Министерство образования республики МолдоваТехнический университет Молдовы

Факультет вычислительной техники, информатики и микроэлектроники

Отчёт

Лабораторная работа № 1

Тема: Циклические вычислительные процессы

выполнил студент гр. CR-254 Романюк Я.

проверил: Фалько Н.

Кишинёв 2025

[Цель работы 2](#_Toc41356)

[Задание 2](#_Toc41357)

[Реализация выполнения задания и равнение с chatgpt 3](#_Toc41359)

[Вывод 8](#_Toc41360)

[Список литературы 9](#_Toc41361)

Цель работы

Использование операторов условия и операторов цикла на языке Си. Работа с гит.

Задание  
**Вариант 20**

1.Вычислить и вывести на экран в виде таблицы значения функции F на интервале от Хнач. до Хкон. с шагом dХ.

*-  при x < 0 и b ≠ 0*

# *F = при х > 0 и b = 0*

*-  +  в остальных случаях*

где a, b, с - действительные числа.

|  |  |
| --- | --- |
| Мой код | Код chatgpt |
| #include <iostream>  #include <cmath>  #include <iomanip>  using namespace std;  const double EPS = 1e-12;  int main() {  double a, b, c;  double StartX, EndX, dX;  double F;  cout << "Введите:" << endl;  cout << "\ta = "; cin >> a;  cout << "\tb = "; cin >> b;  cout << "\tc = "; cin >> c;  cout << "\tX нач. = "; cin >> StartX;  cout << "\tX кон. = "; cin >> EndX;  cout << "\tdX = "; cin >> dX;  if (dX <= 0) {  cerr << "Ошибка: шаг dX должен быть положительным." << endl;  return 1;  }  if (StartX > EndX) {  cerr << "Ошибка: X нач. должен быть меньше или равен X кон." << endl;  return 1;  }  cout << fixed << setprecision(5);  for (double x = StartX; x <= EndX + EPS; x += dX) {  if (x < 0 && fabs(b) > EPS) {  if (fabs(c \* x - a) < EPS) {  cout << "x = " << x << "\tОшибка: деление на 0" << endl;  continue;  }  F = -((2 \* x - c) / (c \* x - a));  }  else if (x > 0 && fabs(b) < EPS) {  if (fabs(x - c) < EPS) {  cout << "x = " << x << "\tОшибка: деление на 0" << endl;  continue;  }  F = (x - a) / (x - c);  }  else {  if (fabs(c) < EPS || fabs(2 \* x) < EPS) {  cout << "x = " << x << "\tОшибка: деление на 0" << endl;  continue;  }  F = -(x / c) - (c / (2 \* x));  }  cout << "x = " << x << "\tF = " << F << endl;  }  return 0;  }   Пример вывода: | #include <iostream>  #include <cmath>  #include <iomanip>  #include <optional>  #include <string>  using namespace std;  const double EPS = 1e-12; // Допуск для сравнения с нулем  // Структура для результата функции  struct FResult {  optional<double> value; // Значение функции, если успешно  string error; // Сообщение об ошибке, если деление на ноль  };  /\*\*  \* @brief Вычисляет значение функции F для заданного x и коэффициентов a, b, c  \*  \* @param x Значение переменной x  \* @param a Коэффициент a  \* @param b Коэффициент b  \* @param c Коэффициент c  \* @return FResult — значение функции и сообщение об ошибке  \*/  FResult computeF(double x, double a, double b, double c) {  FResult result;  if (x < 0 && fabs(b) > EPS) { // b "не ноль"  if (fabs(c \* x - a) < EPS) {  result.error = "Деление на 0: c\*x - a = 0";  return result;  }  result.value = -((2 \* x - c) / (c \* x - a));  }  else if (x > 0 && fabs(b) < EPS) { // b "равно нулю"  if (fabs(x - c) < EPS) {  result.error = "Деление на 0: x - c = 0";  return result;  }  result.value = (x - a) / (x - c);  }  else { // Остальные случаи  if (fabs(c) < EPS) {  result.error = "Деление на 0: c = 0";  return result;  }  if (fabs(2 \* x) < EPS) {  result.error = "Деление на 0: 2\*x = 0";  return result;  }  result.value = -(x / c) + (-c / (2 \* x));  }  return result;  }  int main() {  double a, b, c;  double startX, endX, dX;  // Ввод коэффициентов и диапазона X  cout << "Введите коэффициенты и параметры диапазона X:" << endl;  cout << "\ta = "; cin >> a;  cout << "\tb = "; cin >> b;  cout << "\tc = "; cin >> c;  cout << "\tX нач. = "; cin >> startX;  cout << "\tX кон. = "; cin >> endX;  cout << "\tdX = "; cin >> dX;  cout << fixed << setprecision(5);  // Основной цикл  for (double x = startX; x <= endX + EPS; x += dX) {  FResult result = computeF(x, a, b, c);  if (result.value.has\_value()) {  cout << "x = " << x << "\tF = " << result.value.value() << endl;  } else {  cout << "x = " << x << "\tОшибка: " << result.error << endl;  }  }  return 0;  }  Пример вывода: |

Сравнение

**Основные отличия**

Код chatgpt**:**

**Структура FResult и функция computeF**

Есть отдельная функция:

struct FResult {

optional<double> value;

string error;

};

FResult computeF(double x, double a, double b, double c);

* Функция **инкапсулирует логику вычисления** F для любого x.
* Она возвращает либо значение через optional<double>, либо сообщение об ошибке, если произошло деление на ноль.
* Это делает код **модульным**, легко тестируемым и удобным для повторного использования.
* Проверка деления на ноль здесь **централизована**, нет дублирования кода в основном цикле.

В моем коде:

for (double x = StartX; x <= EndX + EPS; x += dX) {

if (x < 0 && fabs(b) > EPS) { ... }

else if (x > 0 && fabs(b) < EPS) { ... }

else { ... }

}

* Весь код вычисления F находится прямо в main.
* Проверки деления на ноль и формулы повторяются **для каждого случая**.
* Нет отдельной структуры для обработки ошибок.
* Это делает код **менее гибким** и сложнее поддерживаемым, но для простой задачи такой подход тоже работает.

Обработка ошибок:

* **Код chatgpt :** ошибка возвращается как строка, и можно отдельно проверить result.error и реагировать на неё.
* **В** моем **коде:** ошибка выводится сразу через cout внутри цикла.

**Вывод:** первый код более «чистый» и безопасный с точки зрения архитектуры, второй — более прямолинейный.

|  |  |  |
| --- | --- | --- |
| **Параметр** | Код chatgpt | Мой код |
| Модульность | Высокая, функция + структура | Низкая, всё в main |
| Обработка ошибок | Через optional + строку | Сразу через cout |
| Повторяемость / тестирование | Легко тестировать | Трудно тестировать отдельно |
| Проверка диапазона | Нет | Есть (dX > 0, StartX <= EndX) |
| Читаемость | Хорошая, логика выделена | Средняя, длинный main |
| Функциональные вычисления F | То же самое | То же самое |

Вывод

В ходе практической работы была изучена и закреплена работа с операторами условия (if, else if, else, switch) и циклами (for, while, do while) на языке Си. Практические задания позволили на практике реализовать различные алгоритмы, включая ветвления и повторяющиеся действия, что углубило понимание структуры программ и логики их выполнения.

Дополнительно была проведена работа с системой контроля версий **Git**, что позволило освоить основные команды для инициализации репозитория, фиксации изменений и сравнения версий кода. Сравнение собственного кода с предложенным ChatGPT примером показало преимущества и недостатки автоматических решений: выявились оптимизации, которые можно было применить, а также ошибки, которых удалось избежать благодаря ручной проверке.

В целом работа способствовала формированию навыков разработки, тестирования и анализа кода, а также освоению инструментов совместной работы и контроля версий, что является важной частью современного процесса программирования.

Список литературы:

<https://else.fcim.utm.md/pluginfile.php/32157/mod_resource/content/1/devcpp_1.pdf>  
<https://else.fcim.utm.md/pluginfile.php/32209/mod_resource/content/1/labexemplu.pdf>  
<https://chatgpt.com>  
<https://overapi.com/cpp>