



Государственное бюджетное образовательное учреждение высшего образования
Московской области

ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ

Колледж космического машиностроения и технологии

КУРСОВОЙ ПРОЕКТ

По дисциплине «Прикладное программирование»

Тема: **«Разработка абстрактного типа данных “Строка”»**

Выполнил студент:

Швыркин Анатолий Павлович

Группа П2-17

_____ (подпись)

_____ (Дата сдачи работы)

Проверил преподаватель:

Гусятинер Леонид Борисович

_____ (подпись)

_____ (оценка)

Оглавление

Введение.....	3
Глава 1. Теоретическая часть.....	4
1.1 Предметная область.....	4
1.2. Описание существующих разработок.....	6
Глава 2. Практическая часть.....	10
2.1. Построение диаграммы прецедентов.....	10
2.2. Выбор инструментов.....	11
2.3. Построения сценария.....	13
2.4. Построения диаграммы классов.....	14
2.5. Описание главного модуля.....	15
2.6. Описание спецификация к модулям.....	16
2.7. Описание модулей.....	17
2.8. Описание тестовых наборов модулей.....	18
2.9. Описание применения средств отладки.....	21
2.10. Анализ оптимальности использования памяти.....	22
Глава 3. Эксплуатационная часть.....	23
3.1. Руководство оператора.....	23
Заключение.....	29
Список литературы и интернет – источников.....	30
Приложение 1. Код главного модуля main.cpp.....	31
Приложение 2. Код модуля string.cpp.....	33
Приложение 3. Код заголовочного файла string.h.....	35

Введение.

Целью данного курсового проекта является разработка абстрактного типа данных “Строка”

В первой главе будет рассмотрена предметная область и существующие разработки по данной теме.

Во второй главе будут рассмотрены разработанные модули, инструменты для работы, а также код программы.

В третьей главе будет рассмотрено руководство оператора.

Глава 1. Теоретическая часть.

1.1 Предметная область.

1.1.1 Абстрактный тип данных.

Абстрактный тип данных (АТД) — это математическая модель для типов данных, где тип данных определяется поведением с точки зрения пользовательских данных, а именно в терминах возможных значений, возможных операций над данными этого типа и поведения этих операций.

В программировании АТД зачастую представляет из себя интерфейс, который скрывает реализации типов данных. Программисты работают с АТД через интерфейс, поскольку в будущем реализация может измениться. Этот метод схож с инкапсуляцией в объектно — ориентированном программировании. Преимущество такого подхода является именно сокрытие реализации. Если ввне определён только интерфейс, то пока структуры данных связаны с интерфейсом, все программы, работающие с заданной структурой АТД, будут продолжать работать. Основной задачей при разработке, является минимальное изменение внешнего интерфейса, постепенно улучшая реализации, алгоритмы по быстродействию, надёжности и используемой памяти.

Абстрактные типы данных позволяют достичь модульности ПО и иметь при это несколько альтернативных взаимозаменяемых реализаций отдельного модуля. Также концепция АТД имеет широко применение в ООП, т.к. она дополняет ООП и позволяет уменьшить сложность программ в мире, где требования к ПО быстро меняются.

1.1.2. Тип данных “Строка” в программировании.

В программировании строковый тип — это тип данных, значениями которого является произвольная последовательность (цепочка) символов алфавита. Каждая переменная такого типа (строковая переменная) может быть представлена фиксированным количеством байтов либо иметь произвольную длину. Некоторые языки программирования накладывают

ограничения на максимальную длину строки, но в большинстве языков подобные ограничения отсутствуют. При использовании Unicode каждый символ строкового типа может требовать двух или даже четырёх байтов для своего представления.

Основные проблемы в машинном представлении строкового типа:

- строки могут иметь достаточно существенный размер (до нескольких десятков мегабайтов);
- изменяющийся со временем размер — возникают трудности с добавлением и удалением символов.

Реализация в языках программирования:

- В первых языках программирования вообще не было строкового типа; программист должен был сам строить функции для работы со строками того или другого типа.
- В Си используются нуль-терминированные строки с полным ручным контролем со стороны программиста.
- В стандартном Паскале строка выглядит как массив из 256 байтов; первый байт хранил длину строки, в остальных хранится её тело. Таким образом, длина строки не может превышать 255 символов. В Borland Pascal 7.0 также появились строки «а-ля Си» — очевидно, из-за того, что в число поддерживаемых платформ вошла Windows.
- В Object Pascal и C++ STL строка является «чёрным ящиком», в котором выделение/высвобождение памяти происходит автоматически — без участия программиста. При создании строки память выделяется автоматически; как только на строку не останется ни одной ссылки, память возвращается системе. Преимущество этого метода в том, что программист не задумывается над работой строк. С другой стороны, программист имеет недостаточный контроль над работой программы в критичных к скорости участках; также трудно

реализуется передача таких строк в качестве параметра в DLL. Также Object Pascal автоматически следит, чтобы в конце строки был символ с кодом 0.

- В C# и других языках со сборкой мусора строка является неизменяемым объектом; если строку нужно модифицировать, создаётся другой объект. Этот метод медленный и расходует немало временной памяти, но хорошо сочетается с концепцией сборки мусора. Преимущество этого метода в том, что присваивание происходит быстро и без дублирования строк.

1.2. Описание существующих разработок.

В этом разделе будут рассмотрены уже существующие абстрактные типы данных.

1. Очередь - это набор данных, организованный таким образом, что вставка нового элемента производится только с конечной ячейки, а удаление - только с начальной. Очередь работает по принципу FIFO (First In - First Out), что означает “Первый зашёл – первый вышел”. Например как на кассе в магазине.

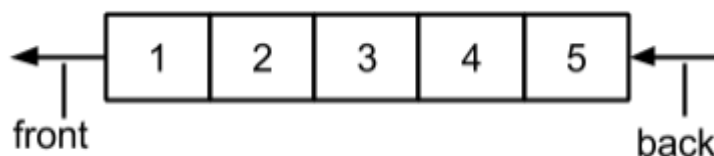


Рисунок 1. Представление АТД "Очередь"

Очередь может быть ограниченного и неограниченного размера в зависимости от задачи.

Очередь предполагает наличие следующих операций:

- `CLEAR(queue)` - делает очередь пустой;
- `IS_EMPTY(queue) : bool` - определяет является ли очередь пустой;

- `IS_FULL(queue) : bool` - определяет является ли очередь полностью заполненной (что имеет смысл только для очередей ограниченного размера);
- `PUSH (queue , value)` - помещает новое значение в конец очереди;
- `POP (queue)` - удаляет значение с начала очереди;
- `FRONT (queue) : value` - возвращает значение в начале очереди.

2. Стек - представляет собой набор данных, доступ, вставка и удаление элементов из которого может происходить только с конца. Стек организовывается по принципу LIFO (Last In - First Out), что означает "Последний вошёл – Первый вышел". В пример можно привести стопку тарелок, тарелка сверху является последней в стеке, но она будет первым элементом вышедшим из стека "тарелок", в то время как первая тарелка попавшая в стек будет являться последним элементом и будет доступна только тогда, когда снимут все тарелки.

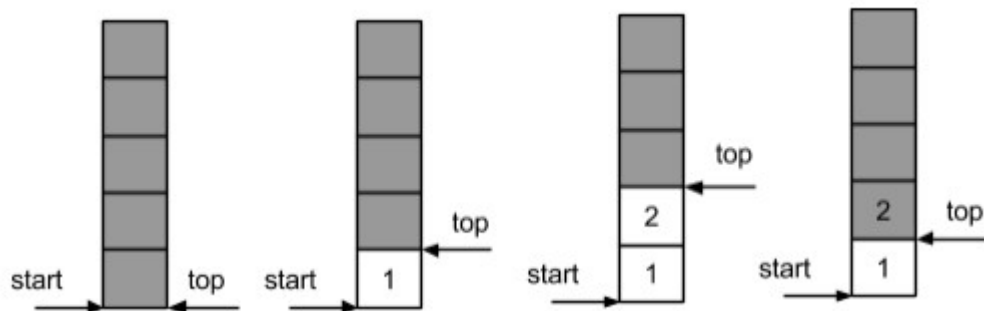


Рисунок 2. Представление АТД "Стек".

В стеке такой же набор операций как и в очереди.

Стеки предполагают наличие следующего набора абстрактных операций:

- `CLEAR(stack)` - делает стек пустым;
- `IS_EMPTY(stack) : bool` - определяет является ли стек пустым;
- `PUSH (stack, value)` - помещает новое значение на вершину стека;
- `POP (stack)` - удаляет значение с вершины стека;

- TOP (stack) : value - возвращает значение на вершине стека;

Если стек имеет ограниченный размер, к перечисленному выше набору операций добавляется:

- IS_FULL(stack) : bool - определяет является ли стек полностью заполненным

3. Список - является наиболее распространенным в программировании абстрактным типом данных. Список представляет собой упорядоченную последовательность значений некоторого типа элементов. Длина списка - это количество содержащихся в нем элементов. Список, длина которого равна 0, называется пустым.

Списки предоставляют свободный доступ к своим элементам, различаемым по позиции. Список имеет начальную и конечную позицию. Позиция может быть также недействительной, что применяется для ситуаций, когда требуется указать на ее некорректность или отсутствие в списке. Какое именно данное используется в качестве позиции в конкретной реализации АД на абстрактном уровне не играет роли, достаточно обеспечить возможность доступа к элементам через позиции, а также операции сравнения значений, играющих роль позиции.

К операциям, определяемым на модели списков, относятся:

- CLEAR(list) - делает список пустым;
- IS_EMPTY(list) : bool - определяет является ли список пустым;
- GET_LENGTH(list) : int - возвращает длину списка;
- GET_NEXT(list, pos) : pos - возвращает следующую позицию после указанной;
- GET_PREV(list, pos) : pos - возвращает предыдущую позицию после указанной;
- GET_FIRST(list) : pos - возвращает первую позицию в списке;

- GET_LAST(list) : pos - возвращает позицию в списке, следующую за последней;
- INSERT (list, pos, value) - вставляет значение в указанную позицию в списке, перемещая все элементы от позиции и далее в следующую более “высокую” позицию;
- RETREIVE (list, pos) : value - возвращает хранящееся в списке значение по указанной позиции;
- DELETE(list, pos) - удаляет из списка элемент, хранящийся по указанной позиции.

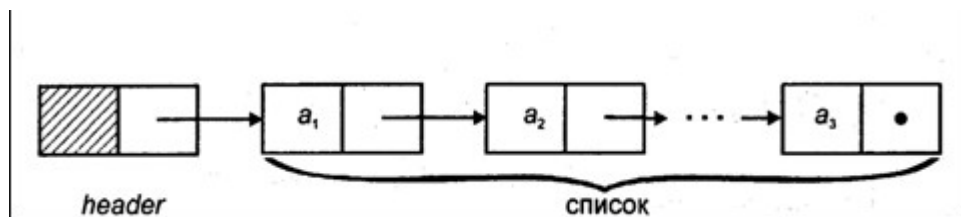
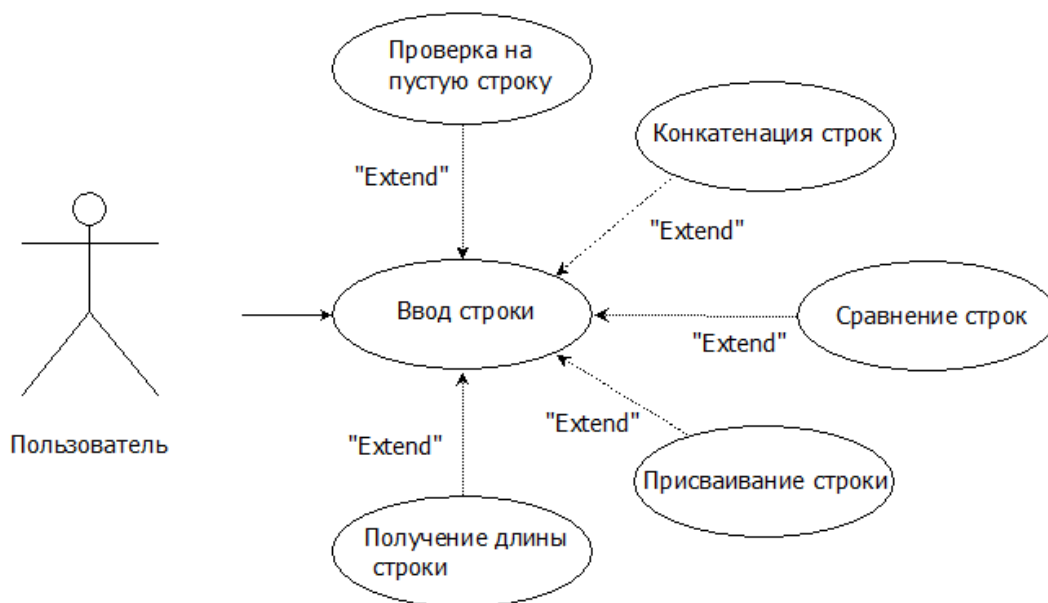


Рисунок 3. Представление АТД "Очередь".

Глава 2. Практическая часть.

2.1. Построение диаграммы прецедентов

Для определения вариантов использования программы к проекту была построена диаграмма прецедентов.



2.2. Выбор инструментов.

В этом разделе будут рассмотрены выбранные инструмент и язык программирование для написания проекта. При выборе инструмента и языка было проведено сравнение по критериям, представленным ниже.

Таблица 1. Критерий выбора инструмента.

Критерий	Важность критерия
Функционал	Ниже высокой
Удобство использования	Высокая
Скорость разработки	Ниже высокой

При сравнении было выделено 3 языка. Сравнение происходило по 10 бальной шкале.

Таблица 2. Оценка языков программирования.

Критерии/ Язык программирования	C++	C	Python
Функционал	9	7	10
Удобство использования	8	5	8
Скорость разработки	8	6	5
Итог	25	18	23

C++ — компилируемый, статически типизированный язык программирования общего назначения. Поддерживает такие парадигмы программирования, как процедурное программирование, объектно-ориентированное программирование, обобщённое программирование. Язык имеет богатую стандартную библиотеку, которая включает в себя распространённые контейнеры и алгоритмы, ввод-вывод, регулярные выражения, поддержку многопоточности и другие возможности.

Таблица 3. Критерии выбора среды разработки.

Критерий	Важность критерия
Простота	Средняя
Функционал	Высокая
Удобство использования	Высокая
Документация на русском языке	Средняя

По данным критериям я сравнил две среды разработки по 10 бальной шкале.

Таблица 4. Оценка сред разработки.

Критерий / Среда разработки	CodeBlocks	Microsoft Visual Studio
Простота	6	5
Функционал	7	10
Удобство использования	8	6
Документация на русском языке	8	6
Итого	29	27

CodeBlocks — свободная кроссплатформенная среда разработки. Написана на C++ и использует библиотеку wxWidgets. Имея открытую архитектуру, может масштабироваться за счёт подключаемых модулей.

2.3. Построения сценария.

В этом разделе будет рассмотрен сценарий использования программы.

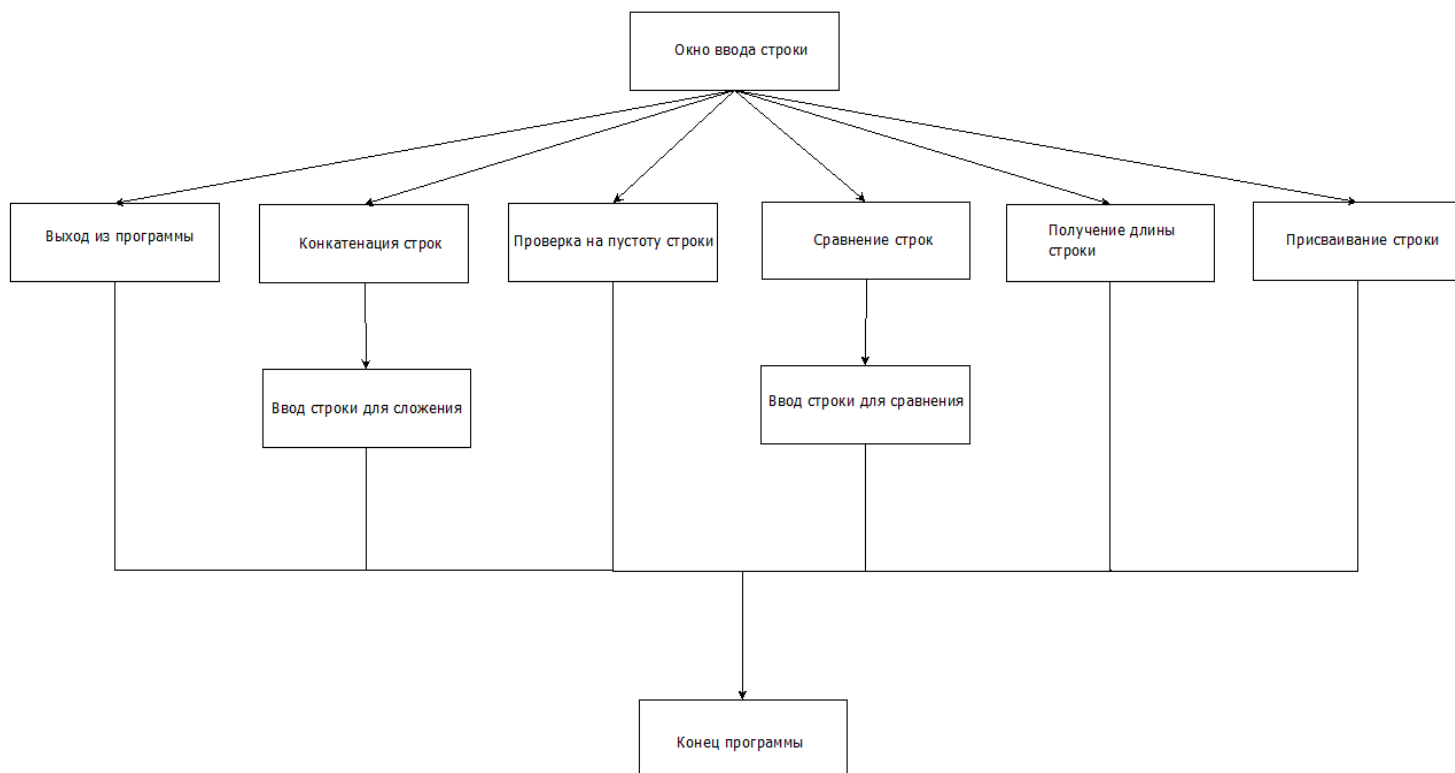


Рисунок 5. Построение сценария.

2.4. Построения диаграммы классов.

В этом разделе представлена диаграмма классов.

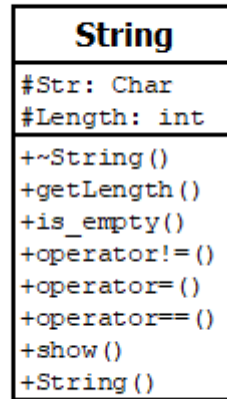


Рисунок 6. Диаграмма классов.

2.5. Описание главного модуля.

В этом разделе представлен главный модуль `main.cpp` к которому подключается модуль содержащий в себе класс и методы.

Листинг 1. Код главного модуля `main.cpp`

```
#include "string.cpp"

using namespace std;

// Функция для отображения меню
int menu(String &s)
{
    int choice;
    cout << "\n1) Проверка на пустую строку" << endl;
    cout << "2) Конкатенация строк" << endl;
    cout << "3) Сравнение строк" << endl;
    cout << "4) Приравнять строку" << endl;
    cout << "5) Получить длину строки" << endl;
    cout << "6) Выход" << endl;
    cin >> choice;
    // Очитска буфера
    cin.clear();
    cin.ignore(32767, '\n');
    // Меню выбора действий со строкой
    // Проверка на пустоту
    switch(choice)
    {
        case 1:
            cout << (s.is_empty() == 0 ? "Не пустая" : "Пустая") << endl;
            break;
        // Конкатенация строк
        case 2:
        {
            String f;
            cout << "Введите строку с которой хотите сложить: ";
            cin >> f;
            s += f;
            cout << s;
            break;
        }
        // Сравнение строк
        case 3:
        {
            cout << "Введите строку с которой хотите сравнить: ";
            String s2;
            cin >> s2;
            cout << (s == s2 ? "Равны" : "Не равны") << endl;
            break;
        }
        // Присваивание строки
        case 4:
        {
            cout << "Введите строку к которой хотите присвоить: ";
```

```

        String s2;
        cin >> s2;
        s = s2;
        cout << s;
        break;
    }
    // Вывод длины строки
    case 5:
        cout << s.getLength() << endl;
        break;
    // Выход из программы
    case 6:
        return 0;
    default:
        menu(s);
        break;
    }
    menu(s);
}
// Главное тело программы
int main()
{
    setlocale(LC_ALL, "Russian");
    String s;
    cout << "Введите строку: ";
    cin >> s;
    menu(s);
    return 0;
}

```

2.6. Описание спецификация к модулям.

В этом разделе описаны спецификации к модулям проекта.

Листинг 3. Модуль string.h

```

#pragma once
#include <iostream>
#include <cstring>

using namespace std;

class String
{
protected:
    char* Str; // массив для строки
    int Length; // длина строки
public:
    String(); // конструктор с объявлением строки
    String& operator = (String& t); // перегрузка операции =
    String& operator += (const String& t); // перегрузка операции +=
    bool operator == (const String& t) const; // перегрузка операции ==
    bool operator != (const String& t) const; // перегрузка операции !=
    bool is_empty() const; // проверка на пустоту строки
    const char* getStr() const; // вывод строки
    int getLength() const; // получение длины строки

```



```

        ostream& show(ostream& os); // вывод строки
        friend ostream& operator << (ostream& os, String& s) // перегрузка //
операции <<
        {
            return s.show(os);
        }
        friend istream& operator >> (istream& is, String& s) // перегрузка //
операции >>
        {
            char buf[10001];
            is.getline(buf, 10001);
            if (s.Length != 0) {
                delete[] s.Str;
            }
            s.Length = strlen(buf);
            s.Str = new char[s.Length + 1];
            strncpy(s.Str, buf, s.Length);
            s.Str[s.Length] = '\0';
            return is;
        }

        ~String();
};

```

2.7. Описание модулей

В данном разделе описаны используемые модули.

Программа состоит из трёх файлов.

1. Главный модуль.

Представляет собой файл main.cpp. К главному модулю подключается модуль string.cpp, который содержит в себе функции.

2. Модуль string.cpp.

- Конструктор класса String
- Конкатенация строк
- Проверка на равенство
- Проверка на неравенство
- Проверка на пустоту
- Деструктор

3. Заголовочный файл string.h

Файл в котором описаны класс, его поля, перегрузка операций, а так же дружественные функции.

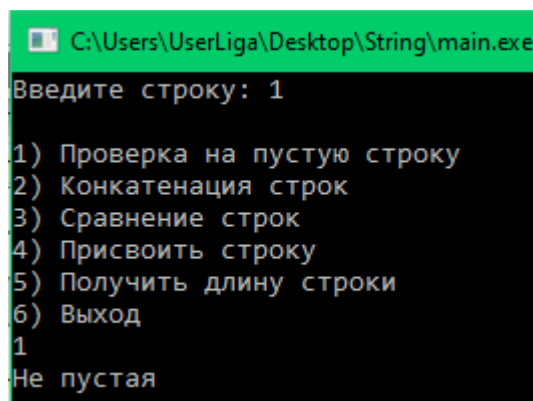
2.8. Описание тестовых наборов модулей.

В этом разделе будут продемонстрированы результаты тестирования “чёрного ящика”.

Тест 1. Проверка пустоты строки.

Действия: ввести строку, в меню выбора операции выбрать 1(Проверка на пустую строку).

Ожидаемый результат: программа должна вывести “Пустая” или “Не пустая” строка в зависимости от того, какой был пользовательский ввод.



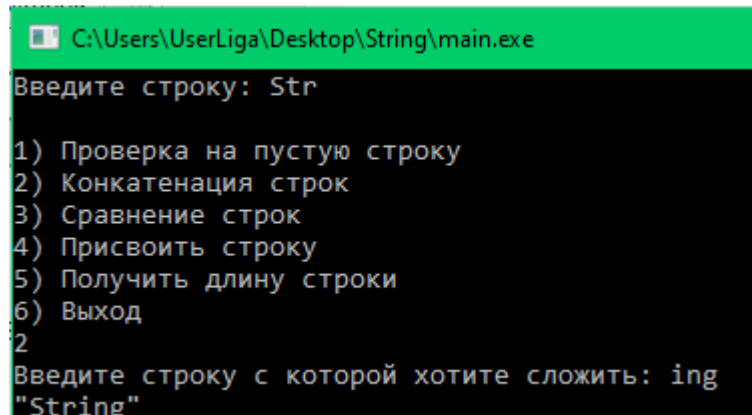
```
C:\Users\UserLiga\Desktop\String\main.exe
Введите строку: 1
1) Проверка на пустую строку
2) Конкатенация строк
3) Сравнение строк
4) Присвоить строку
5) Получить длину строки
6) Выход
1
Не пустая
```

Рисунок 7. Проверка пустая строка или нет.

Тест 2. Конкатенация строк.

Действия: ввести строку, в меню выбора операций нажать 2(Конкатенация строк), ввести вторую строку.

Ожидаемый результат: программа выведет сложит две строки и выведет их в новой.



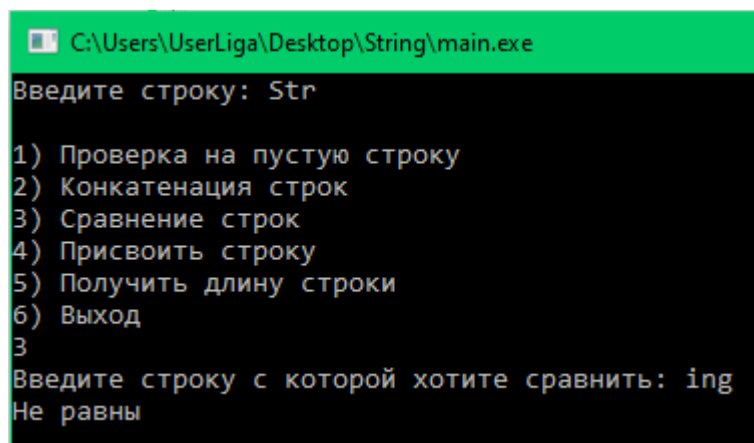
```
C:\Users\UserLiga\Desktop\String\main.exe
Введите строку: Str
1) Проверка на пустую строку
2) Конкатенация строк
3) Сравнение строк
4) Присвоить строку
5) Получить длину строки
6) Выход
2
Введите строку с которой хотите сложить: ing
"String"
```

Рисунок 8. Конкатенация строк.

Тест 3. Сравнение строк.

Действия: ввести строку, в меню выбора операции выбрать 3(Сравнение строк), ввести вторую строку.

Ожидаемый результат: программа должна выдать сообщение о том, равны строки или нет.



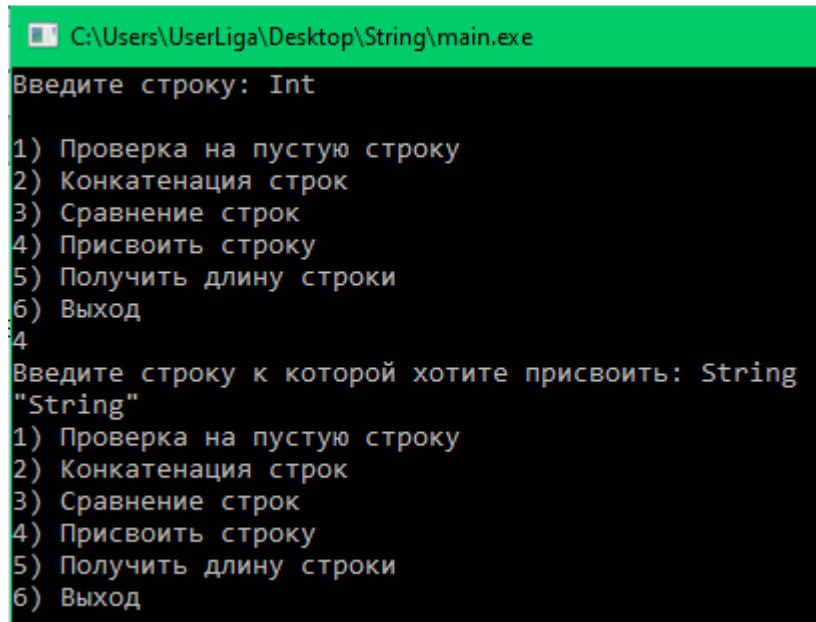
```
C:\Users\UserLiga\Desktop\String\main.exe
Введите строку: Str
1) Проверка на пустую строку
2) Конкатенация строк
3) Сравнение строк
4) Присвоить строку
5) Получить длину строки
6) Выход
3
Введите строку с которой хотите сравнить: ing
Не равны
```

Рисунок 9. Сравнение строк.

Тест 4. Присваивание строки.

Действия: ввести строку, в меню выбора операции выбрать 4(Присвоить строку), введите строку к которой хотите присвоить первую.

Ожидаемый результат: программа выведет первую строку с содержанием второй.



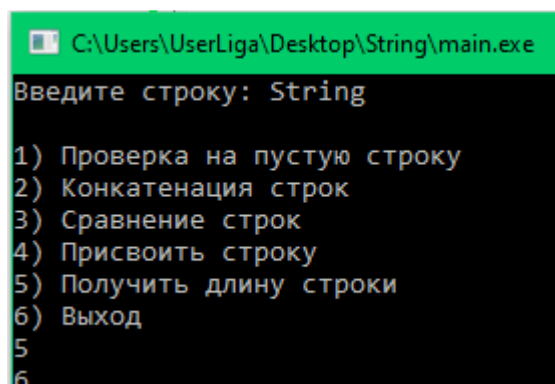
```
C:\Users\UserLiga\Desktop\String\main.exe
Введите строку: Int
1) Проверка на пустую строку
2) Конкатенация строк
3) Сравнение строк
4) Присвоить строку
5) Получить длину строки
6) Выход
4
Введите строку к которой хотите присвоить: String
"String"
1) Проверка на пустую строку
2) Конкатенация строк
3) Сравнение строк
4) Присвоить строку
5) Получить длину строки
6) Выход
```

Рисунок 10. Присваивание строки.

Тест 5. Отображение длины строки.

Действия: ввести строку, в меню выбора операций выбрать 5(Получить длину строки).

Ожидаемый результат: программа выдаст длину строки.



```
C:\Users\UserLiga\Desktop\String\main.exe
Введите строку: String
1) Проверка на пустую строку
2) Конкатенация строк
3) Сравнение строк
4) Присвоить строку
5) Получить длину строки
6) Выход
5
Введите строку к которой хотите присвоить: String
"String"
1) Проверка на пустую строку
2) Конкатенация строк
3) Сравнение строк
4) Присвоить строку
5) Получить длину строки
6) Выход
```

Рисунок 11. Длина строки.

2.9. Описание применения средств отладки.

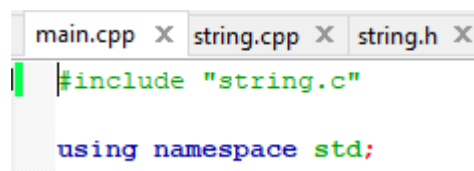
В ходе написания курсового проекта были допущены следующие сообщение:

File	Line	Message
		=== Build file: "no target" in "no project" (compiler: unknown) ===
C:\Users\User...	1	fatal error: string.c: No such file or directory
		=== Build failed: 1 error(s), 0 warning(s) (0 minute(s), 0 second(s)) ===

Рисунок 12. Обнаружение ошибки.

Суть ошибки заключается в том, что компилятор не может найти файл который надо подключить для того, чтобы программа работала.

Для решения этой проблемы нужно перейти в файл main.cpp и обратить внимание на первую строчку(рис. 12).



```
main.cpp X string.cpp X string.h X
#include "string.c"
using namespace std;
```

Рисунок 13. Ошибка в файле main.cpp

Необходимо

изменить расширение подключаемого файла string.c на string.cpp, чтобы компилятор смог подключить этот файл для работы программы.

Результатом отладки стало следующие сообщение:

File	Line	Message
		=== Build file: "no target" in "no project" (compiler: unknown) ===
C:\Users\User...		In function 'int menu(String&)':
C:\Users\User...	62	warning: control reaches end of non-void function [-Wreturn-type]
		=== Build finished: 0 error(s), 1 warning(s) (0 minute(s), 0 second(s)) ===

Рисунок 14. Результат отладки программы.

2.10. Анализ оптимальности использования памяти и быстродействия.

В данном разделе представлен анализ оптимальности использования памяти и быстродействия программы.

Было принято решение разбить проект на три файла, чтобы упростить процесс отладки, чтения и дальнейшей модернизации кода.

Глава 3. Эксплуатационная часть.

3.1. Руководство оператора.

1. Назначение программы.

Функциональное назначение программы.

Основной функцией проекта String является работа со строкой.

Эксплуатационное назначение программы.

Программа String может быть использована в качестве обработки строк.

Состав функций.

- Функции пользователя:
- Проверить пустоту строки
- Конкатенация строк
- Сравнение строк
- Присвоить строку
- Получить длину строки

2. Условия выполнения программы.

Минимальные состав аппаратных средств.

Минимальный требования к используемым техническим средствам:

- ОС – Windows 7, 8.1, 10, Linux
- Процессор с тактовой частотой не ниже 1,6 ГГц.
- ОЗУ – 10 МБ
- Место на жестком диске: 8 КБ+
- Видеоадаптер с минимальным разрешением 360p

Минимальный состав программных средств.

Интегрированная среда разработки CodeBlock.

Требования к пользователю.

Базовые навыки работы с компьютером.

3. Выполнение программы.

Загрузка и запуск программы.

Программа состоит из трёх файлов: “main.cpp”, “string.cpp”, “string.h”. Все эти файлы для корректной работы программы должны находиться в одном каталоге

Чтобы запустить программу, потребуется:

- Запустить IDE CodeBlocks.
- В левом верхнем углу в контекстном меню “File” выбрать функцию “Open”
- Открыть файлы “main.cpp”, “string.cpp”, “string.h”

Далее программу нужно собрать и запустить.

Для этого нужно:

- В верхнем меню инструментов найти кнопку “Build”

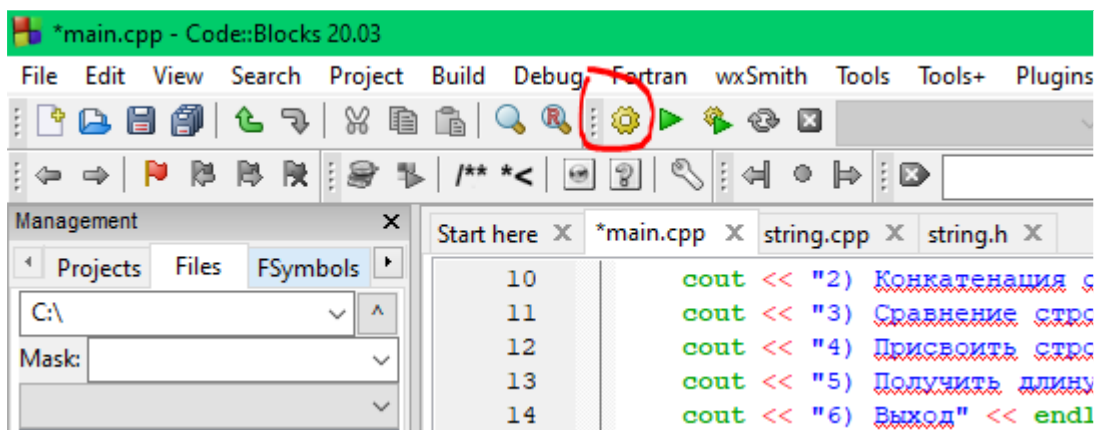


Рисунок 15. Сборка программы.

Если не было выявлено никаких ошибок, компилятор сообщит вам об успешной сборке программы

- Далее нужно нажать кнопку “Run”

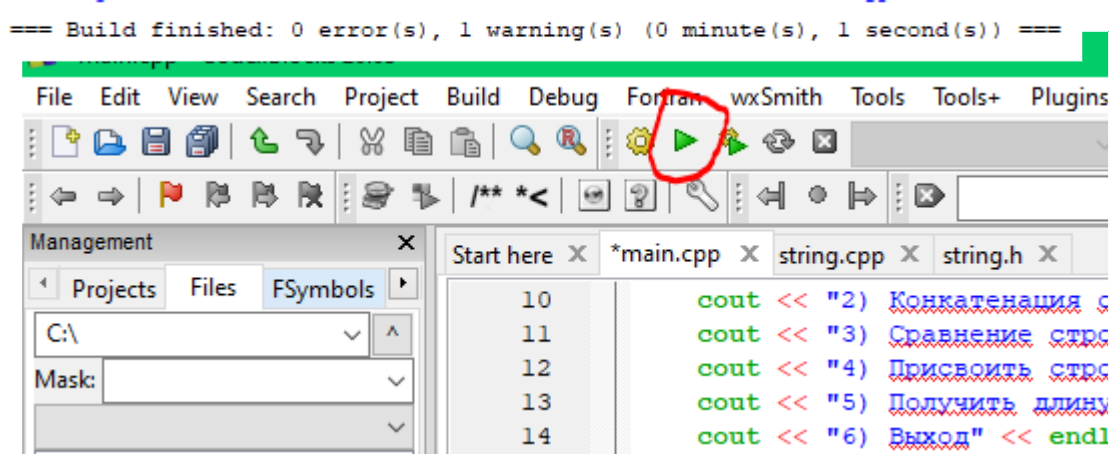


Рисунок 17. Запуск программы.

В итоге у вас должен запускаться исполняемый файл с расширением .exe

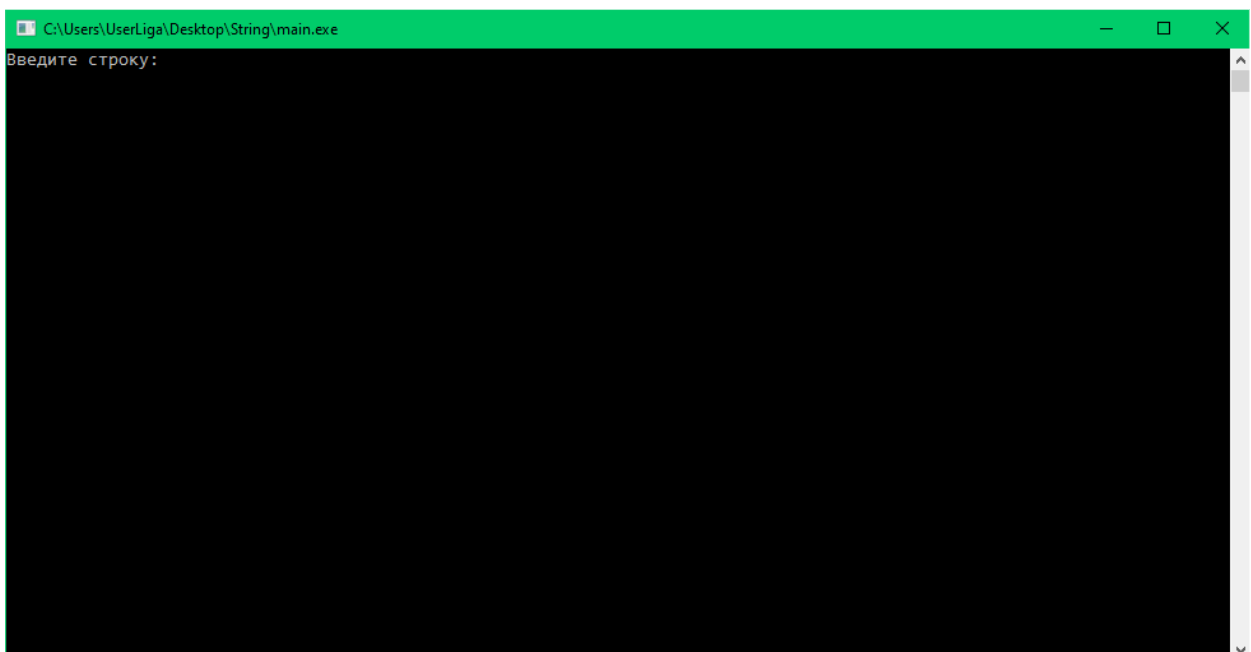


Рисунок 18. Успешный запуск программы.

Выполнение программы.

Для выбора одной из предложенных операций, оператору нужно ввести строку, с которой он хочет работать.

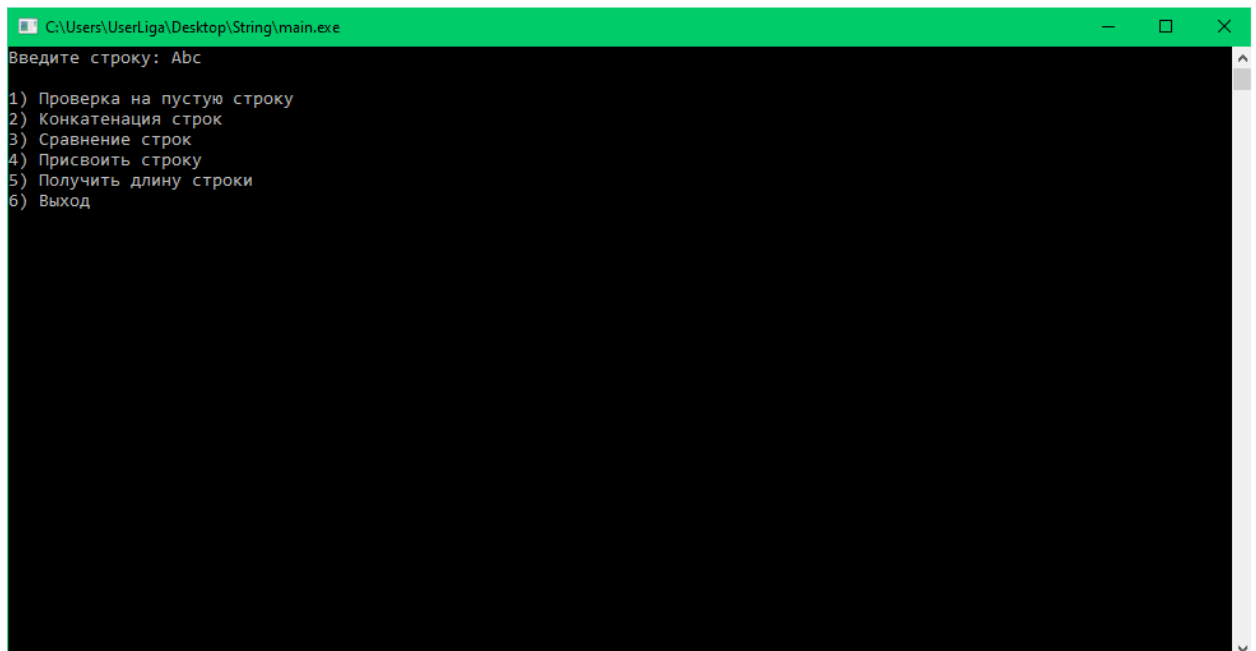


Рисунок 19. Доступ к меню программы.

Далее будут рассмотрены операции, которые может использовать оператор.

Для выбора операции нужно ввести ей соответствующую цифру.

1. Проверка на пустую строку.

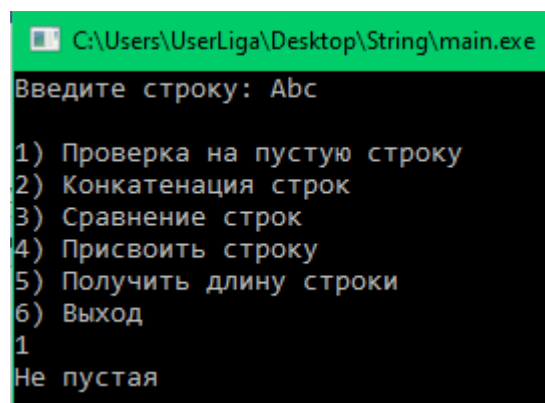
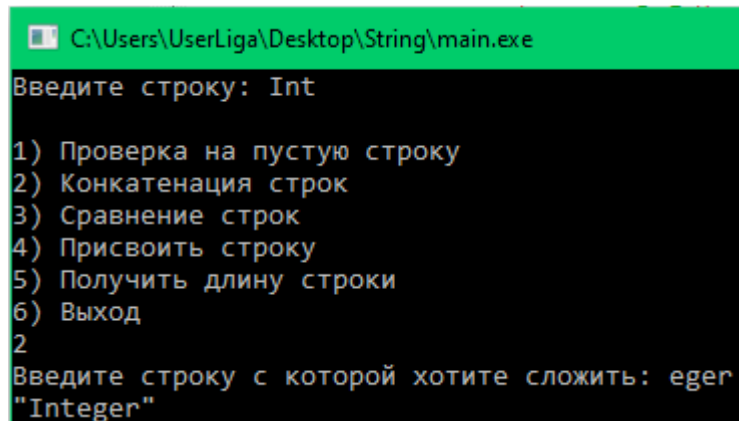


Рисунок 20. Операция проверки пустая строка или нет

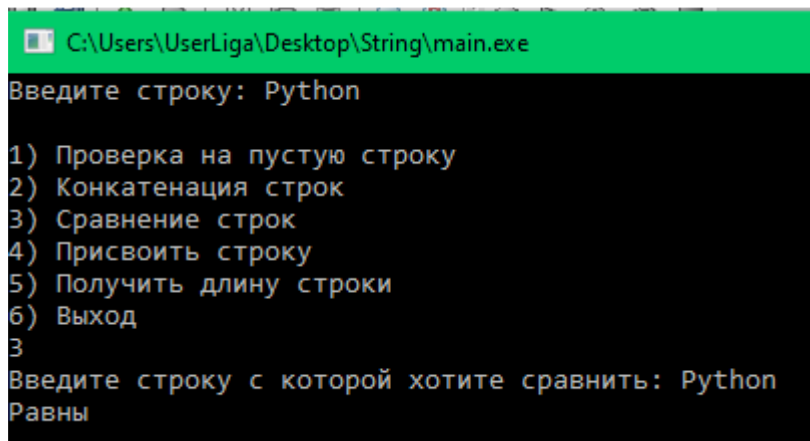
2. Конкатенация строк.



```
C:\Users\UserLiga\Desktop\String\main.exe
Введите строку: Int
1) Проверка на пустую строку
2) Конкатенация строк
3) Сравнение строк
4) Присвоить строку
5) Получить длину строки
6) Выход
2
Введите строку с которой хотите сложить: eger
"Integer"
```

Рисунок 21. Операция сложения строк.

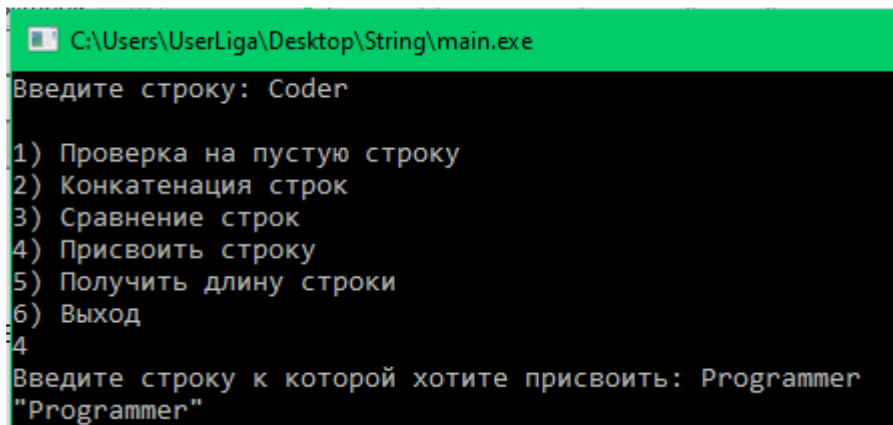
3. Сравнение строк.



```
C:\Users\UserLiga\Desktop\String\main.exe
Введите строку: Python
1) Проверка на пустую строку
2) Конкатенация строк
3) Сравнение строк
4) Присвоить строку
5) Получить длину строки
6) Выход
3
Введите строку с которой хотите сравнить: Python
Равны
```

Рисунок 22. Операция сравнения строк.

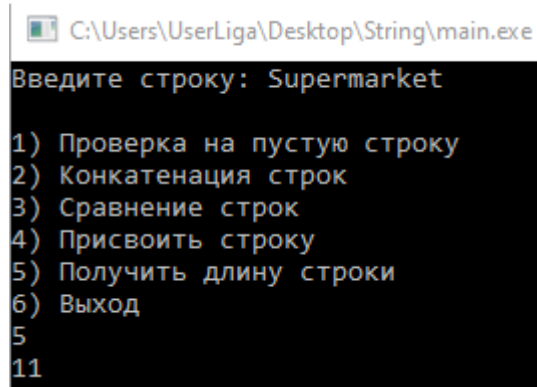
4. Присваивание строки.



```
C:\Users\UserLiga\Desktop\String\main.exe
Введите строку: Coder
1) Проверка на пустую строку
2) Конкатенация строк
3) Сравнение строк
4) Присвоить строку
5) Получить длину строки
6) Выход
4
Введите строку к которой хотите присвоить: Programmer
"Programmer"
```

Рисунок 23. Операция присваивания строк

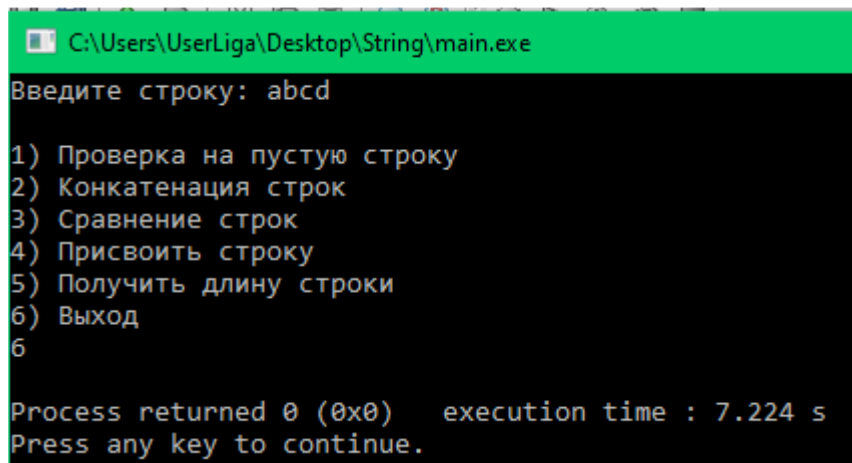
5. Длина строки.



```
C:\Users\UserLiga\Desktop\String\main.exe
Введите строку: Supermarket
1) Проверка на пустую строку
2) Конкатенация строк
3) Сравнение строк
4) Присвоить строку
5) Получить длину строки
6) Выход
5
11
```

Рисунок 24. Операция получения длины строки.

6. Выход из программы.



```
C:\Users\UserLiga\Desktop\String\main.exe
Введите строку: abcd
1) Проверка на пустую строку
2) Конкатенация строк
3) Сравнение строк
4) Присвоить строку
5) Получить длину строки
6) Выход
6
Process returned 0 (0x0) execution time : 7.224 s
Press any key to continue.
```

Рисунок 25. Операция выхода из программы.

Заключение.

В результате выполнения курсового проекта была написана программа "String" для автоматизации процесса покупки валюты.

В ходе работы были проанализированы предметная область, существующие разработки, посвященные данному направлению, получены практические навыки с работой в CodeBlocks

Так же планируется продолжить работу над данным проектом с целью расширения возможностей и удобства для пользователей.

Планы по доработке:

1. Реализация методов и функций, для расширения области работы программы.
2. Создание и последующая доработка интерфейса.
3. Исправление багов.

Список литературы и интернет – источников.

Теоретический материал.

https://ru.wikipedia.org/wiki/Абстрактный_тип_данных

<https://habr.com/ru/post/216725/>

<https://prog-cpp.ru/cpp-atd/>

<https://qna.habr.com/q/565354>

<https://ru.wikipedia.org/wiki/C%2B%2B>

https://ru.wikipedia.org/wiki/Строковый_тип

Приложение 1. Код главного модуля main.cpp

```
#include "string.cpp"

using namespace std;

// Функция для отображения меню
int menu(String &s)
{
    int choice;
    cout << "\n1) Проверка на пустую строку" << endl;
    cout << "2) Конкатенация строк" << endl;
    cout << "3) Сравнение строк" << endl;
    cout << "4) Присвоить строку" << endl;
    cout << "5) Получить длину строки" << endl;
    cout << "6) Выход" << endl;
    cin >> choice;
    // Очитска буфера
    cin.clear();
    cin.ignore(32767, '\n');
    // Меню выбора действий со строкой
    switch(choice)
    {
        case 1:
            cout << (s.is_empty() == 0 ? "Не пустая" : "Пустая") << endl;
            //Проверка на пустоту
            break;
        // Конкатенация строк
        case 2:
        {
            String f;
            cout << "Введите строку с которой хотите сложить: ";
            cin >> f;
            s += f;
            cout << s;
            break;
        }
        // Сравнение строк
        case 3:
        {
            cout << "Введите строку с которой хотите сравнить: ";
            String s2;
            cin >> s2;
            cout << (s == s2 ? "Равны" : "Не равны") << endl;
            break;
        }
        // Присвоение строк
        case 4:
        {
            cout << "Введите строку к которой хотите присвоить: ";
            String s2;
            cin >> s2;
            s = s2;
            cout << s;
        }
    }
}
```



```

        break;
    }
    // Вывод длины строки
    case 5:
        cout << s.getLength() << endl;
        break;
    // Выход из программы
    case 6:
        return 0;
    default:
        menu(s);
        break;
    }
    menu(s);
}
// Главное тело программы
int main()
{
    setlocale(LC_ALL, "Russian");
    String s;
    cout << "Введите строку: ";
    cin >> s;
    menu(s);
    return 0;
}

```

Приложение 2. Код модуля string.cpp

```
#include "string.h"
#include <iostream>
#include <cstring>

using namespace std;

// Конструктор класса String
String::String()
{
    Str = new char[1];
    Str[0] = '\0';
    Length = 0;
}

String& String::operator = (String& t)
{
    // Смена местами значения строк
    swap(Length, t.Length);
    swap(Str, t.Str);
    return *this;
}

String& String::operator += (const String& t)
{
    // Увеличение длины строки
    int newLength = Length + t.Length;
    // Создание новой строки размером длин двух строк
    char* newStr = new char[newLength + 1];
    strcpy(newStr, Str);
    strcat(newStr, t.Str);
    delete[] Str; // очищение исходной строки
    Str = newStr;
    Length = newLength;
    return *this;
}

// Сравнение строк на равенство
bool String::operator == (const String& t) const
{
    return Length == t.Length && strcmp(Str, t.Str) == 0;
}

// Сравнение строк на неравенство
bool String::operator != (const String& t) const
{
    return !(operator == (t));
}

// Проверка строки на пустоту
bool String::is_empty() const
{
    return Str == 0 || Str[0] == '\0';
}

const char* String::getStr() const
```

```

{
    return Str;
}

int String::getLength() const
{
    return Length;
}

ostream& String::show(ostream& os)
{
    return os << "\"" << (Str ? Str : "") << "\""; // Возвращает поток со
строкой
}

// Деструктор - очищает строку и ставит длину строки 0
String::~~String()
{
    Length = 0;
    delete[] Str;
    Str = 0;
}

```

Приложение 3. Код заголовочного файла string.h

```
#pragma once
#include <iostream>
#include <cstring>

using namespace std;

class String
{
protected:
    char* Str; // массив для строки
    int Length; // длина строки
public:
    String(); // конструктор с объявлением строки
    String& operator = (String& t); // перегрузка операции =
    String& operator += (const String& t); // перегрузка операции +=
    bool operator == (const String& t) const; // перегрузка операции ==
    bool operator != (const String& t) const; // перегрузка операции !=
    bool is_empty() const; // проверка на пустоту строки
    const char* getStr() const; // вывод строки
    int getLength() const; // получение длины строки

    ostream& show(ostream& os); // вывод строки
    friend ostream& operator << (ostream& os, String& s) // перегрузка
операции <<
    {
        return s.show(os);
    }
    friend istream& operator >> (istream& is, String& s) // перегрузка
операции >>
    {
        char buf[10001];
        is.getline(buf, 10001);
        if (s.Length != 0) {
            delete[] s.Str;
        }
        s.Length = strlen(buf);
        s.Str = new char[s.Length + 1];
        strncpy(s.Str, buf, s.Length);
        s.Str[s.Length] = '\0';
        return is;
    }

    ~String();
};
```