

wrefresh(inputwin);

e = check(inputwin, 3, 1, "Invalid E, try again: ");

wclear(inputwin);

box(inputwin, 0, 0);

mvwprintw(inputwin, 0, 1, "Input:");

mvwprintw(inputwin, 1, 1, "Enter X: %f", x);

mvwprintw(inputwin, 3, 1, "Enter E: %f", e);

wrefresh(inputwin);

//вычислям стандартным способом

double ln\_std = log(x);

//вычисляем кол-во сложений в рекурсивном алгоритме

int n = 1;

double term = pow((x - 1), n) / (n \* pow(x, n));

while (fabs(term) >= e)

{

n++;

term = pow((x - 1), n) / (n \* pow(x, n));

}

//начинаем вычислять время, затраченное на выполнение рекурсивного алгоритма

auto start\_rec = high\_resolution\_clock::now();

//вычисляем с помощью рекурсивного алгоритма

double ln\_rec = ln\_recursive(x, e);

//заканчиваем вычислять время

auto end\_rec = high\_resolution\_clock::now();

//вычислям сколько затратили

duration<float> duration\_rec = end\_rec - start\_rec;

//начинаем вычислять время, затраченное на выполнение иттерационного алгоритма

auto start\_iter = high\_resolution\_clock::now();

//вычисляем с помощью иттерационного алгоритма

double ln\_iter = ln\_iterative(x, e);

//заканчиваем вычислять время

auto end\_iter = high\_resolution\_clock::now();

//вычислям сколько затратили

duration<float> duration\_iter = end\_iter - start\_iter;

//выводим все значения

mvwprintw(outputwin, 1, 1, "ln(x) using recursive algorithm: %f", ln\_rec);

mvwprintw(outputwin, 3, 1, "ln(x) using standard library function: %f", ln\_std);

mvwprintw(outputwin, 7, 1, "Number of terms for recursive algorithm: %d", n);

mvwprintw(outputwin, 5, 1, "ln(x) using iterative algorithm: %f", ln\_iter);

mvwprintw(outputwin, 9, 1, "Execution time for recursive algorithm (ms): %f", duration\_rec.count());

mvwprintw(outputwin, 11, 1, "Execution time for iterative algorithm (ms): %f", duration\_iter.count());

wrefresh(outputwin);

curs\_set(0);

}

if (choice == 10 && highlight == 2)

{

wclear(outputwin);

box(outputwin, 0, 0);

mvwprintw(outputwin, 0, 1, "Output:");

mvwprintw(outputwin, 1, 1, "Have a good day!");

wrefresh(outputwin);

work = 0;

break;

}

}

wgetch(menuwin);

endwin();

return 0;

}

**Вывод**

Проведя все описания и тесты вышестоящих функций я пришёл к выводу, что программа, которую я написал, работает корректно и соответствует всем вышеописанным мной критериями.