**МИНОБРНАУКИ РОССИИ**
**САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ**
**ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**
**«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)**
**Кафедра информационной безопасности**

# КУРСОВАЯ РАБОТА

**по дисциплине «Программирование»**

**Тема: Приложение «Калькулятор»**

Студент гр. 4361 _____ Артемьев Д. Н.

Преподаватель _____ Халиуллин Р. А.

Санкт-Петербург
2024

**ЗАДАНИЕ**
**НА КУРСОВУЮ РАБОТУ**

Студент Артемьев Д.Н.

Группа 4361

Тема работы: Приложение «Калькулятор»

Исходные данные:
Разработать на языке программирования C или C++ (по выбору студента) приложение для выполнения простых расчетов с использованием основных арифметических операций (сложение, вычитание, умножение, деление). Значения для расчета (не менее двух значений) и арифметическая операция вводятся пользователем. Поддержка вещественных значений (значений с плавающей точкой) не обязательна. Допускается возникновение целочисленного переполнения в результате выполнения арифметической операции. Ввод/вывод значений должен осуществляться в десятичной системе счисления. Приложение должно иметь консольный или графический интерфейс (по выбору студента). В интерфейсе приложения допускается использовать буквы латинского алфавита для транслитерации букв алфавита русского языка. Интерфейс приложения должен быть интуитивно понятным и содержать подсказки для пользователя. В исходном коде приложения должны быть реализованы функции. В исходном коде приложения должны быть реализованы проверки аргументов реализованных функций и проверки возвращаемых функциями значений (для всех функций — как сторонних, так и реализованных). Приложение должно корректно обрабатывать ошибки, в том числе ошибки ввода/вывода, выделения/освобождения памяти и т. д.

Содержание пояснительной записки:
Введение, теоретическая часть, реализация программы, результаты тестирования программы, заключение, список использованных источников, прило-

жение 1 – руководство пользователя, приложение 2 – блок-схема алгоритма, приложение 3 – исходный код программы.

Предполагаемый объем пояснительной записки:
Не менее 25 страниц.

Дата выдачи задания: 10.11.2024

Дата сдачи реферата: 23.12.2024

Дата защиты реферата: 28.12.2024

Студент гр. 4361 _____ Артемьев Д. Н.

Преподаватель _____ Халиуллин Р. А.

## АННОТАЦИЯ

Цель работы – реализация программы с консольным интерфейсом, при помощи которой можно выполнять арифметические операции, а именно: сложение, вычитание, умножение и целочисленное деление – с двумя целыми числами.

Язык реализации программы – C. Были написаны проверки реализованных функций на подходящие значения, например, проверка на действительность указателей, а также проверки сторонних функций, в том числе ошибки ввода/вывода. В интерфейсе используется английский язык.

Результатом работы является приложение, соответствующее заданию.

## SUMMARY

The purpose of the work is to implement a program with a console interface that allows arithmetic operations, namely addition, subtraction, multiplication and integer addition, to be performed on two integers.

The implementation language of the program is C. Checks of the implemented functions for suitable values were written, for example pointer validity checks, as well as checks of third-party functions, including I/O errors. The user interface is in English.

The result of the work is an application that meets the requirements.

**СОДЕРЖАНИЕ**

# ВВЕДЕНИЕ

Цель работы – реализация программы «Калькулятор» с консольным интерфейсом, позволяющая выполнять арифметические операции такие, как сложение, вычитание, умножение и целочисленное деление.

Программа написана на языке программирования C.

Программа работает следующим образом: пользователь вводит арифметическое выражение без пробелов, и ему выводится результат, если в ходе работы программы не было никаких ошибок, в противном случае же будет выведено сообщение об ошибке.

Код содержит пять реализованных функций.

Четыре (GetSum, GetDiff, GetProd, GetQuo) – по одной на каждую арифметическую операцию – в качестве аргументов принимают два целых числа и указатель на переменную, в которой будет храниться результат. Кроме того, в них реализована проверка на действительность указателей, а в функции для получения результата деления еще проверка на то, что делитель не равен нулю.

Пятая функция (GetChoice) определяет, какую операцию хочет выполнить пользователь, анализируя символ, введенный человеком. Это реализовано при помощи оператора switch…case, а, так как при использовании этой условной конструкции есть случай default, то есть если ввод не совпадает ни с одним из заданных случаев, программа обработает ошибку ввода несуществующего оператора.

Кроме того, в функции main были написаны проверки ввода и вывода всего, кроме сообщений об ошибках.

Так что, если описывать кратко, то программа принимает арифметическое выражение, при помощи функции GetChoice определяет, какую операцию необходимо выполнить, и обращается к одной из четырех функции в зависимости от оператора, а затем возвращает результат вычислений.

# 1. ТЕОРЕТИЧЕСКАЯ ЧАСТЬ

Язык с появился в 1969–1973 годах, на ранних стадиях разработки операционной системы UNIX; наиболее продуктивным в плане развития языка был 1972 год. Ещё одна волна изменений достигла пика между 1977 и 1979 годами, когда была продемонстрирована переносимость ОС UNIX. В середине этого второго периода появилось первое широкодоступное описание языка: книга «Язык программирования С», часто называемая «белой книгой». Наконец, в середине 1980-х годов С был официально стандартизирован комитетом ANSI X3J11, который взял на себя ответственность за внесение в язык дальнейших изменений. До начала 1980-х годов язык ассоциировался практически лишь с одной UNIX, хотя уже тогда существовали его компиляторы для различных машинных архитектур и операционных систем. В последнее время область его применения существенно расширилась, и сегодня он входит в число наиболее часто используемых

С – универсальный язык программирования. Он тесно связан с системой UNIX, так как был разработан в этой системе, которая, как и большинство программ, работающих в ней, написаны на С. Однако язык не привязан жестко к какой-то одной операционной системе или машине. Хотя он и назван «языком системного программирования», поскольку удобен для написания компиляторов и операционных систем, оказалось, что на нем столь же хорошо писать большие программы другого профиля.

Многие важные идеи С взяты из языка BCPL, автором которого является Мартин Ричарде. Влияние BCPL на С было косвенным – через язык B, разработанный Кеном Томпсоном в 1970 г. для первой системы UNIX, реализованной на PDP-7.

BCPL и B – «бестиповые» языки. В отличие от них С обеспечивает разнообразие типов данных. Базовыми типами являются символы, а также целые и числа с плавающей точкой различных размеров. Кроме того, имеется возможность получать целую иерархию производных типов данных из указа-

телей, массивов, структур и объединений. Выражения формируются из операторов и операндов. Любое выражение, включая присваивание и вызов функции, может быть инструкцией. Указатели обеспечивают машинно-независимую адресную арифметику.

В качестве результата функции могут возвращать значения базовых типов, структур, объединений и указателей. Любая функция допускает рекурсивное обращение к себе. Как правило, локальные переменные функции – «автоматические», т. е. они создаются заново при каждом обращении к ней. Определения функций нельзя вкладывать друг в друга, но объявления переменных разрешается строить в блочно-структурной манере. Функции программы на С могут храниться в отдельных исходных файлах и компилироваться независимо. Переменные по отношению к функции могут быть внутренними и внешними. Последние могут быть доступными в пределах одного исходного файла или всей программы.

С – язык сравнительно «низкого уровня». Однако это вовсе не умаляет его достоинств, просто С имеет дело с теми же объектами, что и большинство компьютеров, т. е. с символами, числами и адресами. С ними, можно оперировать при помощи арифметических и логических операций, выполняемых реальными машинами.

В течение многих лет единственным определением языка С было первое издание книги «Язык программирования С». В 1983 г. Институтом американских национальных стандартов (ANSI) учреждается комитет для выработки современного исчерпывающего определения языка С. Результатом его работы явился стандарт для С («ANSI-C»), выпущенный в 1988 г. Большинство положений этого стандарта уже учтено в современных компиляторах.

Стандарт базируется на первоначальном справочном руководстве. По сравнению с последним язык изменился относительно мало. Одной из целей стандарта было обеспечить, чтобы в большинстве случаев существующие программы оставались правильными или вызывали предупреждающие сообщения компиляторов об изменении поведения.

C не является «строго типизированным» языком, но в процессе его развития контроль за типами был усилен. В первой версии C хоть не одобрялся, но разрешался бесконтрольный обмен указателей и целых, что вызывало большие нарекания, но это уже давным-давно запрещено. Согласно стандарту теперь требуется явное объявление или явное указание преобразования, что уже и реализовано в хороших компиляторах. Новый вид объявления функций – еще один шаг в этом направлении. Компилятор теперь предупреждает о большей части ошибок в типах и автоматически не выполняет преобразования данных несовместимых типов. Однако основной философией C остается то, что программисты сами знают, что делают; язык лишь требует явного указания об их намерениях.

К началу 1973 года основа современной версии C была готова. Язык и компилятор были достаточно мощными, чтобы позволить нам тем же летом переписать с их помощью ядро Unix для PDP-11 (в 1972 году Томпсон предпринял попытку написать систему на ранней версии C – когда в нём ещё не было структур – но быстро отказался от этой идеи). Также в этот период компилятор был перенесён на другие имеющиеся у нас ЭВМ – в частности, на Honeywell 635 и IBM 360/370. Поскольку язык не мог существовать в «вакууме» – были разработаны прототипы современных библиотек. В частности, Леск написал «переносимый пакет ввода-вывода», который позже был переработан и стал библиотекой «стандартного ввода-вывода» языка C. Хотя в ней не описывались некоторые дополнения к языку, которые вскоре стали общепринятыми, эта книга служила справочником по C в течение более 10 лет – до тех пор, пока не был разработан и принят его официальный стандарт. Керниган написал почти всю основную часть и главу о взаимодействии с операционной системой Unix.

C 1973 по 1980 год в язык были добавлены спецификаторы типов unsigned и long.

C повлиял на несколько языков, ставших его прямыми наследниками – хотя он не соперничает с многочисленным «потомством», которое оставил

Pascal.

C, как и любой другой язык программирования, не свободен от недостатков. Уровень старшинства некоторых операторов не является общепринятым, некоторые синтаксические конструкции могли бы быть лучше. Тем не менее, как оказалось, C – чрезвычайно эффективный и выразительный язык, пригодный для широкого класса задач.

# 2. РЕАЛИЗАЦИЯ ПРОГРАММЫ

## 2.1. Использованное ПО

Используемый язык программирования: C.

Операционная система: Windows 10 (x64).

Среда разработки: Code::Blocks 20.03 (x64).

Компилятор: GNU GCC Compiler.

## 2.2. Реализованные функции

Всего в программе есть шесть реализованных функций, а именно GetSum, GetDiff, GetProd, GetQuo, GetChoice, а также функция main.

1. Функция GetSum.

Функция GetSum вычисляет сумму двух переменных.

Исходный код функции GetSum находится в файле main.c.

Объявление функции:

```
int GetSum(int a, int b, int* res);
```

Тип функции: `int`.

Аргументы функции:

- `a` – первое слагаемое, тип аргумента: `int`;
- `b` – второе слагаемое, тип аргумента: `int`;
- `res` – указатель на переменную, в которую будет возвращена сумма переменных, тип аргумента: `int*`.

Возвращаемое функцией значение:

- 0 – функция завершилась без ошибок;
- 3 – значение третьего аргумента является некорректным.

2. Функция GetDiff.

Функция GetDiff вычисляет разность двух переменных.

Исходный код функции GetDiff находится в файле main.c.

Объявление функции:

```
int GetDiff(int a, int b, int* res);
```

Тип функции: `int`.

Аргументы функции:

- `a` – уменьшаемое, тип аргумента: `int`;
- `b` – вычитаемое, тип аргумента: `int`;
- `res` – указатель на переменную, в которую будет возвращена разность переменных, тип аргумента: `int*`.

Возвращаемое функцией значение:

- 0 – функция завершилась без ошибок;
- 3 – значение третьего аргумента является некорректным.

3. Функция GetProd.

Функция GetProd вычисляет произведение двух переменных.

Исходный код функции GetProd находится в файле main.c.

Объявление функции:

```
int GetProd(int a, int b, int* res);
```

Тип функции: `int`.

Аргументы функции:

- `a` – первый множитель, тип аргумента: `int`;
- `b` – второй множитель, тип аргумента: `int`;
- `res` – указатель на переменную, в которую будет возвращено произведение переменных, тип аргумента: `int*`.

Возвращаемое функцией значение:

- 0 – функция завершилась без ошибок;
- 3 – значение третьего аргумента является некорректным.

4. Функция GetQuo.

Функция GetQuo вычисляет частное двух переменных.

Исходный код функции GetQuo находится в файле main.c.

Объявление функции:

```
int GetQuo(int a, int b, int* res);
```

Тип функции: `int`.

Аргументы функции:

- `a` – делимое, тип аргумента: `int`;
- `b` – делитель, тип аргумента: `int`;
- `res` – указатель на переменную, в которую будет возвращено частное переменных, тип аргумента: `int*`.

Возвращаемое функцией значение:

- 0 – функция завершилась без ошибок;
- 2 – значение второго аргумента является некорректным, то есть равно нулю;
- 3 – значение третьего аргумента является некорректным.

5. Функция GetChoice.

Функция GetChoice определяет, какую арифметическую операцию выбрал пользователь, отталкиваясь от математического символа.

Исходный код функции GetChoice находится в файле main.c.

Объявление функции:

```
int GetChoice(char c, int a, int b, int* res);
```

Тип функции: `int`.

Аргументы функции:

- `c` – символ математической операции, тип аргумента: `char`;
- `a` – первая переменная, введенная пользователем, тип аргумента: `int`;
- `b` – вторая переменная, введенная пользователем, тип аргумента: `int`;
- `res` – указатель на переменную, в которую будет возвращен результат переменных, тип аргумента: `int*`.

Возвращаемое функцией значение:

- 0 – функция завершилась без ошибок;
- 1 – значение первого аргумента является некорректным;
- 4 – значение четвертого аргумента является некорректным.

6. Функция main.

Функция main служит отправной точкой для выполнения программы.

Исходный код функции main находится в файле main.c.

Объявление функции:

```
int main();
```

Тип функции: `int`.

У данной функции нет аргументов.

Возвращаемое функцией значение:

- 0 – функция завершилась без ошибок;
- 1 – ошибка вывода;
- 2 – ошибка ввода;
- 3 – введен некорректный оператор (ошибка оператора);
- 4 – ошибка деления на ноль;
- 5 – ошибка некорректности указателя на переменную, в которой хранится результат.

# 3. РЕЗУЛЬТАТЫ ТЕСТИРОВАНИЯ ПРОГРАММЫ

На рисунке 1 представлено то, что видит пользователь при запуске программы.

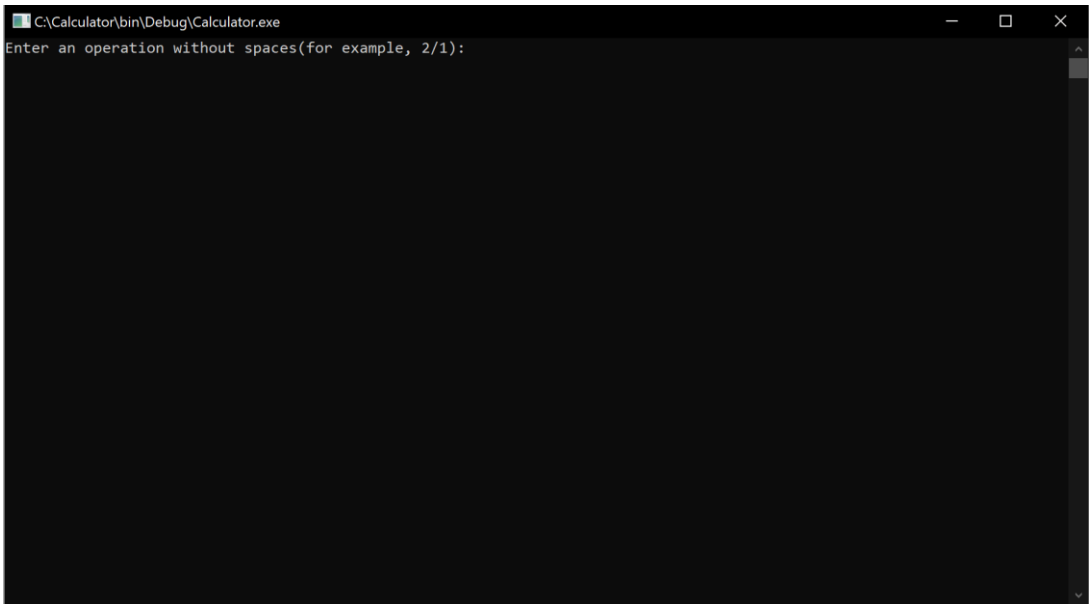



Рисунок 1 – Стартовый экран программы

Пользователю предложено ввести арифметическое выражение без пробелов, на рисунке 2 показано, что вернет программа.

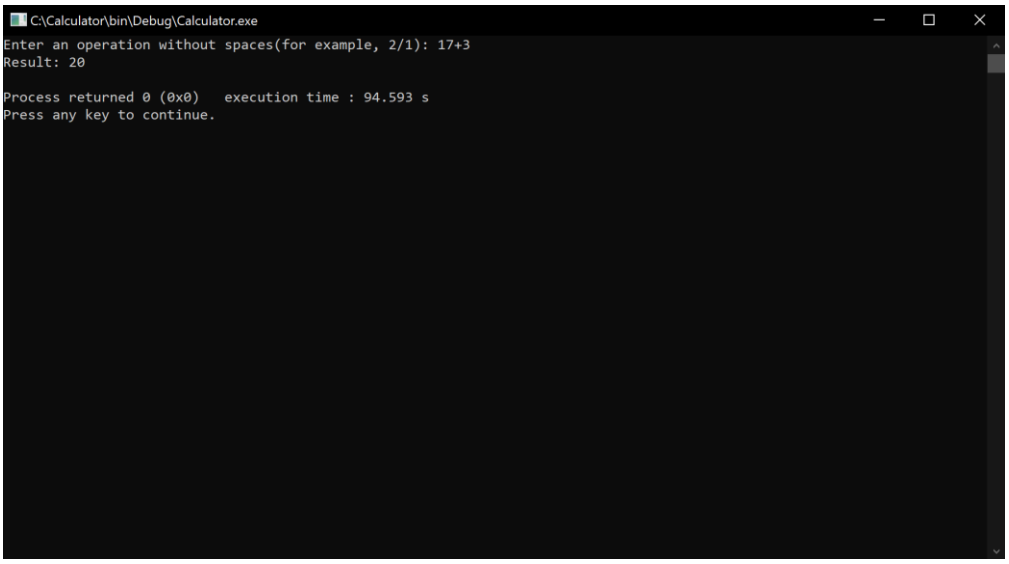



Рисунок 2 – Вывод результата вычислений

Программа корректно обрабатывает ошибки. На рисунке 3 показано, что программа выводит в случае попытки деление на ноль.

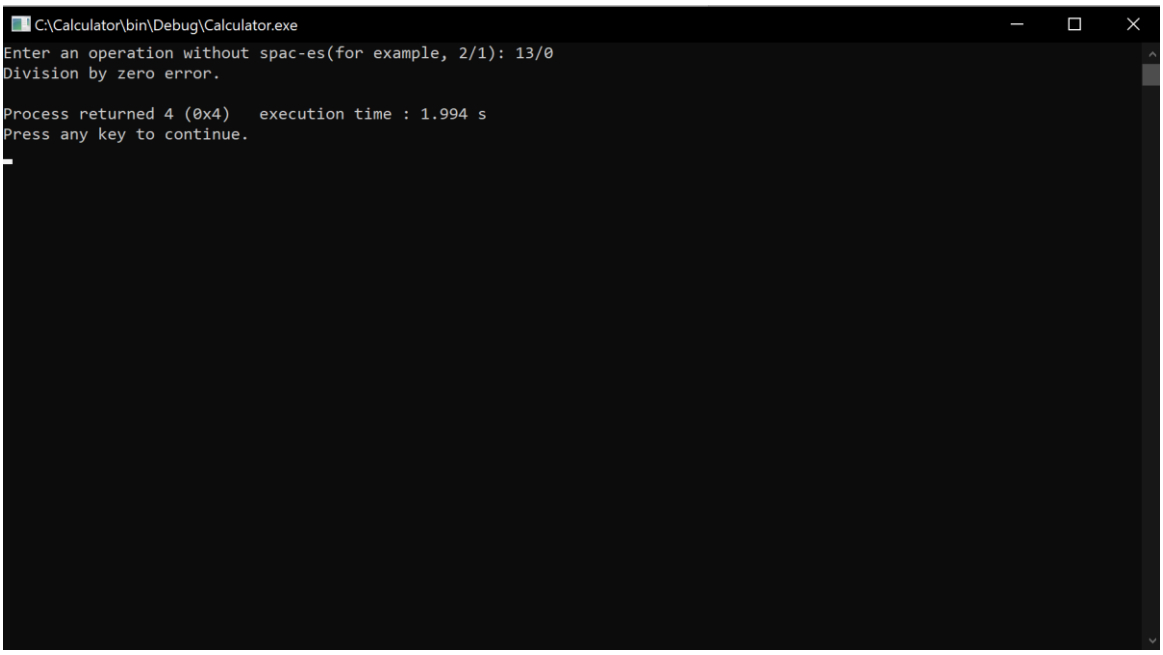



Рисунок 3 – Обработка ошибки деления на ноль

На рисунке 4 показано, что программа выводит в случае попытки использования какого-либо символа, кроме разрешенных, то есть, кроме «+», «-», «*» и «/».

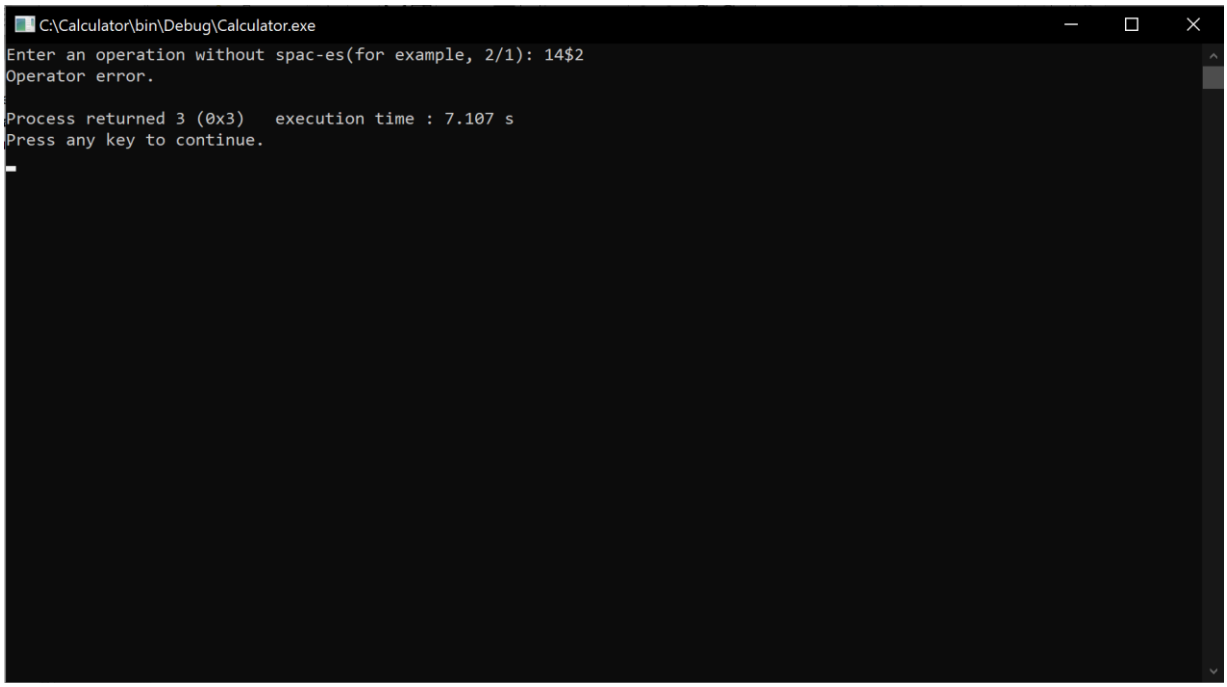



Рисунок 4 – Обработка ошибки оператора

На рисунке 5 показано, что программа выводит в случае попытки ввода арифметического выражения с пробелами.

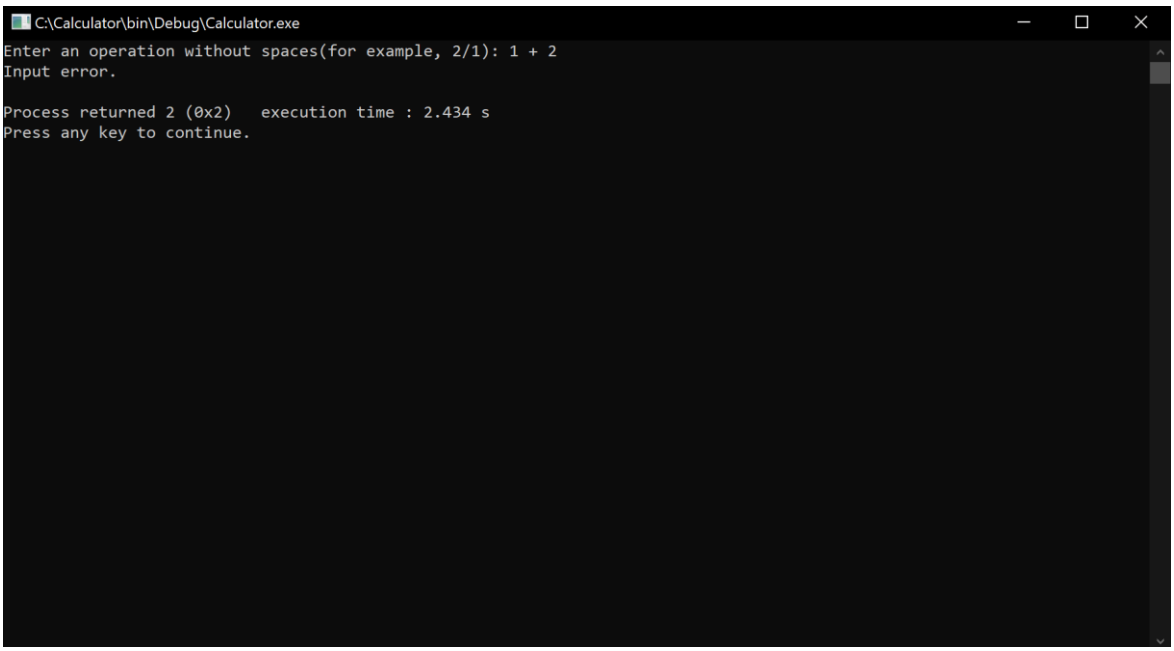



Рисунок 5 – Обработка ошибки ввода

Кроме того, программа корректно обрабатывает ошибки вывода и недействительного указателя.

# ЗАКЛЮЧЕНИЕ

Программа, написанная на языке программирования C, позволяет выполнять простые расчеты с использованием основных арифметических операций (сложения, вычитания, умножения, деления). Ввод/вывод значений осуществляется в десятичной системе счисления. Интерфейс интуитивно понятен. В коде реализованы функции, а также проверка аргументов и возвращаемых функцией значений. Приложение корректно обрабатывает ошибки.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Брайан В. Керниган, Деннис М. Ритчи, Язык программирования Си – М.: Вильямс 2019. — 288 С.

2. Dennis M. Ritchie, The development of the C programming language // Programming languages (Electronic computers) – History: Bergin, Thomas J; Gibson, Richard G; History of Programming Languages Conference (2nd : 1993 : Cambridge, Mass.) / New York : ACM Press ; Reading, Mass. : Addison-Wesley Pub. Co. — 1996. — P. 671 — 698.

# ПРИЛОЖЕНИЕ 1. РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ

1. Краткое описание программы.

Перед вами программа «Калькулятор». Вы можете ввести арифметическое выражение без пробелов с использованием одной из четырех операций, а именно сложения, вычитания, умножения или целочисленного деления. Программа в свою очередь вернет результат вычислений или оповещение о произошедшей ошибке.

2. Минимальные системные требования.

- Операционная система: Windows 10 (x64);
- Процессор: не менее 1 ГГц или SoC;
- ОЗУ: 2 ГБ;
- Место на жестком диске: 20 ГБ;
- Видеоадаптер: DirectX 9 или более поздняя версия с драйвером WDDM 1.0;
- Экран: 800 x 600.

3. Установка программы.

Для установки программы необходимо скопировать исполняемый файл программы Calculator.exe в выбранную пользователем директорию. Других файлов необходимых для работы программы нет.

4. Запуск программы.

Для запуска программы необходимо запустить исполняемый файл программы Calculator.exe из директории, в которую была установлена программа.

5. Работа с программой.

После запуска программы откроется ее стартовый экран, он показан на рисунке 6.

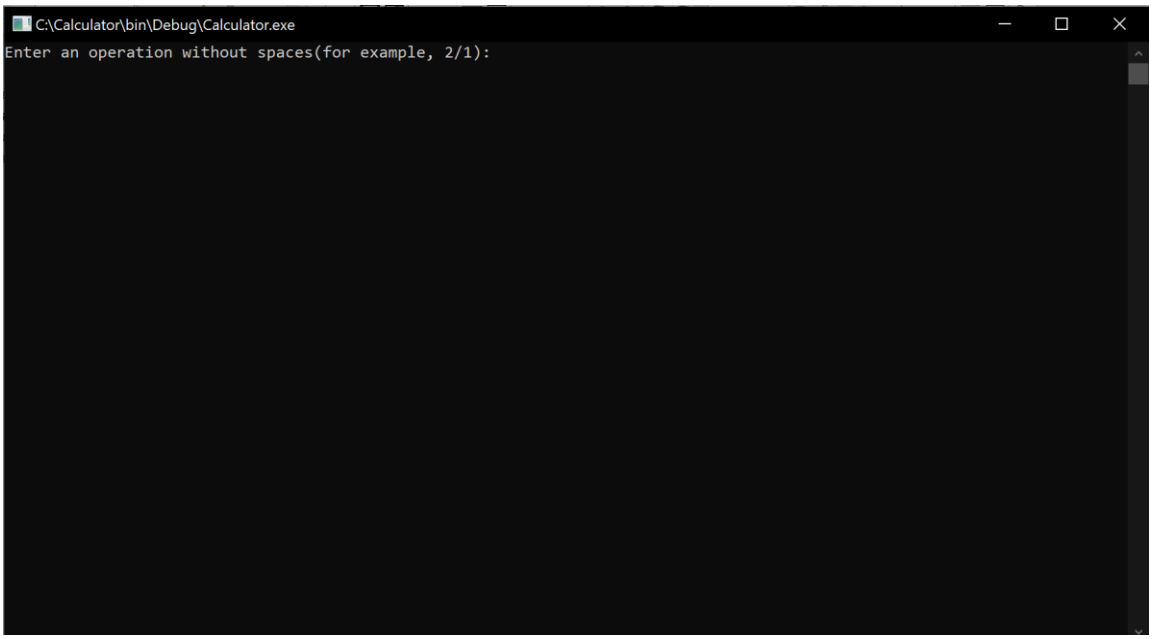



Рисунок 6 – Стартовый экран программы

На нем указана подсказка для пользователя, а именно пример корректного ввода и информация о том, что вставлять выражение надо без пробелов.

То есть необходимо ввести два целых числа в десятичной системе счисления, а между ними символ арифметической операции, без пробелов.

В вводе может быть использован один из четырех математических символов, а именно «+» для сложения, «-» для вычитания, «*» для умножения и «/» для целочисленного деления.

Если ввод будет корректным, то программа выведет результат, как на рисунке 7.

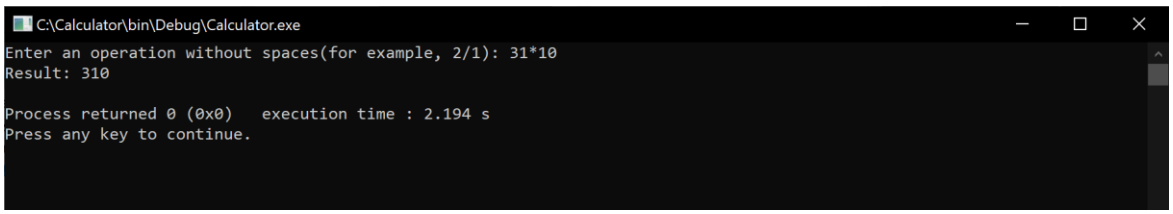



Рисунок 7 – Корректная работа программы

Выше был указан список возможных математических символов, если же попробовать использовать любую другую, то программа выдаст ошибку оператора, как на рисунке 8.

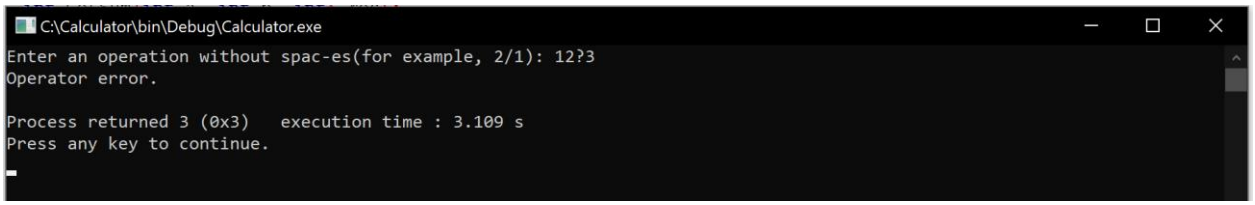



Рисунок 8 – Ошибка некорректного оператора

Как уже было сказано выше, для корректной работы программы необходимо вводить выражение без пробелов, если же ввести с пробелами, программа выдаст ошибку ввода, как на рисунке 9.


Рисунок 9 – Ошибка ввода

Кроме того, если пользователь введет выражение, которое включает в себя деление на ноль, программа обработает и эту ошибку, как на рисунке 10.

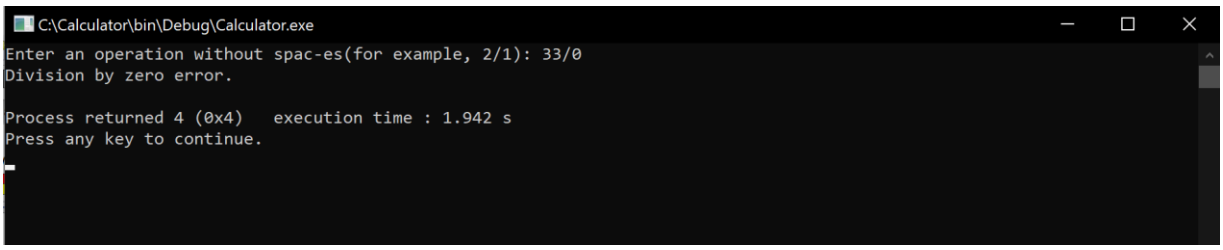
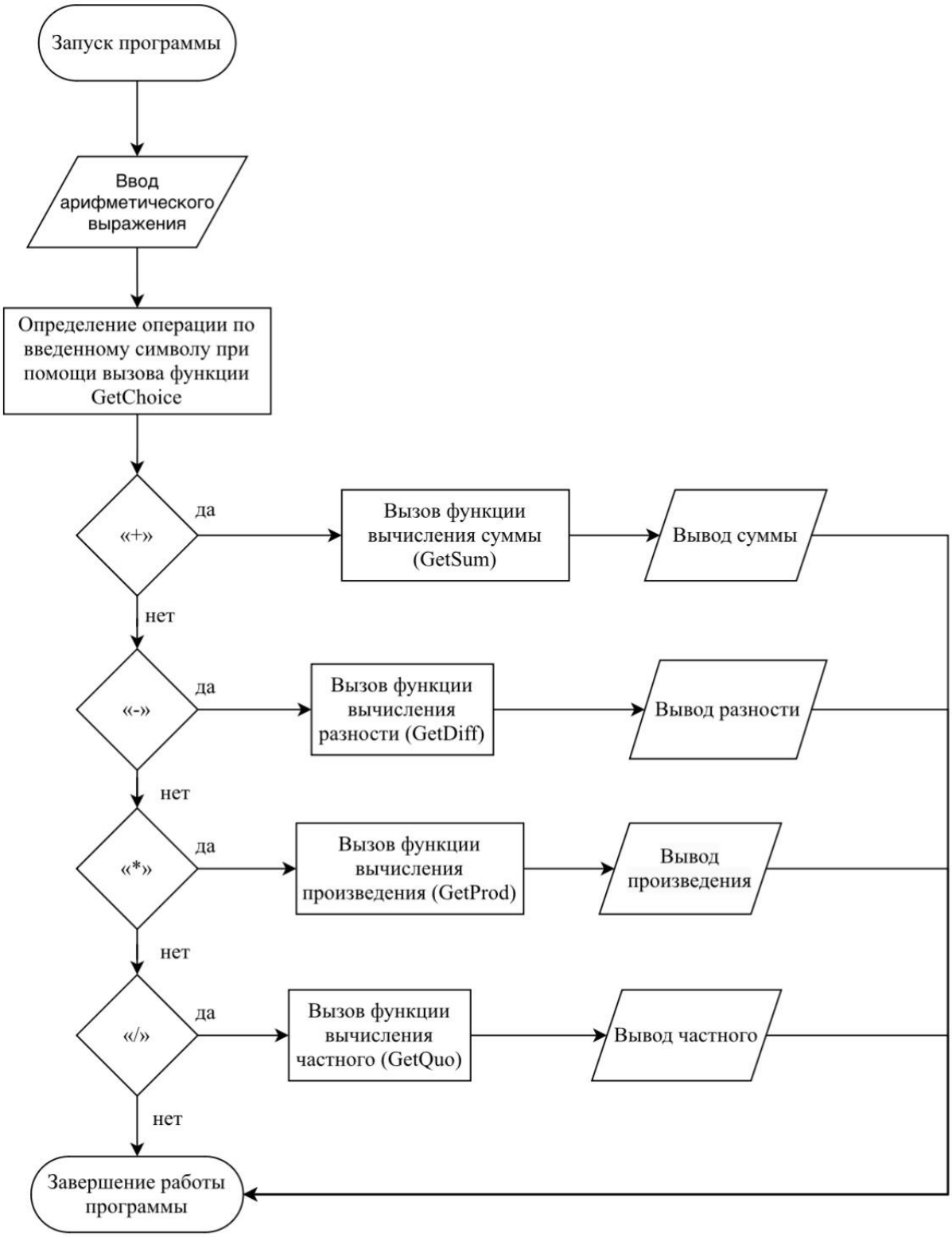



Рисунок 10 – Ошибка деления на ноль

Кроме этого, программа также обрабатывает ошибки недействительных операторов в реализованных функциях, а также ошибки вывода.

Получается, что при корректном вводе прогорамма выводит корректный результат вычислений, а при некорректном сообщает пользователю, что именно надо исправить в арифметическом выражении, чтобы избежать проблем в работе с программой.

# ПРИЛОЖЕНИЕ 2. БЛОК-СХЕМА АЛГОРИТМА



## ПРИЛОЖЕНИЕ 3. ИСХОДНЫЙ КОД

```
#include <stdio.h>
#include <stddef.h>

int GetSum(int a, int b, int* res);
int GetDiff(int a, int b, int* res);
int GetProd(int a, int b, int* res);
int GetQuo(int a, int b, int* res);
int GetChoice(char c, int a, int b, int* res);

int GetSum(int a, int b, int* res)
{
    if (res == NULL)
    {
        return 3;
    }
    *res = a + b;
    return 0;
}

int GetDiff(int a, int b, int* res)
{
    if (res == NULL)
    {
        return 3;
    }
    *res = a - b;
    return 0;
}
```

```
int GetProd(int a, int b, int* res)
{
    if (res == NULL)
    {
        return 3;
    }
    *res = a * b;
    return 0;
}

int GetQuo(int a, int b, int* res)
{
    if (b == 0)
    {
        return 2;
    }
    if (res == NULL)
    {
        return 3;
    }
    *res = a / b;
    return 0;
}

int GetChoice(char c, int a, int b, int* res)
{
    if (res == NULL)
    {
        return 4;
```

```
        }

        switch (c)
        {
            case '+':
                return GetSum(a, b, res);
            case '-':
                return GetDiff(a, b, res);
            case '*':
                return GetProd(a, b, res);
            case '/':
                return GetQuo(a, b, res);
            default:
                return 1;
        }
        return 0;
    }

    int main()
    {
        int a, b, res, status;
        char c;
        if (printf("Enter an operation without spac-
es(for example, 2/1): ") < 0)
        {
            printf("Output error.\n");
            return 1;
        }
        if (scanf("%d%c%d", &a, &c, &b) != 3)
        {
```

```
            printf("Input error.\n");
            return 2;
        }
        status = GetChoice(c, a, b, &res);
        if (status == 0)
        {
            if(printf("Result: %d\n", res) < 0)
            {
                printf("Output error.\n");
                return 1;
            }
        }
        else if (status == 1)
        {
            printf("Operator error.\n");
            return 3;
        }
        else if (status == 2)
        {
            printf("Division by zero error.\n");
            return 4;
        }
        else if (status == 3 || status == 4)
        {
            printf("Pointer error.\n");
            return 5;
        }
        return 0;
    }
```