

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. 3363

Минко Д.А.

Преподаватель

Халиуллин Р.А.

Санкт-Петербург

2023

ЗАДАНИЕ НА КУРСОВУЮ РАБОТУ

Студент Минко Д.А.

Группа 3363

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, возникновение и начало развития языка Си, реализация программы, основные сведения о программном обеспечении, реализованные функции, результат тестирования программы, сведения о результатах тестирования программы, заключение, список использованных источников, приложение 1 – руководство пользователя, приложение 2 – блок-схема алгоритма, приложение 3 – исходный код программы.

Предполагаемый объем пояснительной записки:

Не менее 30 страниц.

Дата выдачи задания: 12.11.2023

Дата сдачи реферата: 25.12.2023

Дата защиты реферата: 25.12.2023

Студент

Минко Д.А.

Преподаватель

Халиуллин Р.А.

АННОТАЦИЯ

Курсовая работа посвящена разработке приложения для выполнения простых расчётов с использованием основных арифметических операций (сложение, вычитание, умножение, деление) на языке программирования С.

Работа охватывает процесс проектирования, написание и тестирование программного кода для создания функционального калькулятора. В работе представлены алгоритмы базовых математических операций, а также обработки ошибок ввода и вывода данных. Помимо этого, в рамках данной работы создано руководство по использованию итогового приложения, включающее в себя пошаговые инструкции, продемонстрирована работа с математическими операциями и предоставлена информация об обработке ошибок в процессе использования программы.

SUMMARY

The course work is devoted to the development of an application for performing simple calculations using basic arithmetic operations (addition, subtraction, multiplication, division) in the C programming language.

The work covers the design process, writing and testing program code to create a functional calculator. The paper presents algorithms for basic mathematical operations, as well as processing errors in data input and output. In addition, as part of this work, a manual for using the final application has been created, which includes step-by-step instructions, work with mathematical operations has been demonstrated, and information about error handling while using the program has been provided.

СОДЕРЖАНИЕ

Введение	6
1. Теоретическая часть	7
1.1. Начало развития языков программирования	7
1.2. Возникновение языка Си	9
2. Реализация программы	11
2.1. Основные сведения о программном обеспечении	11
2.2. Реализованные функции	11
3. Результаты тестирования программы	15
3.1. Сведения о результатах тестирования программы	15
Заключение	23
Список использованных источников	24
Приложение 1. Руководство пользователя	25
Приложение 2. Блок-схема алгоритма	27
Приложение 3. Исходный код программы	28

ВВЕДЕНИЕ

Цель работы заключается в написании программы, которая будет выполнять простые расчёты с использованием основных арифметических операций (сложение, вычитание, умножение, деление). Для реализации был выбран язык программирования С.

Для достижения поставленной цели поставлены следующие задачи:

1. Создать блок-схему, описывающую алгоритм работы данной программы.
2. Реализовать функции для следующих арифметических действий с целыми числами:
 - Сложение;
 - Вычитание;
 - Умножение;
 - Деление.
3. Реализовать проверки аргументов реализованных функций.
4. Реализовать проверки возвращаемых функциями значений для всех функций.
5. Реализовать корректную обработку ошибок, которые могут возникать в результате выполнения программы.
6. Разработать “Руководство пользователя”.

1. ТЕОРЕТИЧЕСКАЯ ЧАСТЬ

1.1. Начало развития языков программирования

Первой попыткой создать программируемый вычислитель была работа Чарльза Бэббиджа. Аналитическая машина напоминала механизм сложной музыкальной шкатулки и применялась исключительно для математических выражений. Она использовала первый в мире компьютерный язык – язык программирования, который назывался “Коды операций аналитической машины”. Создан он был примерно в 1837 году. Сам алгоритм написала Ада Лавлейс, которая считается первым программистом во всем мире.

Конрад Цузе – немецкий инженер – успешно работал в 1942-1946 гг. над первым в мире языком программирования высокого уровня (ЯВПУ) под названием “Планкалкюль” (Plankalkül). Z4 – его обрабатывающая машина – являлась абсолютно механической, язык не имел ни компиляторов, ни интерпретаторов. Тем не менее, Цузе на этом языке написал программу, анализирующую ситуацию на шахматной доске.

Джон Мокли предложил проект электронной вычислительной машины (ЭВМ), работающей на вакуумных лампах. Инженеры начали с простейших алгоритмов, всего через год ЭВМ была полностью создана. В 1945 году данная ЭВМ, получившая название ENIAC (Электронный числовой интегратор и вычислитель) была перемещена в институт баллистических исследований Армии США, с ее помощью исследовали различные военные задачи. Над проектом также работали Джон фон Нейман и Преспер Эккерт.

В 1957 году Джон Бэкус предложил язык для ЭВМ, являющийся высокоуровневым, понятным, компилируемым языком. Таковым стал Fortran, первый ЯПВУ для ЭВМ, получивший в дальнейшем развитие и ставший прародителем многих языков. На его основе позднее разработали Algol, ответ на недостатки Fortran. на Algol в большей степени основываются языки ветви, содержащей C, однако это лишь промежуточное звено, Fortran в

целом оказался более используем, поэтому не будем включать Algol в схему как незначительный.

Программирование развивалось, но каждый язык требовал больших затрат времени для обучения специалистов. Джон Кемени и Томас Курц в 1964 году на основе Fortran создали язык, предназначенный для популяризации программирования – Basic. Язык прекрасно справлялся со своей задачей, о чем говорят небывалые успехи в использовании языка для обучения студентов, а в дальнейшем школьников, на протяжении 50-ти лет.

Следующие три языка стоит рассматривать как связку. Первый из них CPL – язык, предназначенный для написания компиляторов к другим языкам. Он являлся низкокачественным, и стал черновиком для BCPL, разработанного в Кембриджском университете Мартином Ричардсом. Мартин просто убрал из CPL все лишнее и весной 1967 подготовил компилятор своего языка. BCPL был прост в портируемости, отчего стал популярен в те годы. Из BCPL развился язык B, созданный Кеном Томпсоном в сотрудничестве с Денисом Ритчи в 1969. Он больше подходил для реализации простых подсистем, однако был недостаточно эффективным для написания чего-то большего. Что интересно, классическая программа “Hello, world!” была впервые написана на языке B.

Николос Вирт и Кэтлин Йенсен стремились к комбинации производительности и простоты. И в 1970 году им удалось создать язык, подходящий под эти параметры – Pascal. Язык был предназначен изначально для обучения программированию, был создан так, что код понимался интуитивно, а составить на нем можно было любые алгоритмы. Позднее Pascal получил популярность даже за рамками преподавательской деятельности.

Отсутствие типизации в то время невероятно усложняло работу с символами и адресацией, именно поэтому Деннис Ритчи в 1972 представил язык C миру. Однако, C имел множество недостатков. Одним из них было

отсутствие понятия контрактности, вследствие чего работы плохих программистов превращались в настоящий хаос.

1.2. Возникновение языка Си

Язык программирования Си был разработан в 1972 г. сотрудником фирмы Bell Laboratories Деннисом Ритчи (Dennis M. Ritchie), который является одним из авторов операционной системы UNIX. Язык Си впоследствии был использован для программирования этой системы, а также богатой библиотеки обслуживающих программ, поскольку, являясь универсальным языком общего назначения, язык Си удобен для программирования системных задач.

В отличие от многих языков программирования (Ада, Алгол-60, Алгол-68 и т.д.), которые вступали в силу после принятия соответствующих национальных и международных стандартов, язык Си вначале был создан как рабочий инструмент, не претендующий на широкое применение. Стандарта на язык Си до 1989 г. не существовало, и в качестве формального описания разработчики компиляторов использовали первое издание книги Б. Кернигана и Д. Ритчи, вышедшее в США в 1978 г. (переведена на русский язык в 1985 г.). Роль неформального стандарта языка Си сохранилась за этой книгой и в настоящее время. Не случайно в литературе и документации по компиляторам на эту работу обозначается специальным сокращением K&R. Второе издание книги Б. Кернигана и Д. Ритчи описывает язык Си в стандартизованном Американским институтом национальных стандартов виде (стандарт ANSI языка Си).

Появление микрокомпьютеров укрепило позиции языка Си. Было создано более 30 его новых компиляторов, а после проведения Американским национальным институтом стандартов ANSI (American National Standard Institute) работ по стандартизации в области программирования начали разрабатываться компиляторы, соответствующие опубликованному весной 1986 г. проекту стандарта.

Первым компилятором по стандарту ANSI явилась система Турбо Си версии 1.0 фирмы Borland International. Эта система, состоящая из компилятора языка Си, связанного с ним редактора, компоновщика и библиотек, обеспечила пользователям удобную интегрированную операционную среду, а также существенно облегчила профессиональное программирование, в котором определяющими параметрами являются высокая скорость компиляции, высокое качество генерируемого кода и малая потребность в оперативной памяти.

Несмотря на то что Турбо Си 1.0 оправдал ожидания большинства профессиональных программистов, он имел ряд недоработок из-за поспешности его внедрения на рынок. К таким недоработкам относились ошибки в некоторых библиотечных функциях и в редакторе, отсутствие графической библиотеки и отладчика. Весной 1988 г. фирма Borland представила пользователям усовершенствованную версию системы Турбо Си под номером 1.5. В ней исключены ошибки версии 1.0 и добавлена обширная графическая библиотека, насчитывающая около 100 новых функций, а также заложена разработка отладчика. В конце 1988 г. появилась версия Турбо Си 2.0 Professional. В её составе, кроме интегрированного компилятора и компоновщика с использованием команд строки, входят расширенная графическая библиотека, встроенная система запуска программ, а также ассемблер с высоким быстродействием и новый символьный отладчик.

2. РЕАЛИЗАЦИЯ ПРОГРАММЫ

2.1. Основные сведения о программном обеспечении

Для написания программы был выбран язык программирования C, операционная система Windows 11, version 22H2, среда разработки Code::Blocks, компилятор – GNU GCC.

2.2. Реализованные функции

Реализованные функции: `GetDivision()`, `GetAddition()`, `GetSubtraction()`, `GetMultiplication()`, `main()`.

Функция `main`.

Функция `main` позволяет рассчитать по двум переменным одну из арифметических операций – сложение, вычитание, умножение, деление, на выбор пользователя.

Исходный код функции `main` находится в файле `main.c`.

Объявление функции:

```
int main ();
```

Тип функции: `int`.

Аргументы функции:

- Отсутствуют.

Возвращаемое функцией значение:

- 0 — функция завершилась без ошибок;
- 1 — возникла ошибка функции `printf()`;
- 2 — возникла ошибка функции `printf()`;
- 3 — возникла ошибка функции `printf()`;
- 4 — возникла ошибка функции `printf()`;
- 5 — возникла ошибка функции `printf()`;
- 6 — возникла ошибка функции `printf()`;
- 7 — возникла ошибка функции `printf()`;
- 8 — возникла ошибка функции `printf()`;

- 9 — возникла ошибка функции printf();
- 10 — возникла ошибка функции printf();
- 11 — возникла ошибка функции printf();
- 12 — возникла ошибка функции printf();
- 13 — возникла ошибка функции printf();

Функция GetDivision.

Функция GetDivision определяет значение целочисленного деления двух чисел.

Исходный код функции GetDivision находится в файле main.c.

Объявление функции:

```
int GetDivision (int FirstNumber, int SecondNumber,
int* Value);
```

Тип функции: int.

Аргументы функции:

- FirstNumber — переменная, в которой хранится значение делимого тип аргумента: int;
- SecondNumber — переменная, в которой хранится значение делителя тип аргумента: int;
- Value — указатель на переменную, в которую будет возвращено частное от аргументов FirstNumber и SecondNumber, тип аргумента: int*.

Возвращаемое функцией значение:

- 0 — функция завершилась без ошибок;
- 2 — значение второго аргумента функции является некорректным;
- 3 — значение третьего аргумента функции является некорректным.

Функция GetAddition.

Функция GetAddition определяет значение целочисленного сложения двух чисел.

Исходный код функции GetAddition находится в файле main.c.

Объявление функции:

```
int GetAddition (int FirstNumber, int SecondNumber,  
int* Value);
```

Тип функции: int.

Аргументы функции:

- FirstNumber — переменная, в которой хранится значение первого слагаемого, тип аргумента: int;
- SecondNumber — переменная, в которой хранится значение второго слагаемого, тип аргумента: int;
- Value — указатель на переменную, в которую будет возвращено значение сложения аргументов FirstNumber и SecondNumber, тип аргумента: int*.

Возвращаемое функцией значение:

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

Функция GetSubtraction.

Функция GetSubtraction определяет значение целочисленной разности.

Исходный код функции GetSubtraction находится в файле main.c.

Объявление функции:

```
int GetSubtraction (int FirstNumber, int  
SecondNumber, int* Value);
```

Тип функции: int.

Аргументы функции:

- FirstNumber — переменная, в которой хранится значение уменьшаемого, тип аргумента: int;
- SecondNumber — переменная, в которой хранится значение вычитаемого тип аргумента: int;

- Value — указатель на переменную, в которую будет возвращено значение разности от аргументов FirstNumber и SecondNumber, тип аргумента: int*.

Возвращаемое функцией значение:

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

Функция GetMultiplication.

Функция GetMultiplication определяет значение целочисленного умножения двух чисел.

Исходный код функции GetMultiplication находится в файле main.c.

Объявление функции:

```
int GetMultiplication (int FirstNumber, int  
SecondNumber, int* Value);
```

Тип функции: int.

Аргументы функции:

- FirstNumber — переменная, в которой хранится значение первого множителя, тип аргумента: int;
- SecondNumber — переменная, в которой хранится значение второго множителя, тип аргумента: int;
- Value — указатель на переменную, в которую будет возвращено значение произведения аргументов FirstNumber и SecondNumber, тип аргумента: int*.

Возвращаемое функцией значение:

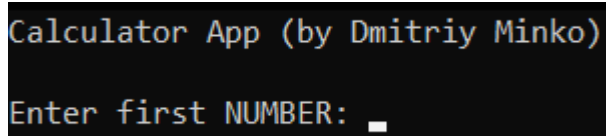
- 0 — функция завершилась без ошибок;
- 3 — значение третьего аргумента функции является некорректным.

3. РЕЗУЛЬТАТЫ ТЕСТИРОВАНИЯ ПРОГРАММЫ

3.1. Сведения о результатах тестирования программы

1. Ввод первого числа.

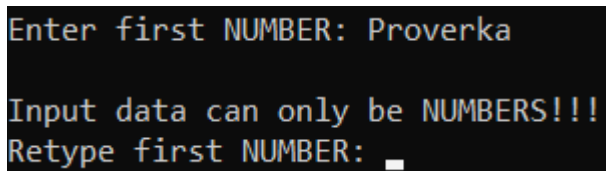
При запуске программы на экран выводится сообщение, о названии приложения, его авторе, и о том, что нужно ввести первое число (рисунок 1).



```
Calculator App (by Dmitriy Minko)
Enter first NUMBER: _
```

Рисунок 1 – Главный экран при запуске программы

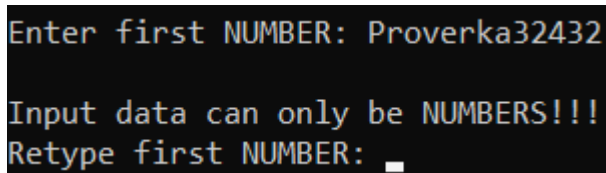
От пользователя требуется ввести целое число. В случае ввода значения в формате “символы”, программа попросит ввести число заново (рисунок 2).



```
Enter first NUMBER: Proverka
Input data can only be NUMBERS!!!
Retype first NUMBER: _
```

Рисунок 2 – Ввод значения в формате “символы”

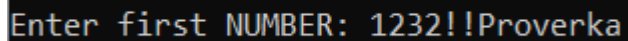
В случае – “символы” + “число”, программа попросит ввести число заново (рисунок 3).



```
Enter first NUMBER: Proverka32432
Input data can only be NUMBERS!!!
Retype first NUMBER: _
```

Рисунок 3 – Ввод значения в формате “символы” + “число”

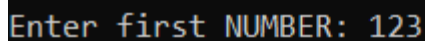
В случае – “число” + “символы” (рисунок 4), программа запишет значение “числа” и перейдёт к следующему этапу (в данном случае запятая и точки тоже считаются символами).



```
Enter first NUMBER: 1232!!Proverka
```

Рисунок 4 – Ввод значения в формате “число” + “символы”

В случае – “число” (рисунок 5), программа запишет значение введённого числа и перейдёт к следующему этапу.

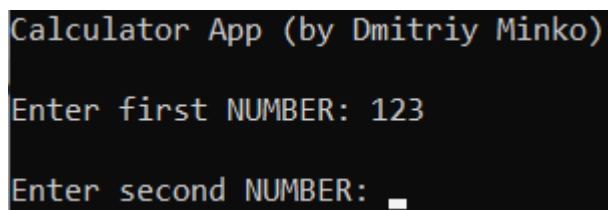


```
Enter first NUMBER: 123
```

Рисунок 5 – Ввод значения в формате “число”

2. Ввод второго числа.

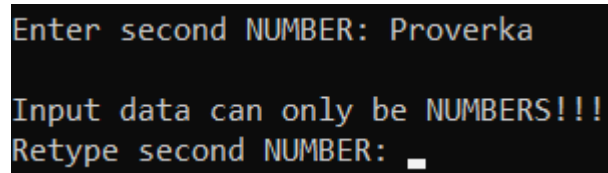
После успешного ввода первого числа, программа запросит ввести второе число (рисунок 6).



```
Calculator App (by Dmitriy Minko)
Enter first NUMBER: 123
Enter second NUMBER: _
```

Рисунок 6 – Запрос на ввод второго числа

От пользователя требуется ввести целое число. В случае ввода значения в формате “символы”, программа попросит ввести число заново (рисунок 7).



```
Enter second NUMBER: Proverka
Input data can only be NUMBERS!!!
Retype second NUMBER: _
```

Рисунок 7 – Ввод значения в формате “символы”

В случае – “символы” + “число”, программа попросит ввести число заново (рисунок 8).


```
Enter second NUMBER: Proverka123
Input data can only be NUMBERS!!!
Retype second NUMBER: _
```

Рисунок 8 – Ввод значения в формате “символы” + “число”

В случае – “число” + “символы” (рисунок 9), программа запишет “число” и перейдёт к следующему этапу (в данном случае запятая и точки тоже считаются символами).

```
Enter second NUMBER: 12,21
```

Рисунок 9 – Ввод значения в формате “число” + “символы”

В случае – “число” (рисунок 10), программа запишет значение введённого числа и перейдёт к следующему этапу.

```
Enter second NUMBER: 11
```

Рисунок 10 – Ввод значения в формате “число”

3. Вывод ранее введённых значений (рисунок 11).

```
First number = 123
Second number = 11
```

Рисунок 11 – Вывод ранее введённых значений

4. Ввод операции.

Вывод списка арифметических операций доступных пользователю с присвоением каждой из них своей цифры (рисунок 12).

```
List of valid arithmetic operations:
1) "+" - Addition
2) "-" - Subtraction
3) "*" - Multiplication
4) "/" - Division
```

Рисунок 12 – Вывод списка доступных арифметических операций

Запрос программой ввести число, которой соответствует арифметическая операция (рисунок 13).

```
Enter NUMBER of the operation you want to perform on numbers: _
```

Рисунок 13 – Запрос ввода числа соответствующего арифметической операции

От пользователя требуется ввести целое число, которой соответствует арифметическая операция. В случае ввода значения в формате “символы”, программа попросит ввести число заново (рисунок 14).

```
Enter NUMBER of the operation you want to perform on numbers: Proverka  
Input data can only be NUMBERS!!!  
Retype operation NUMBER: _
```

Рисунок 14 – Ввод значения в формате “символы”

В случае – “символы” + “число”, программа попросит ввести число заново (рисунок 15).

```
Enter NUMBER of the operation you want to perform on numbers: Proverka1  
Input data can only be NUMBERS!!!  
Retype operation NUMBER: _
```

Рисунок 15 – Ввод значения в формате “символы” + “число”

В случае – “число” (которому не соответствует арифметическая операция) + “символы”, программа попросит ввести число заново (рисунок 16).

```
Enter NUMBER of the operation you want to perform on numbers: 12Proverka  
You can only enter numbers CORRESPONDING to arithmetic operations!!!  
Retype operation NUMBER:
```

Рисунок 16 – Ввод значения в формате “число” (несоответствующее арифметической операции) + “символы”

В случае – “число” (которому не соответствует арифметическая операция), программа попросит ввести число заново (рисунок 17).

```
Enter NUMBER of the operation you want to perform on numbers: 5
You can only enter numbers CORRESPONDING to arithmetic operations!!!
Retype operation NUMBER: _
```

Рисунок 17 – Ввод значения в формате “число” (несоответствующее арифметической операции)

В случае – “число” (которому соответствует арифметическая операция) + “символы” (рисунок 18), программа запишет значение “числа” и перейдёт к следующему этапу (в данном случае запятая и точки тоже считаются символами).

```
Enter NUMBER of the operation you want to perform on numbers: 1Proverka
```

Рисунок 18 – Ввод значения в формате “число” (соответствующее арифметической операции) + “символы”

В случае – “число” (которому соответствует арифметическая операция), программа запишет значение введённого числа и выведет итоговое выражение и его значение (рисунок 19).

```
Enter NUMBER of the operation you want to perform on numbers: 3
123 * 11 = 1353
```

Рисунок 19 – Ввод значения в формате “число” (соответствующее арифметической операции)

5. Итоговый вид запуска программы (рисунок 20).

```
Calculator App (by Dmitriy Minko)

Enter first NUMBER: 123

Enter second NUMBER: 11

First number = 123
Second number = 11

List of valid arithmetic operations:
1) "+" - Addition
2) "-" - Subtraction
3) "*" - Multiplication
4) "/" - Division

Enter NUMBER of the operation you want to perform on numbers: 3

123 * 11 = 1353
Process returned 0 (0x0)   execution time : 4.016 s
Press any key to continue.
```

Рисунок 20 – Итоговое окно успешно выполненной программы

Дополнительная информация:

В случае, если арифметической операцией является деление, а вторым введенным значением – “0”, то программа выдаст ошибку – “ERROR!!! Division by zero is not possible, run the program again.”, попросит запустить программу заново и завершит свою работу с возвращённым значением “2” (Рисунок 21).

```
Calculator App (by Dmitriy Minko)

Enter first NUMBER: 123

Enter second NUMBER: 0

First number = 123
Second number = 0

List of valid arithmetic operations:
1) "+" - Addition
2) "-" - Subtraction
3) "*" - Multiplication
4) "/" - Division

Enter NUMBER of the operation you want to perform on numbers: 4

ERROR!!! Division by zero is not possible, run the program again.
Division ERROR!

Process returned 2 (0x2)   execution time : 3.690 s
Press any key to continue.
```

Рисунок 21 – Окно вывода программы при делении на ноль

В случае если в исходном коде, в реализованных функциях, указатель будет ссылаться на несуществующую область памяти, то программа выведет в консоль "Pointer ERROR!" завершит свою работу с возвращённым значением "3" (рисунок 22).

```
Calculator App (by Dmitry Minko)

Enter first NUMBER: 123

Enter second NUMBER: 11

First number = 123
Second number = 11

List of valid arithmetic operations:
1) "+" - Addition
2) "-" - Subtraction
3) "*" - Multiplication
4) "/" - Division

Enter NUMBER of the operation you want to perform on numbers: 1
Pointer ERROR!
Addition ERROR!

Process returned 3 (0x3)   execution time : 3.731 s
Press any key to continue.
```

Рисунок 22 – Окно вывода программы при ошибке указателя

В случае если в исходном коде аргумент функции `printf()`, будет ссылаться на несуществующую область памяти, то программа выведет в консоль “Printf ERROR!” и завершит свою работу с возвращённым ненулевым значение (рисунок 23).

```
Calculator App (by Dmitry Minko)

Enter first NUMBER: 123

Printf ERROR!

Process returned 4 (0x4)   execution time : 1.681 s
Press any key to continue.
```

Рисунок 23 – Окно вывода программы при ошибке функции вывода в
консоль

ЗАКЛЮЧЕНИЕ

Во время выполнения данной курсовой работы была выполнена поставленная цель, а именно, было написано приложение на языке C, которое выполняет простые расчёты с использованием основных арифметических операций (сложение, вычитание, умножение, деление). По мере достижения цели также достигнуты поставленные задачи – создана блок-схема описывающая алгоритм работы данной программы, написано руководство пользователя, реализованы функции используемых арифметических действий, реализованы проверки аргументов реализованных функций, реализованы проверки возвращаемых функциями значений для всех функций, реализована корректная обработка ошибок, которые могут возникать в результате выполнения программы.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Белецкий Я. Энциклопедия языка Си: Пер. с польск.– М.: Мир, 1992.– 687 с.,ил. ISBN 5-03-002113-2
2. Керниган Б., Ритчи. Д. Язык программирования С, 2-е изд.: Пер. с англ. – СПб.: ООО “Диалектика”, 2020.– 288 с.: ил. – Парал. тит. англ. ISBN 978-5-907144-14-9 (рус.)
3. Развитие языков программирования // CYBERLENINKA URL: <https://cyberleninka.ru/article/n/razvitie-yazykov-programmirovaniya> (дата обращения: 20.12.2023).
4. Подбельский В. В., Фомин С. С. Программирование на языке Си: Учеб. пособие. – 2-е доп. изд. – М.: Финансы и статистика, 2005. 600 с.: ил. ISBN 5-279-02180-6

ПРИЛОЖЕНИЕ 1. РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ

Программа “Калькулятор”.

Программа последовательно считывает с клавиатуры два числа и арифметическую операцию, а затем выводит результат арифметической операции произведённых с введёнными числами. Числа рекомендуется вводить последовательно через клавишу “Enter”, для упрощения работы с программой. Перед каждым действием пользователя программа выводит инструкцию для следующего действия требуемого от пользователя.

Минимальные требования:

- Процессор: 1 гигагерц (ГГц) или быстрее с двумя и более ядрами на совместимом 64-разрядном процессоре или системе на микросхеме (SOC);
- ОЗУ: 4 гигабайта (ГБ);
- Требуемое свободное место на жёстком диске: 58 Кб;
- Служба хранилища: 64 Гб или большее хранилище;
- Программное обеспечение системы: UEFI (для единого extensible Firmware Interface, современной версии PC BIOS) и Secure Boot;
- TPM: доверенный модуль платформы (TPM) версии 2.0;
- Видеокарты: совместим с DirectX 12 или более поздней версии с драйвером WDDM 2.0;
- Отображения: дисплей высокой четкости (720p), который больше 9” по диагонали, 8 бит на цветной канал;
- Операционная система: Windows 11;
- Среда разработки, например, Code::Blocks последней версии.

1. Установка программы.

Установка не требуется, нужно запустить исполняемый файл Calculator.exe, скопировав его в любую директорию.

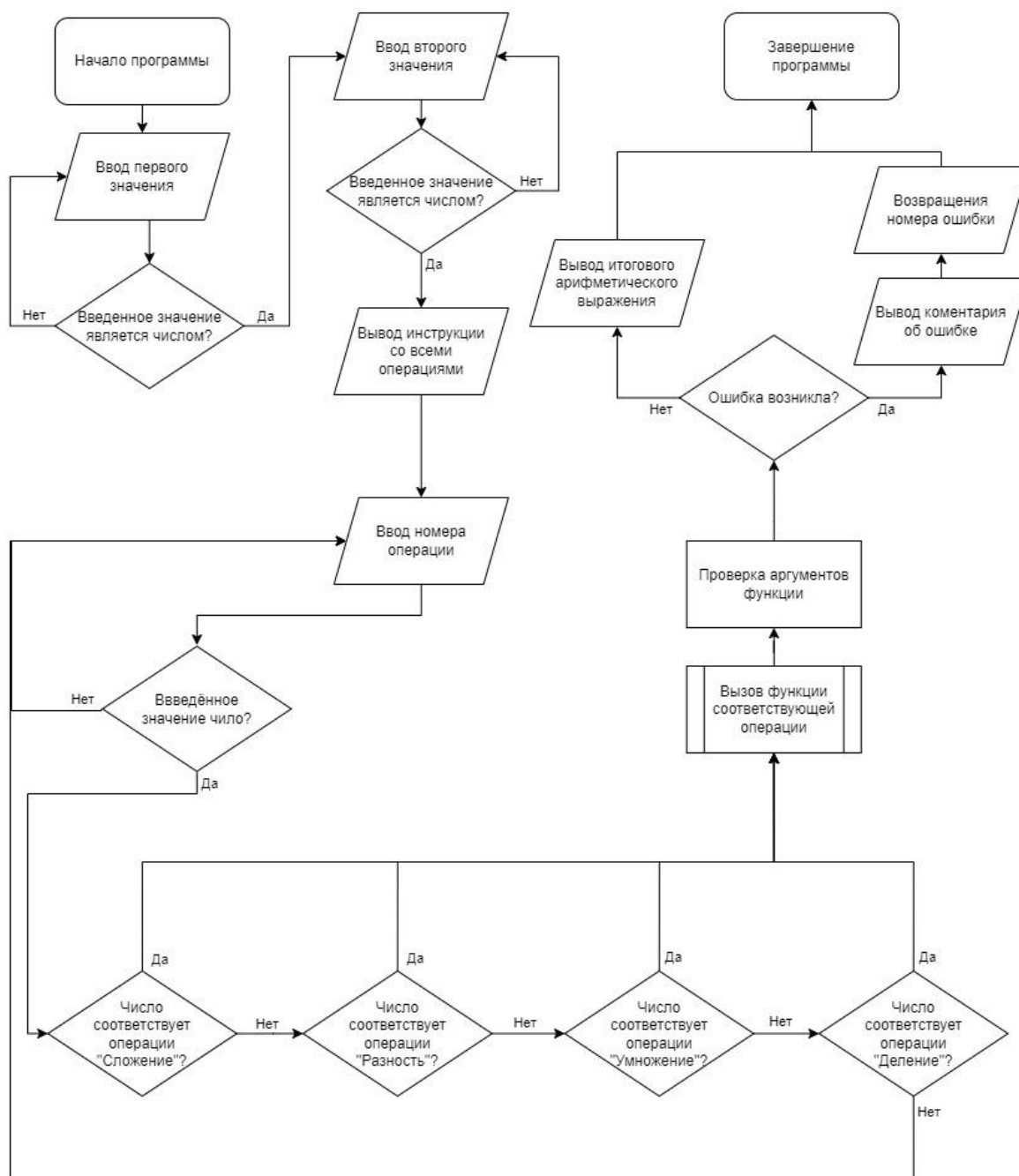
2. Запуск программы.

Запустите файл Calculator.exe.

3. Работа с программой.

Программа выведет название программы, после запросит пользователя ввести с клавиатуры первое значение в формате числа. Далее программа попросит ввести второе значение в формате числа. После успешного выполнения двух предыдущих действий будет выведен список арифметических операций присутствующих в данном калькуляторе. Пользователю требуется ввести номер соответствующей операции, которую он хочет применить к введённым ранее значениям. После этого на экране появится итоговое выражение и его результат. На протяжении всего времени пользования программой в рабочую консоль выводятся подробные инструкции на английском языке. Ввод каждого числа должен завершаться клавишей “Enter”. В случае ввода значений не поддерживаемых форматов программа запросит ввести значение повторно. В случае возникновения ошибки внутри программы, она выдаст номер ошибки и досрочно завершит работу. Программа поддерживает ввод только целочисленных значений. Успешность выполнения программы можно посмотреть в строке `return` “число”. Если оно равно “0” программа выполнена успешно, в ином случае программа завершилась с ошибкой, в этом случае внимательно прочитайте выведенный в консоли комментарий к этой ошибке. В случае несоблюдения правил “Руководства пользователя”, пользователь берёт полную ответственность за результат выполнения программы на себя.

ПРИЛОЖЕНИЕ 2. БЛОК-СХЕМА АЛГОРИТМА



ПРИЛОЖЕНИЕ 3. ИСХОДНЫЙ КОД ПРОГРАММЫ

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

//Деление
int GetDivision(int FirstNumber, int SecondNumber, int
*Value)
{
    if (Value == NULL)
    {
        printf("Pointer ERROR!\n");
        return 3;
    }
    else
    {
        if (SecondNumber == 0)
        {
            printf("\nERROR!!! Division by zero is not
possible, run the program again.\n");
            return 2;
        }
        else
        {
            *Value = FirstNumber / SecondNumber;
        }
    }
    return 0;
}

//Разность
int GetSubtraction(int FirstNumber, int SecondNumber, int
*Value)
{
    if (Value == NULL)
    {
        printf("Pointer ERROR!\n");
        return 3;
    }
    else
    {
        *Value = FirstNumber - SecondNumber;
    }
    return 0;
}

//Умножение
int GetMultiplication(int FirstNumber, int SecondNumber, int
*Value)
{

```

```

        if (Value == NULL)
        {
            printf("Pointer ERROR!\n");
            return 3;
        }
        else
        {
            *Value = FirstNumber * SecondNumber;
        }
        return 0;
    }

    //CyMma
    int GetAddition(int FirstNumber, int SecondNumber, int
*Value)
    {
        if (Value == NULL)
        {
            printf("Pointer ERROR!\n");
            return 3;
        }
        else
        {
            *Value = FirstNumber + SecondNumber;
        }
        return 0;
    }

    int main()
    {
        int first;
        int second;
        int fun;
        int Result;
        int c;
        int Results;
        int Flag = 0;
        int FlagExit;
        int des;
        //Ввод первого числа
        do
        {
            if (Flag == 0) //Флаг для определения была ли уже
выведена первая подсказка
            {
                Result = printf("Calculator App (by Dmitry
Minko)\n\nEnter first NUMBER: ");
                if (Result < 0)
                {
                    printf("\nPrintf ERROR!\n");
                    return 1;
                }
                Flag = 1;
            }
        } while (Flag == 0);
    }

```

```

    }
    else
    {
        Result = printf("\nInput data can only be
NUMBERS!!!\nRetype first NUMBER: ");
        if (Result < 0)
        {
            printf("\nPrintf ERROR!\n");
            return 2;
        }
    }

    Results = scanf("%d", &first);
    if (Results != 1) while (((c = getchar()) != EOF) &&
(c != '\n'));

    }
    while (Results != 1);
    while (((c = getchar()) != EOF) && (c != '\n'));
//Удаление символов, которые введены после числа

    //Ввод второго числа
    Flag = 0; //Обнуления флага для использование в
следующих частях кода
    do
    {
        if (Flag == 0) //Флаг для определения была ли уже
выведена первая подсказка
        {
            Result = printf("\nEnter second NUMBER: ");
            if (Result < 0)
            {
                printf("\nPrintf ERROR!\n");
                return 3;
            }
            Flag = 1;
        }
        else
        {
            Result = printf("\nInput data can only be
NUMBERS!!!\nRetype second NUMBER: ");
            if (Result < 0)
            {
                printf("\nPrintf ERROR!\n");
                return 4;
            }
        }

        Results = scanf("%d", &second);
        if (Results != 1) while (((c = getchar()) != EOF) &&
(c != '\n'));

```

```

    }
    while (Results != 1);

    while (((c = getchar()) != EOF) && (c != '\n'));
    //Удаление символов, которые введены после числа

    //Инструкция к операциям
    Result = printf("\nFirst number = %d\n",first);
    if (Result < 0)
    {
        printf("\nPrintf ERROR!\n");
        return 5;
    }

    Result = printf("Second number = %d\n\n",second);
    if (Result < 0)
    {
        printf("\nPrintf ERROR!\n");
        return 6;
    }

    Result = printf("List of valid arithmetic
operations:\n1) \"+\" - Addition\n2) \"-\" - Subtraction\n3)
\"*\" - Multiplication\n4) \"/\" - Division\n\nEnter NUMBER of
the operation you want to perform on numbers: ");
    if (Result < 0)
    {
        printf("\nPrintf ERROR!\n");
        return 7;
    }

    //Ввод операции
    Flag = 0; //Обнуления флага для использование в
следующих частях кода
    do
    {
        FlagExit = 0;
        if ((Flag == 1) && (Results == 1))
        {
            Result = printf("\nYou can only enter numbers
CORRESPONDING to arithmetic operations!!!\nRetype operation
NUMBER: ");
            if (Result < 0)
            {
                printf("\nPrintf ERROR!\n");
                return 8;
            }
            while (((c = getchar()) != EOF) && (c != '\n'));
            //Для избежание ошибки вывода двух подсказок в случае ввода
строки в которой содержатся сначала цифры, а потом другие
символы
        }
        Results = scanf("%d",&des);

```

```

        if (Results != 1)
        {
            while (((c = getchar()) != EOF) && (c != '\n'));

            Result = printf("\nInput data can only be
NUMBERS!!!\nRetype operation NUMBER: ");
            if (Result < 0)
            {
                printf("\nPrintf ERROR!\n");
                return 9;
            }
        }

        if (des == 1)
        {
            Result = GetAddition(first, second, &fun);
            if (Result != 0)
            {
                printf("Addition ERROR!\n");
                return Result;
            }
            Result = printf("\n%d + %d =
%d\n", first, second, fun);
            if (Result < 0)
            {
                printf("\nPrintf ERROR!\n");
                return 10;
            }
        }
        else if (des == 2)
        {
            Result = GetSubtraction(first, second, &fun);
            if (Result != 0)
            {
                printf("Subtraction ERROR!\n");
                return Result;
            }

            Result = printf("\n%d - %d =
%d\n", first, second, fun);
            if (Result < 0)
            {
                printf("\nPrintf ERROR!\n");
                return 11;
            }
        }
        else if (des == 3)
        {
            Result = GetMultiplication(first, second, &fun);
            if (Result != 0)
            {

```



```

        printf("Multiplication ERROR!\n");
        return Result;
    }
    Result = printf("\n%d * %d =
%d\n", first, second, fun);
    if (Result < 0)
    {
        printf("\nPrintf ERROR!\n");
        return 12;
    }
}
else if (des == 4)
{
    Result = GetDivision(first, second, &fun);
    if (Result != 0)
    {
        printf("Division ERROR!\n");
        return Result;
    }

    Result = printf("\n%d / %d =
%d\n", first, second, fun);
    if (Result < 0)
    {
        printf("\nPrintf ERROR!\n");
        return 13;
    }
}
else
{
    FlagExit = 1; //Flag для проверки есть ли
арифметическая операция, принадлежащая введённому числу
    Flag = 1; //Flag для проверки начальных подсказок
}

}
while ((Results != 1) || (FlagExit != 0));

return 0;
}

```