**Правительство Российской Федерации**

Федеральное государственное автономное образовательное учреждение высшего образования

«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ

«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»

Кафедра компьютерной безопасности

**Отчёт к лабораторной работе №2 по дисциплине «Языки программирования»**

Работу выполнила студентка группы СКБ221

Е. В. Гриднева

Москва 2022

**Постановка задачи:**

Предстоит написать консольное выражение, которое при вводе флага –forward считает математическое приложение введённое пользователем через std::cin (или из файла при наличии флага --file) напрямую, то есть "5 + 5" и выводит 10.

Иначе, если введён флаг --reverse, то, соответственно, необходимо подсчитать выражение, записанное с помощью ОПН – обратной польской нотации, например, "1 2 + 4 × 3 +" и выводит 15.

# **Алгоритм решения задачи**

Для выполнения поставленной задачи разработан следующий алгоритм:

1. После написания программы, используя компилятор g++ внутри терминала с дополнением WSL, я скомпилировала свой файл командой: g++ lab.cpp -o lab, при этом назвав исполняемый файл “lab”. В дальнейших пунктах будет описана суть кода.
2. Далее необходимо было запустить исполняемый файл с нужными мне флагами командой “./lab …”, где вместо “…” пользователь мог ввести флаги, название файла. Пример в виде ОПЗ или в виде прямой записи, в случае его чтения из терминала, вводится уже после выполнения вышеперечисленной команды.
3. В случае, если пример был введен в виде прямой записи, требовался перевод в ОПЗ.
4. Уже в виде ОПЗ char-массив передавался в функцию подсчёта с учётом приоритетов операндов.
5. Для упрощения работы с терминалом мной был создан Makefile с целями all, clean, tests, lab. Цель «tests» отвечает за автоматические тесты программы на текстовых файлах, «lab» отвечает за автоматическую компиляцию программы, «clean» - за отчистку временных файлов.

# **Выполнение задачи**

1. В функции main (int argc, char\*\* argv) я считала флаги, записала имя файла и, в случае ошибки ввода, выводила сообщение об этом. После данных действий происходила инициализация массива с примером внутрь программы с помощью функции initialization (argv, file, m).
2. В функции initialization (argv, file, m) я учла варианты задания примера. В случае, если пример был введен с помощью файла, я использовала встроенные функции fopen и fclose, а также функцию fgetc для вывода данных из файла во внутренний массив программы. В случае ввода выражения из терминала я использовала функцию std::cin.getline (), встроенную в библиотеку iostream.
3. После инициализации, если введен флаг --forward, я вызывала функцию forward (char\* m), где происходило преобразование прямого вида выражения в ОПЗ. В данной функции я использовала структуру, которая отвечала за стек, в котором хранились все символы из выражения, исключая скобочки. Внутри данной функции использовались функции InChar, отвечающая за внесение в стек знака операции, и OutChar, отвечающие за выборку последнего зашедшего знака операции из стека. Выборка элементов из стека, либо внесение их туда, происходили за счет приоритетов операций, устанавливаемых скобками. Например, в примере 5+5\*7 приоритет у умножения выше, чем у плюса, а в примере (5+5) \* 7 приоритет выше у сложения. За приоритеты отвечала отдельная функция Priority с switch-кейсами внутри нее. После преобразования в ОПЗ вызывала функцию reverse (char \*m).
4. Функция reverse (char \*m) вызывалась и в случае ввода флага --reverse, когда запись выражения уже была в виде ОПЗ. Данная функция отвечала за подсчет того, что записано в виде ОПЗ. Использовала дополнительный массив типа double, чтобы корректно выводился ответ с нецелыми значениями. Он всегда состоит из 2-3 элементов, а именно десятичных чисел, с которыми необходимо провести операцию. В случае, если выражение выглядит как 5 4 + 7 \*, то в массив сначала запишутся по порядку два числа 5 и 4 (их индексы в массиве 0 и 1), а потом произойдет операция сложения, которая запишет свой ответ 9 на место первого из операндов (то есть станет элементом массива с индексом 0). При этом число 7 запишется сразу после 9 (то есть будет с индексом 1). После проведенной операции будет выведен конечный результат.

**Тестирование**

1. С текстовыми файлами

**Изображение выглядит как текст

Автоматически созданное описание**

1. С вводом из терминала

Изображение выглядит как текст

Автоматически созданное описание

**Код программы**

Приложение А

#include <iostream>

#include <math.h>

struct Stack {

char c;

Stack \*next;

};

void reverse(char\* m, int l);

void forward(char \*m);

void initialization(char\*\* argv, int file, char\* m);

int Priority(char c);

Stack \*InChar(Stack \*p, char s);

Stack \*OutChar(Stack \*p, char \*s);

int lenght(char \*m);

int main(int argc, char\*\* argv) {

int file = 0;

int metod = 0;

char m[1024];

int l = 0;

for (int i = 1; i < argc; i++) {

if (argv[i][0] == '-') {

if (argv[i][3] == 'i') {

file = 1;

}

else if (argv[i][2] == 'r') {

metod = 1;

}

else if (argv[i][3] == 'o') {

metod = 2;

}

}

else if (file) {

file = i;

}

}

if (metod == 1) {

initialization(argv, file, m);

l = lenght(m);

std::cout << "Input: " << m << std::endl;

reverse(m, l);

}

else if (metod == 2) {

initialization(argv, file, m);

std::cout << "Input: " << m << std::endl;

forward(m);

}

else {

std::cout << "Input error 1" << std::endl;

}

return 0;

}

void initialization(char\*\* argv, int file, char\* m) {

int j = 0;

if (file != 0) {

FILE\* fp;

if ((fp = fopen(argv[file], "r")) == NULL) {

std::cout << "File doesn't exist" << std::endl;

}

else {

while ((m[j] = fgetc(fp)) != EOF) {

j++;

}

m[j] = '\0';

fclose(fp);

}

} else {

std::cin.getline(m, 1024);

}

}

void forward(char \*m) {

Stack \*t;

Stack \*Op = NULL;

char a, Out[128];

int k = 0, l = 0;

while (m[k] != '\0') {

if (m[k] >= '0' && m[k] <= '9') {

Out[l++] = m[k];

}

else if (m[k] == '(') {

Op = InChar(Op, m[k]);

}

else if (m[k] == ')') {

while ((Op->c) != '(') {

Op = OutChar(Op, &a);

if (Op == NULL) {

a = '\0';

}

Out[l++] = a;

}

t = Op;

Op = Op ->next;

delete t;

}

else if (m[k] == '+' || m[k] == '-' || m[k] == '\*' || m[k] == '/' || m[k] == '^') {

while (Op != NULL && Priority(Op->c) >= Priority(m[k])) {

Op = OutChar(Op, &a);

Out[l++] = a;

}

Op = InChar(Op, m[k]);

}

k++;

}

while (Op != NULL) {

Op = OutChar(Op, &a);

Out[l++] = a;

}

Out[l] = '\0';

reverse(Out, l);

}

void reverse(char \*m, int l) {

std::cout << "Polish: " << m << std::endl;

double Res[4];

int k = 0;

double rez = 0;

for (int i = 0; i < l; i++) {

if (m[i] >= '0' && m[i] <= '9') {

Res[k++] = (int)m[i] - 48;

}

else if (m[i] == '+' || m[i] == '-' || m[i] == '\*' || m[i] == '/' || m[i] == '^') {

switch(m[i]) {

case '+': rez = Res[k - 2] + Res[k-1]; break;

case '-': rez = Res[k - 2] - Res[k-1]; break;

case '\*': rez = Res[k - 2] \* Res[k-1]; break;

case '/': rez = Res[k - 2] / Res[k-1]; break;

case '^': rez = pow(Res[k - 2],Res[k-1]); break;

}

Res[k-2] = rez;

k = k - 1;

}

}

std::cout << "Result: " << rez << std::endl;

}

int Priority(char c) {

int prior = 0;

switch (c) {

case '^':

prior = 4;

break;

case '\*':

prior = 3;

break;

case '/':

prior = 3;

break;

case '+':

prior = 2;

break;

case '-':

prior = 2;

break;

case '(':

prior = 1;

break;

}

return prior;

}

Stack\* InChar(Stack \*p, char s) {

Stack \*t = new Stack;

t->c = s;

t->next = p;

return t;

}

Stack\* OutChar(Stack \*p, char \*s) {

Stack \*t = p;

\*s = p->c;

p = p->next;

delete t;

return p;

}

int lenght (char \*m) {

int len = 0, i = 0;

while (m[i++] != '\0') {

len++;

}

return len;

}

Приложение Б (Makefile)

G++ = g++ -Wall -Werror -Wextra

all: clean lab tests

lab:

$(G++) lab.cpp -o lab

tests:

./lab --forward --file forward1.txt

./lab --reverse --file reverse1.txt

./lab --forward --file forward2.txt

./lab --reverse --file reverse2.txt

./lab --reverse --file doesnot\_exist.txt

clean:

rm -rf lab

**Вывод**

В ходе данной лабораторной работы я научилась переводить выражения, записанные в прямом виде в вид ОПЗ, рассчитывать выражения, записанные в виде ОПЗ, а также использовать Makefile’ы для работы с программами.