МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

**«БЕЛГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ
ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ им. В. Г. ШУХОВА»
(БГТУ им. В.Г. Шухова)**

Кафедра программного обеспечения вычислительной техники и автоматизированных систем

Расчетно-графическое задание

по дисциплине: Математическая логика и теория алгоритмов
тема: «Интерпретатор языка Brainfuck **»

Выполнил: ст. группы ПВ-202
Аладиб Язан

Проверил:
Куценко Дмитрий Александрович
Бондаренко Татьяна Владимировна

Белгород 2022 г.

Brainfuck — это необычный язык программирования, разработанный с целью создания полноценного языка Тьюринга с минимально возможным размером компилятора.

Brainfuck не предназначен для практического использования, он состоит всего из восьми команд, что делает создание простых программ сложным процессом, требующим от программистов проверки пределов своих возможностей.

Языки на основе Brainfuck :

символ	Описание
>	**Следующая ячейка**
<	**Предыдущая ячейка**
+	**Значение текущей ячейки увеличивается на 1**
-	**Значение текущей ячейки уменьшается на 1**
.	**Вывод значения текущей ячейки**
,	**Запрос значения текущей ячейки**
[	**Начало цикла (у COW есть особенность - пропускается первая команда цикла)**
]	**Конец цикла**

Функция is_correct_word
Заголовок: `remove_spaces()`:
Назначение: *удалить пробелы*
Листинг:

```
void Brainfuck::remove_spaces() {
    input.erase(std::remove_if(input.begin(), input.end(), ::isspace),
input.end());
}
```

Функция `remove_new_lines`
Заголовок: `remove_new_lines()`
Назначение: *удалить новые строки*
Листинг:

```
void Brainfuck::remove_new_lines() {
    input.erase(std::remove(input.begin(), input.end(), '\n'), input.end());
}
```

Функция `find_matching_end_bracket`
Заголовок: `find_matching_end_bracket()`
Назначение: *найти подходящую концевую скобку*
Листинг:

```
void Brainfuck::find_matching_end_bracket() {
    int expected = 0;
    int found = 0;
    while (current_char < input.length()) {
        if (input[current_char] == '[') expected++;
        if (input[current_char] == ']') {
            found++;
            if (expected == found) break;
        }
        current_char++;
    }
}
```

Функция `find_matching_start_bracket`
Заголовок: `find_matching_start_bracket()`
Назначение: *найти подходящую начальную скобку*
Листинг:

```
void Brainfuck::find_matching_start_bracket() {
    int expected = 0;
    int found = 0;
    while (current_char >= 0) {
        if (input[current_char] == ']') expected++;
        if (input[current_char] == '[') {
            found++;
            if (expected == found) break;
        }
```

```
        current_char--;
    }
}
```

Функция op_codes
Заголовок: op_codes(*const char*& cur, std::string& output)
Назначение: *перемещение указателя зависит от символа*
Листинг:

```
void Brainfuck::op_codes(const char& cur, std::string& output) {
    switch (cur) {
        case '>':
            if (current_index == cells.size() - 1) {
                cells.emplace_back(0);
            } else {
                current_index++;
            }
            break;
        case '<':
            if (current_index == 0) {
                cells.emplace_front(0);
            } else {
                current_index--;
            }
            break;
        case '+':
            cells[current_index]++;
            break;
        case '-':
            cells[current_index]--;
            break;
        case '.':
            if (!ascii) {
                output += std::to_string(cells[current_index]);
            } else {
                output += cells[current_index];
            }
            break;
        case ',': {
            std::string in;
            std::cout << "Введите число от 0 до 255:\n ";
            std::cin >> in;
            if (!in.empty() && std::all_of(in.begin(), in.end(), ::isdigit))
{
                auto c_in = stoi(in);
                if (c_in >= 0 && c_in <= 255) {
                    cells[current_index] = c_in;
                } else {
                    std::cout << "Неверный ввод! Введите число от 0 до 255!
:\n" << std::endl;
                }
            } else {
                std::cout << "Неверный ввод!" << std::endl;
            }
            break;
```

```
        }
        case '[':
            if (cells[current_index] == 0) {
                find_matching_end_bracket();
            }
            break;
        case ']':
            if (cells[current_index] != 0) {
                find_matching_start_bracket();
            }
            break;
        default:
            std::cerr << "Недействительная программа Brainfuck! Неизвестный
символ в " << current_index << std::endl;
            break;
    }
}
```

Функция `execute`
Заголовок: `execute(std::string& output, bool ascii)`
Назначение: *выполнить и вывести*
Листинг:

```
//выполнить и вывести
void Brainfuck::execute(std::string& output, bool ascii) {
    this->ascii = ascii;
    while (current_char < input.length()) {
        op_codes(input[current_char], output);
        current_char++;
    }
}
```

Класс `Brainfuck`
Листинг:

```
class Brainfuck {

    public:
        Brainfuck(const std::string& input);
        void execute(std::string& output, bool ascii);

    private:
        std::deque<std::uint8_t> cells;
        std::string input;
        bool ascii;
        int current_index;
        int current_char;
        void remove_spaces();
        void remove_new_lines();
        void find_matching_end_bracket();
        void find_matching_start_bracket();
    void op_codes(const char& cur, std::string& output);
};
```

файл brainfuck.h :

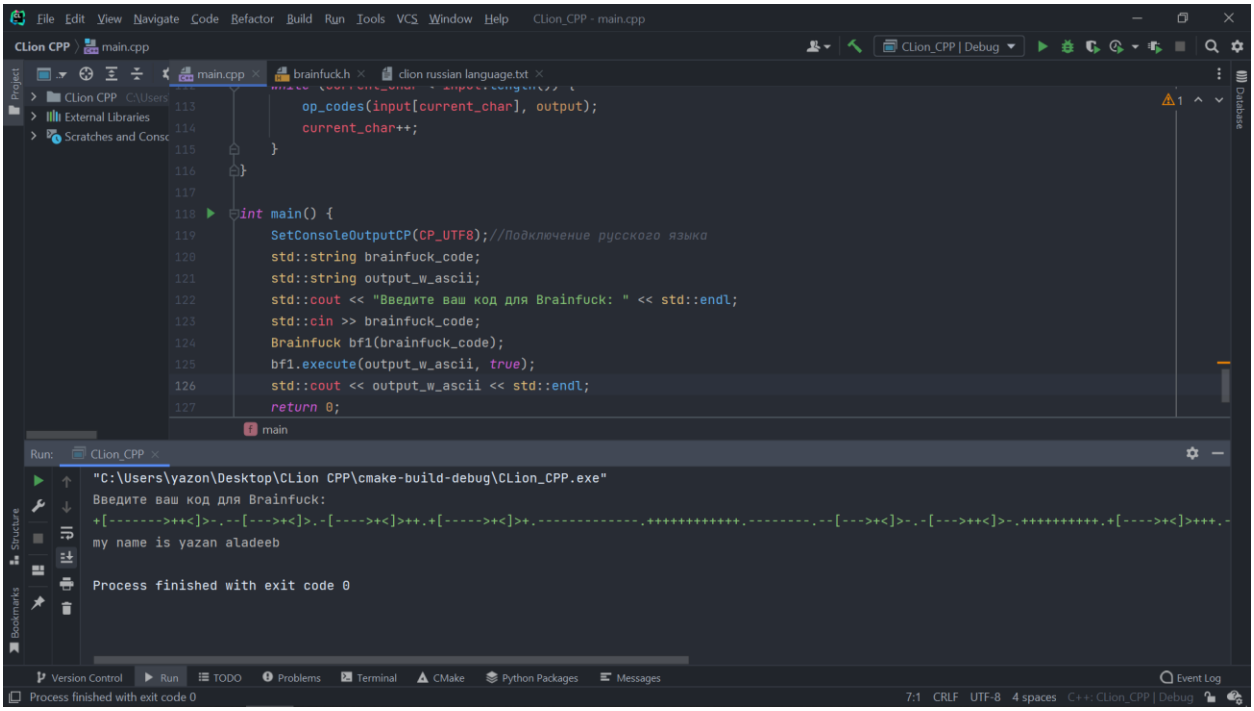
```
#include <deque>
#include <string>

class Brainfuck {

    public:
        Brainfuck(const std::string& input);
        void execute(std::string& output, bool ascii);

    private:
        std::deque<std::uint8_t> cells;
        std::string input;
        bool ascii;
        int current_index;
        int current_char;
        void remove_spaces();
        void remove_new_lines();
        void find_matching_end_bracket();
        void find_matching_start_bracket();
    void op_codes(const char& cur, std::string& output);
};
```

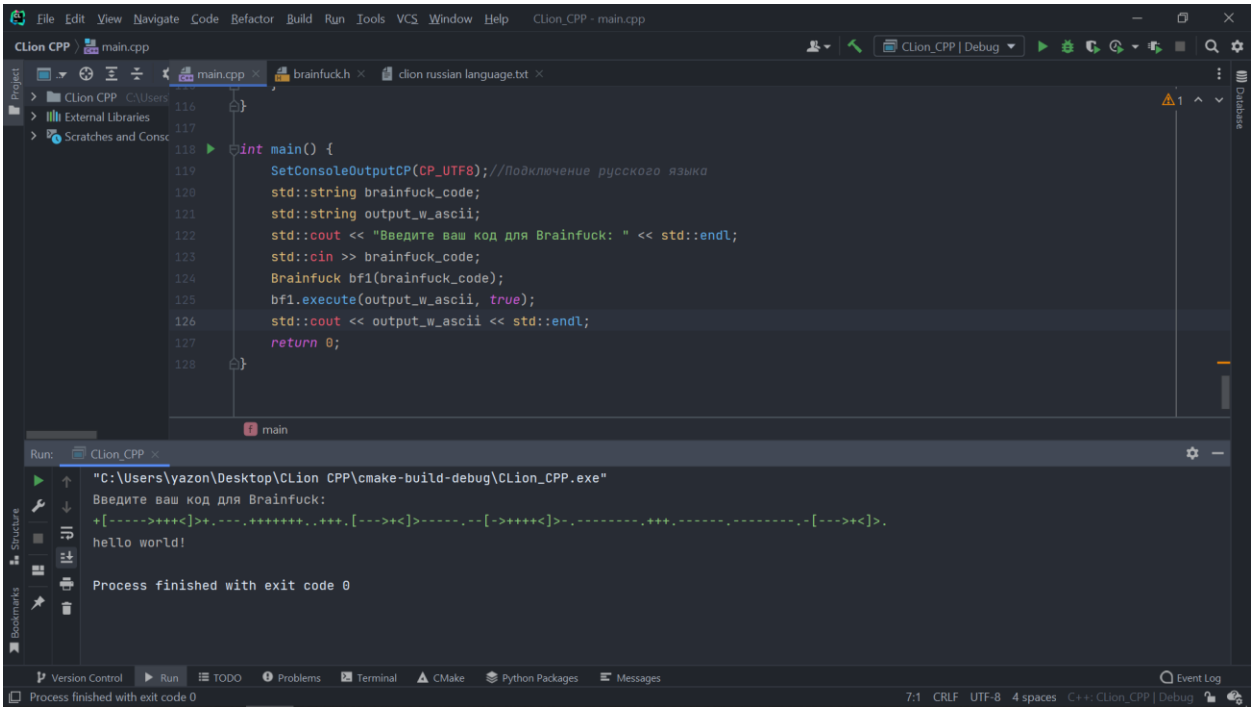
файл main.cpp :

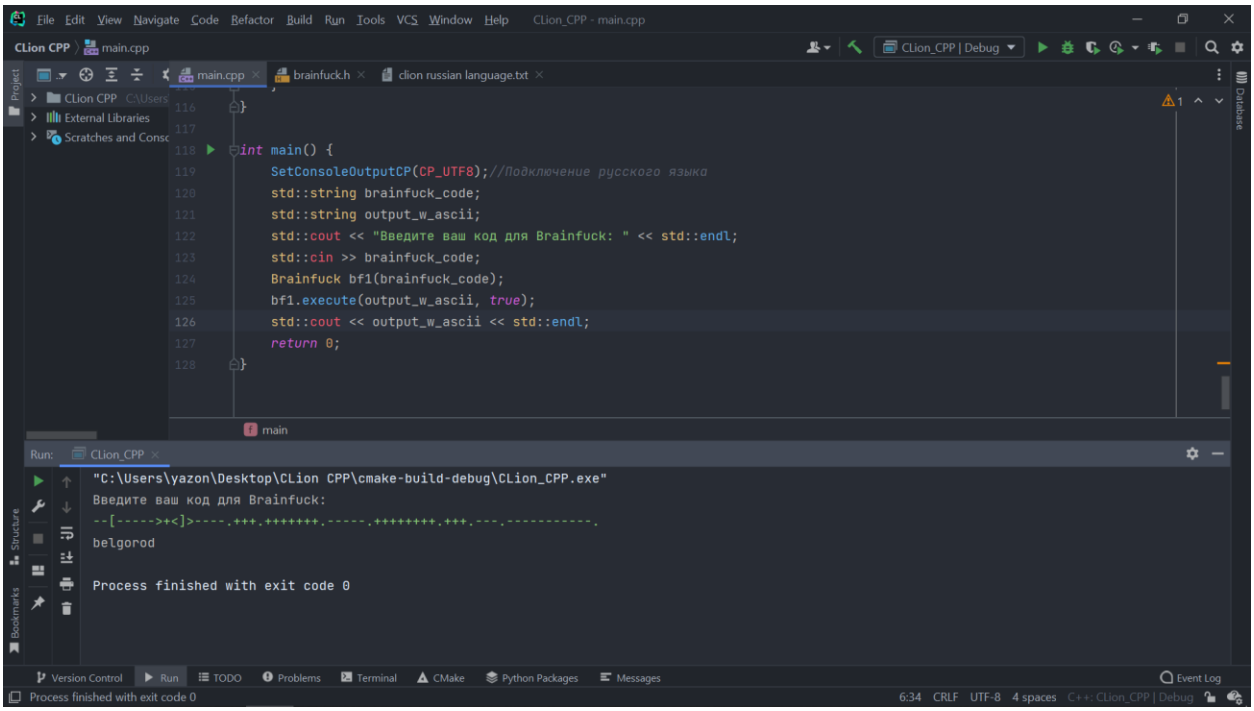
```
#include "brainfuck.h"
#include <algorithm>
#include <iostream>
#include <string>
#include <windows.h>

Brainfuck::Brainfuck(const std::string& input) : input(input) {
    remove_spaces();
    remove_new_lines();
    cells.assign(30'000, 0);
    current_index = 0;
    current_char = 0;
}
//удалить пробелы
void Brainfuck::remove_spaces() {
    input.erase(std::remove_if(input.begin(), input.end(), ::isspace),
input.end());
}
//удалить новые строки
void Brainfuck::remove_new_lines() {
    input.erase(std::remove(input.begin(), input.end(), '\n'), input.end());
}
//найти подходящую концевую скобку
void Brainfuck::find_matching_end_bracket() {
    int expected = 0;
```

```
    int found = 0;
    while (current_char < input.length()) {
        if (input[current_char] == '[') expected++;
        if (input[current_char] == ']') {
            found++;
            if (expected == found) break;
        }
        current_char++;
    }
}
//найти подходящую начальную скобку
void Brainfuck::find_matching_start_bracket() {
    int expected = 0;
    int found = 0;
    while (current_char >= 0) {
        if (input[current_char] == ']') expected++;
        if (input[current_char] == '[') {
            found++;
            if (expected == found) break;
        }
        current_char--;
    }
}
//перемещение указателя зависит от символа
void Brainfuck::op_codes(const char& cur, std::string& output) {
    switch (cur) {
        case '>':
            if (current_index == cells.size() - 1) {
                cells.emplace_back(0);
            } else {
                current_index++;
            }
            break;
        case '<':
            if (current_index == 0) {
                cells.emplace_front(0);
            } else {
                current_index--;
            }
            break;
        case '+':
            cells[current_index]++;
            break;
        case '-':
            cells[current_index]--;
            break;
        case '.':
            if (!ascii) {
                output += std::to_string(cells[current_index]);
            } else {
                output += cells[current_index];
            }
            break;
        case ',': {
            std::string in;
            std::cout << "Введите число от 0 до 255:\n ";
            std::cin >> in;
```

```
            if (!in.empty() && std::all_of(in.begin(), in.end(), ::isdigit))
{
                auto c_in = stoi(in);
                if (c_in >= 0 && c_in <= 255) {
                    cells[current_index] = c_in;
                } else {
                    std::cout << "Неверный ввод! Введите число от 0 до 255!
:\n" << std::endl;
                }
            } else {
                std::cout << "Неверный ввод!" << std::endl;
            }
            break;
        }
        case '[':
            if (cells[current_index] == 0) {
                find_matching_end_bracket();
            }
            break;
        case ']':
            if (cells[current_index] != 0) {
                find_matching_start_bracket();
            }
            break;
        default:
            std::cerr << "Недействительная программа Brainfuck! Неизвестный
символ в " << current_index << std::endl;
            break;
    }
}
//выполнить и вывести
void Brainfuck::execute(std::string& output, bool ascii) {
    this->ascii = ascii;
    while (current_char < input.length()) {
        op_codes(input[current_char], output);
        current_char++;
    }
}

int main(){
    SetConsoleOutputCP(CP_UTF8);//Подключение русского языка
    std::string brainfuck_code;
    std::string output_w_ascii;
    std::cout << "Введите ваш код для Brainfuck: " << std::endl;
    std::cin >> brainfuck_code;
    Brainfuck bf1(brainfuck_code);
    bf1.execute(output_w_ascii, true);
    std::cout << output_w_ascii << std::endl;
    return EXIT_SUCCESS;
}
```


File Edit View Navigate Code Refactor Build Run Tools VCS Window Help CLion_CPP - main.cpp
CLion CPP main.cpp
CLion_CPP | Debug
main.cpp brainfuck.h clion russian language.txt
CLion CPP C:\Users
External Libraries
Scratches and Consc
op_codes(input[current_char], output);
current_char++;
}
}
int main() {
SetConsoleOutputCP(CP_UTF8);//Подключение русского языка
std::string brainfuck_code;
std::string output_w_ascii;
std::cout << "Введите ваш код для Brainfuck: " << std::endl;
std::cin >> brainfuck_code;
Brainfuck bf1(brainfuck_code);
bf1.execute(output_w_ascii, true);
std::cout << output_w_ascii << std::endl;
return 0;
main
Run: CLion_CPP
"C:\Users\yazon\Desktop\CLion CPP\cmake-build-debug\CLion_CPP.exe"
Введите ваш код для Brainfuck:
+[------>++<]>-.--[--->+<]>.-[--->+<]>++.+[----->+<]>+.-------------.++++++++++++.--------.--[--->+<]>-.-[--->++<]>-.++++++++++.+[---->+<]>+++.-
my name is yazan aladeeb
Process finished with exit code 0
Version Control Run TODO Problems Terminal CMake Python Packages Messages Event Log
Process finished with exit code 0
7:1 CRLF UTF-8 4 spaces C++: CLion_CPP | Debug


File Edit View Navigate Code Refactor Build Run Tools VCS Window Help CLion_CPP - main.cpp
CLion CPP main.cpp
CLion_CPP | Debug
main.cpp brainfuck.h clion russian language.txt
CLion CPP C:\Users
External Libraries
Scratches and Consc
}
int main() {
SetConsoleOutputCP(CP_UTF8);//Подключение русского языка
std::string brainfuck_code;
std::string output_w_ascii;
std::cout << "Введите ваш код для Brainfuck: " << std::endl;
std::cin >> brainfuck_code;
Brainfuck bf1(brainfuck_code);
bf1.execute(output_w_ascii, true);
std::cout << output_w_ascii << std::endl;
return 0;
}
main
Run: CLion_CPP
"C:\Users\yazon\Desktop\CLion CPP\cmake-build-debug\CLion_CPP.exe"
Введите ваш код для Brainfuck:
+[----->+++<]>+.---.+++++++..+++.[--->+<]>-----.--[->++++<]>-.--------.+++.------.--------.-[--->+<]>.
hello world!
Process finished with exit code 0
Version Control Run TODO Problems Terminal CMake Python Packages Messages Event Log
Process finished with exit code 0
7:1 CRLF UTF-8 4 spaces C++: CLion_CPP | Debug


File Edit View Navigate Code Refactor Build Run Tools VCS Window Help
CLion_CPP - main.cpp
CLion CPP
main.cpp
CLion_CPP | Debug
main.cpp
brainfuck.h
clion russian language.txt
CLion CPP
External Libraries
Scratches and Consc
}
int main() {
SetConsoleOutputCP(CP_UTF8);//Подключение русского языка
std::string brainfuck_code;
std::string output_w_ascii;
std::cout << "Введите ваш код для Brainfuck: " << std::endl;
std::cin >> brainfuck_code;
Brainfuck bf1(brainfuck_code);
bf1.execute(output_w_ascii, true);
std::cout << output_w_ascii << std::endl;
return 0;
}
main
Run:
CLion_CPP
"C:\Users\yazon\Desktop\CLion CPP\cmake-build-debug\CLion_CPP.exe"
Введите ваш код для Brainfuck:
--[-----><]>----.+++.+++++++.-----.++++++++.+++.---.-----------.
belgorod
Process finished with exit code 0
Version Control
Run
TODO
Problems
Terminal
CMake
Python Packages
Messages
Event Log
Process finished with exit code 0
6:34 CRLF UTF-8 4 spaces C++: CLion_CPP | Debug