Міністерство освіти і науки України Національний технічний університет "ХПІ" кафедра "Інформатика та інтелектуальна власність"

**РОЗРАХУНКОВО-** **ГРАФІЧНЕ ЗАВДАННЯ**

**(або КУРСОВА РОБОТА)**

з дисципліни "*Програмування"*

**Тема**

" 24. Розбір вираження"

Виконав студент групи КН-320г

Шабаш Ігор Володимирович

Керівник РГЗ (або проекту)

Андрій Васильович Івашко

Харків 2020

Оглавление

[1 ДОКЛАДНИЙ ОПИС 3](#_Toc68558571)

[1.1 Завдання 3](#_Toc68558572)

[1.2 Постанова питання 3](#_Toc68558573)

[1.3 Польський інверсний запис опис 3](#_Toc68558574)

[1.4 Алгоритм для обчислення значення виразу 4](#_Toc68558575)

[2 ФУНКЦІЇ ПЕРЕВІРКИ 4](#_Toc68558576)

[2.1 Перевірка на зайві дужки 4](#_Toc68558577)

[2.2 Код перевірки на зайві крапки у числі 5](#_Toc68558578)

[2.3 Код перевірки на зайві символи 5](#_Toc68558579)

[2.4 Код перевірки на зайві операції та числа 6](#_Toc68558580)

[2.5 Додаткові функції для «чистого» коду 6](#_Toc68558581)

[3 ФУНКЦІЯ ПЕРЕВЕДЕННЯ СТРОКИ ДО ОБЕРНЕНОГО ПОЛЬСЬКОГО ЗАПИСУ 7](#_Toc68558582)

[4 РЕАЛІЗАЦІЯ МЕТОДА ОБЧИСЛЕННЯ 9](#_Toc68558583)

[4.1 Функції для перевірки знаку операції 9](#_Toc68558584)

[4.2 Функції для обчислення за знаком двох чисел 10](#_Toc68558585)

[4.3 Функції для обчислення 10](#_Toc68558586)

[5. ОСНОВНА ФУНКЦІЯ 11](#_Toc68558587)

[6. ВИСНОВКИ 12](#_Toc68558588)

## 1 ДОКЛАДНИЙ ОПИС

### Завдання

Користувач задає рядок, в якій містяться операції: множення, ділення, додавання, віднімання. Операції виконуються над цілими і дійсними числами. Видати на екран результат обчислення виразу.

### 1.2 Постанова питання

Для вирішення цього завдання я вирішив використовувати ***Польський інверсний запис***

Це дозволить нам задавати пріоритет операцій та дужок.

По-перше треба перевірити правильність введеного запису. Перевірки: не закриті дужки, зайвий знак операції, зайве число, правильність введеного числа, зайві символи або літери.

По-друге, після усіх можливих перевірок переводимо введений запис до *Польського інверсивного.*

По-третє, знаходимо результат за допомогою перекладу типу String до типу Float.

### 1.3 Польський інверсний запис опис

***Зворотний по́льський запис*** – (зворотний бездужковий запис, постфіксна нотація, польський інверсний запис ) – форма запису математичних виразів, в якій знаки операцій розташовано після операндів. Розташування знаків операцій перед операндами використовує польська нотація.

Зворотний польський запис є зручним для застосування в обчислювальних пристроях. Наприклад, для обчислення виразу

a + b

слід виконати такі дії:

обчислити a

обчислити b

скласти результати

Саме така послідовність і задається польським інверсним записом:

a b +

### 1.4 Алгоритм для обчислення значення виразу

Для всіх символів виконуємо такі дії:

Якщо Аі число, то вкласти його у стек;

Якщо Аі оператор, то:

Витягуємо зі стека два числа;

Виконуємо дію із числами і результат вкладаємо в стек;

Якщо Аі є функцією то:

Витягуємо зі стека одне число;

Визначаємо значення функції із відповідним аргументом та поміщаємо результат у стек;

В кінці роботи в стеку знаходитиметься результат виразу.

Приклад:

Маємо вираз: 12 + 2 \* ( ( 3 \* 4 ) + ( 10 / 5 ) )

Вираз у польському інверсному записі: 12 2 3 4 \* 10 5 / + \* +

Порядок дій над ним буде такий:

## 2 ФУНКЦІЇ ПЕРЕВІРКИ

### 2.1 Перевірка на зайві дужки

int check\_bkl(string infix) {

int status = 0;

int left = 0;

for (int i = 0; i < infix.size(); i++)

if (infix[i] == '(') left++;

int right = 0;

for (int i = 0; i < infix.size(); i++)

if (infix[i] == ')') right++;

if (left == right)

{

status = 1;

}

else

{

cout << "Incorect enter, Reenter expresion:" << endl;

if (left > right) {

cout << "the number of brackets on the left is greater than on the right" << endl;

}

else

cout << "the number of brackets on the right is greater than on the left" << endl;

system("pause");

}

return status;

}

### 2.2 Код перевірки на зайві крапки у числі

bool checkDoubleDot(string input) {

int size = input.size();

short dot = 0;

for (int i = 0; i < size; i++)

{

if (input[i] == '.') {

dot++;

}

}

if (dot > 1) {

cout << input << " have two or more dots"<<endl;

return false;

}

return true;

}

### 2.3 Код перевірки на зайві символи

bool checkLetters(string input) {

int size = input.size();

for (int i = 0; i < size; i++)

{

if ((!isdigit(input[i]) && input[i]!='.' && input[0] != '(' && input[0] != ')') && !checkSign(input)) {

cout << input << " have extra letter" << endl;

return false;

}

}

return true;

}

### 2.4 Код перевірки на зайві операції та числа

bool checkExtraSignAndDigit(vector<string> inputVector){

inputVector.erase(std::remove(inputVector.begin(), inputVector.end(), ")"), inputVector.end());

inputVector.erase(std::remove(inputVector.begin(), inputVector.end(), "("), inputVector.end());

for (int i = 0; i < inputVector.size() - 1; ++i) {

string temp = inputVector[i];

string temp2 = inputVector[i+1];

if (checkSign(temp) == checkSign(temp2)) {

cout << temp << " " << temp2 << " is not corect input, you miss sign or number" << endl;

return false;

}

}

return true;

}

### 2.5 Додаткові функції для «чистого» коду

Функції для видалення зайвих дужок

string delete\_paren(string str) {

if (str.front() == '(') {

str.erase(0, 1);

if (str.back() == ')') {

str.erase(str.size() - 1);

}

}

return str;

}

Функції для друкування вектора, треба для відкладки

void print\_vetor(vector<string> outputList) {

cout << "vector: ";

for (int out = 0; outputList.size() > out; out++) {

cout << " " << outputList[out] << " ";

}

cout << endl;

}

Функції для переведення строки у вектор

vector<string> StrToVector(string input) {

std::stringstream ss(input);

std::istream\_iterator<std::string> begin(ss);

std::istream\_iterator<std::string> end;

std::vector<std::string> vstrings(begin, end);

std::copy(vstrings.begin(), vstrings.end(), std::ostream\_iterator<std::string>(std::cout, "\n"));

system("cls");

return vstrings;

}

Функції для видалення будь-якого символу з строки

string deleteSymbolFromString(string input,char symbol) {

input.erase(std::remove(input.begin(), input.end(), symbol), input.end());

return input;

}

Функції для введення виразу

string inputString() {

string temp;

string fullString;

std::string equal = "=";

for (int out = 0; out != 1;)

{

getline(cin, temp);

fullString = fullString + ' ' + temp;

if (temp.find(equal) != string::npos) {

out++;

}

}

return fullString;

}

## 3 ФУНКЦІЯ ПЕРЕВЕДЕННЯ СТРОКИ ДО ОБЕРНЕНОГО ПОЛЬСЬКОГО ЗАПИСУ

vector<string> StringToRPN(string infix) {

istringstream iss(infix);

vector<string> tokens;//store the tokens here

while (iss)

{

string temp;

iss >> temp;

tokens.push\_back(temp);

}

vector<string> outputList;//output vector

stack<string> s;//main stack

//operator: +, -, \*, /, ^, ()

//operands: 1234567890

for (unsigned int i = 0; i < tokens.size(); i++) //read from right to left

{

if (TryParse(tokens[i]))

{

outputList.push\_back(tokens[i]);

}

if (tokens[i] == "(")

{

s.push(tokens[i]);

}

if (tokens[i] == ")")

{

while (!s.empty() && s.top() != "(")

{

outputList.push\_back(s.top());

s.pop();

}

s.pop();

}

if (isOperator(tokens[i]) == true)

{

while (!s.empty() && Priority(s.top()) >= Priority(tokens[i]))

{

outputList.push\_back(s.top());

s.pop();

}

s.push(tokens[i]);

}

}

//pop any remaining operators from the stack and insert to outputlist

while (!s.empty())

{

outputList.push\_back(s.top());

s.pop();

}

return outputList;

}

bool TryParse(const string& symbol)

{

bool isNumber = false;

for (unsigned int i = 0; i < symbol.size(); i++)

{

if (!isdigit(symbol[i]))

{

isNumber = false;

}

else

{

isNumber = true;

}

}

return isNumber;

}

int Priority(const string& c)

{

if (c == "^")

{

return 3;

}

if (c == "\*" || c == "/")

{

return 2;

}

if (c == "+" || c == "-")

{

return 1;

}

else

{

return 0;

}

}

bool isOperator(const string& c)

{

return (c == "+" || c == "-" || c == "\*" || c == "/" || c == "^");

}

## 4 РЕАЛІЗАЦІЯ МЕТОДА ОБЧИСЛЕННЯ

### 4.1 Функції для перевірки знаку операції

bool is\_sign(string str) {

std::string plus = "+";

std::string minus = "-";

std::string mult = "\*";

std::string div = "/";

if (str.find(plus) != string::npos) {

return true;

}

else if (str.find(minus) != string::npos) {

return true;

}

else if (str.find(mult) != string::npos) {

return true;

}

else if (str.find(div) != string::npos) {

return true;

}

else

return false;

}

### 4.2 Функції для обчислення за знаком двох чисел

float Opertion(string str,float a, float b) {

std::string plus="+";

std::string minus="-";

std::string mult="\*";

std::string div="/";

if (str.find(plus) != string::npos) {

return a + b;

}

else if (str.find(minus) != string::npos) {

return a - b;

}

else if (str.find(mult) != string::npos) {

return a \* b;

}

else if (str.find(div) != string::npos) {

if (b == 0) {

cout << "error: cannot be divided by zero"<<endl;

}

return a / b;

}

else

return 0;

}

### 4.3 Функції для обчислення

vector<string> count\_RPN(vector<string> outputList){

string operation;

float a;

float b;

int current\_step;

for (int step = 0; outputList.size() != 1; step++)

{

operation = outputList[step];

//print\_vetor(outputList);

if (is\_sign(operation)) {

current\_step = step;

step--;

b = stof(outputList[step]);

outputList.erase(outputList.begin() + current\_step);

step--;

a = stof(outputList[step]);

outputList.erase(outputList.begin() + --current\_step);

/\*cout << "a: " << a << endl;

cout << "b: " << b << endl;

cout <<"result: "<< Opertion(operation, a, b)<<endl;\*/

outputList[--current\_step] = to\_string(Opertion(operation, a, b));

}

}

return outputList;

}

## 5. ОСНОВНА ФУНКЦІЯ

int main()

{

string infix;// = "12.3 \* 3 + 1 \* 0 + 2 / 0 + ( ( 5 \* 1 ) + 1 ) / 0 + 1=";//our infix expression

for (int status = 0; status != 2 ; )

{

cout << "Enter expresion, after every symbol put space \"( ( a + b ) \* c )\":" << endl;

infix = deleteSymbolFromString(inputString(),'=');

status=check\_bkl(infix);

if (status == 1) {

vector<string> temp = StrToVector(infix);

if (checkExtraChar(temp)) {

status++;

}

system("pause");

}

system("cls");

}

vector<string> outputList = StringToRPN(infix);

outputList = count\_RPN(outputList);

cout <<infix <<" = " <<outputList[0];

return 0;

}

## 6. ВИСНОВКИ

Вході виконання роботи було вирішено поставлене завдання, обрахувати введений вираз, з усіма математичними правилами, такі як пріоритети знаків операцій та дужок, а також перевірки правильності введення строки для запобігання помилок в ході роботи програми. Також були використанні знання з інших дисциплін( архітектура ОС), саме про польський інверсний запис, який допоміг знайти правильні пріоритети знаків операцій та дужок.

Були також використанні принципи SOLID та KISS, також декомпозиція питання на декілька окремих модулей(ФУНКЦІЇ ПЕРЕВІРКИ, ФУНКЦІЯ ПЕРЕВЕДЕННЯ СТРОКИ ДО ОБЕРНЕНОГО ПОЛЬСЬКОГО ЗАПИСУ, РЕАЛІЗАЦІЯ МЕТОДА ОБЧИСЛЕННЯ)