Міністерство освіти і науки України
Національний технічний університет України „КПІ”
Факультет інформатики та обчислювальної техніки

Кафедра автоматизованих систем обробки
інформації та управління

ЗВІТ

до лабораторної роботи № 1

з дисципліни ООП

Виконав студент *ІП-61 Кушка Михайло Олександрович*

(№ групи, прізвище, ім’я, по батькові)

Прийняв *Головченко М.М.*

(посада, прізвище, ім’я, по батькові)

Київ 2017

ЗМІСТ

1. **Мета роботи 3**
2. **Постановка задачі 4**
3. **Аналітичні викладки 5**
4. **UML-діаграма класів 6**
5. **Вихідний код програми 7**
6. **Приклади роботи програми 13**
7. **Висновки 14**

1. МЕТА РОБОТИ

Мета роботи - вивчити основні концепції об'єктно-орієнтованого програмування. Вивчити особливості використання класів і об'єктів, а також особливості застосування конструкторів і деструкторів.

2. ПОСТАНОВКА ЗАДАЧІ

Визначити клас «Алгебраїчний вираз», з закритим елементом-рядком fx - який являє собою вираження алгебри однієї змінної. Передбачити в класі:

- конструктор ініціалізації, для введення рядка з клавіатури (за замовчуванням рядок порожній).

Реалізувати в класі метод знаходження значення алгебраїчного виразу при заданій змінній. Для спрощення виконання використовувати список (стек).

Визначити деструктор класу.

3. АНАЛІТИЧНІ ВИКЛАДКИ

В С ++ для зниження накладних витрат на виклики функцій - особливо невеликих - передбачені вбудовані функції:

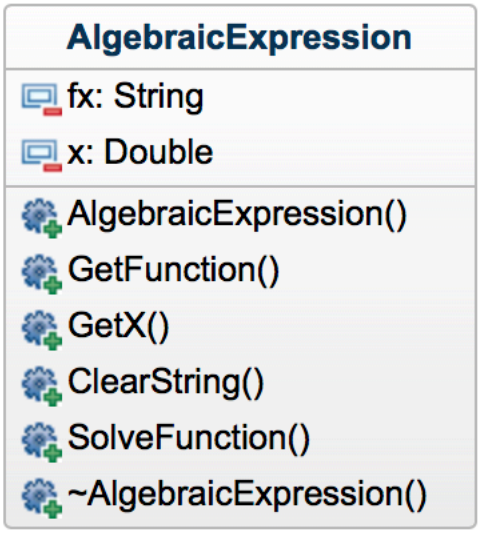
inline тип імя_функциі (параметри).

С ++ дозволяє визначити кілька функцій з одним і тим же ім'ям, якщо ці функції мають різні набори параметрів. ця особливість називається перевантаженням функції. При виклику перевантаженої функції компілятор С ++ визначає відповідну функцію шляхом аналізу кількості, типів і порядку слідування аргументів у виклику.

Клас - різновид абстрактного типу даних в об'єктно-орієнтованому програмуванні, що характеризується способом свого побудови. Поряд з поняттям «об'єкта» клас є ключовим поняттям в ООП.

Функції-елементи класу - це функції, які мають доступ до будь-яких інших функцій-елементів та до будь-яких елементів-даними. Найбільш часто використовуються функції, що реалізують арифметичні методи, методи відображення і порівняння даних. Особливим видом функцій-елементів є конструктори і деструктори.

4. UML-ДІАГРАМА КЛАСІВ



5. ВИХІДНИЙ КОД ПРОГРАМИ

shunting_yard.cpp

```
//
//  shunting_yard.cpp
//  Lab1
//
//  Created by Kushka Misha on 9/13/17.
//  Copyright © 2017 Kushka Misha. All rights reserved.
//

#include "shunting_yard.hpp"


double f(string s, double x)
{
    // Insert '*' if nesessary ("2x" -> "2*x")
    list<char> good = {'+', '-', '*' , '/', '('};
    for (int i = 0; i < s.length(); i++) {
        if (s[i] == 'x' && i != 0) {
            if (!(find(good.begin(), good.end(), s[i-1]) != good.end())) {
                s.insert(i, "*");
            }
        }
    }
    list<string>queue = ShuntingAlgorithm(s, x);

    return ReversePolish(queue);
}

list<string> ShuntingAlgorithm(string s, double x)
{
    /**
     * The shunting algorithm
     */
    list<char> stack;
    list<string> queue;
    string prev = "o";
    string number;

    map<char, int> precedence = {
        {'-', 1}, {'+', 1},
        {'*', 2}, {'/', 2},
        {'^', 3}
    };

    // replace x^3 by x*x*x and so on
    while (s.find("x^") != -1)
    {
        //        cout << "here" << endl;
        int pos = s.find("x^");
        int n = s[pos + 2] - '0';
        s.erase(pos, 3);
        string power = "";

        for (int i = 0; i < n; ++i)
            power += "x*";
        power.erase(power.end() - 1);
```

```
        s.insert(pos, power);
    }

    // replace 'x' by x-value
    s = ReplaceAll(s, "x", to_string(x));

    // replace all '--' by '+'
    s = ReplaceAll(s, "--", "+");
    if (s[0] == '+')
        s.erase(s.begin(), s.begin() + 1);

    // work with '-'
    bool numbers_before = false;
    int i = 0;
    while(i < s.length())
    {
        if (isdigit(s[i]))
            numbers_before = true;
        i++;
        if (s[i] == '-' && numbers_before && isdigit(s[i - 1]))
        {
            s.insert(i, "+");
            i++;
        }
    }

    for (int i = 0; i < s.length(); ++i)
    {
        if (isdigit(s[i]) || s[i] == '.' || s[i] == '-')
        {
            // It's a number
            if (prev.length() == 1)
            {
                if (isdigit(prev[0]) || prev[0] == '.' || prev[0] == '-')
                {
                    number = prev + string(1, s[i]);
                    queue.pop_back();
                    queue.push_back(number);
                    prev = number;
                }
                else
                {
                    queue.push_back(string(1, s[i]));
                    prev = s[i];
                }
            }
            else
            {
                number = prev + string(1, s[i]);
                queue.pop_back();
                queue.push_back(number);
                prev = number;
            }
        }
        else
        {
            if (s[i] == '+' || s[i] == '-' || s[i] == '*' || s[i] == '/' || s[i] ==
'^')
            {
                // It's an operator
                while (!stack.empty())
                {
```

```
                    if (precedence[stack.back()] > precedence[s[i]])
                    {
                        queue.push_back(string(1, stack.back()));
                        stack.pop_back();
                    }
                    else
                        break;
                }
                stack.push_back(s[i]);
            }
            else
            {
                if (s[i] == '(')
                    stack.push_back('(');
                if (s[i] == ')')
                {
                    while (!stack.empty())
                    {
                        if (stack.back() != '(')
                        {
                            queue.push_back(string(1, stack.back()));
                            stack.pop_back();
                        }
                        else
                        {
                            stack.pop_back();
                            break;
                        }
                    }
                }
            }
            prev = s[i];
        }
    }

    // While there's operators on the stack, pop them to the queue
    while (!stack.empty())
    {
        queue.push_back(string(1, stack.back()));
        stack.pop_back();
    }

    return queue;
}

double ReversePolish(list<string> queue)
{
    /**
     * Reverse polish
     */
    double a, b, result = 0;
    list<double>fstack;
    string str;

    for (list<string>::iterator p = queue.begin(); p != queue.end(); ++p)
    {
        str = *p;
        if (str[0] == '-')
        {
            fstack.push_back(stod(str));
            continue;
        }
```

```
        if (isdigit(str[0]))
            fstack.push_back(stod(str));
        else
        {
            a = fstack.back();
            fstack.pop_back();
            b = fstack.back();
            fstack.pop_back();

            switch (str[0])
            {
                case '+':
                    result = b + a;
                    break;
                case '-':
                    result = b - a;
                    break;
                case '*':
                    result = b * a;
                    break;
                case '/':
                    result = (double) b / a;
                    break;
                case '^':
                    result = (double) pow((double) b, a);
                    break;
            }
            fstack.push_back(result);
        }
    }

    return fstack.back();
}

string ReplaceAll(string str, const string& from, const string& to) {
    // Replace all occurences in the string.
    size_t start_pos = 0;
    while((start_pos = str.find(from, start_pos)) != string::npos) {
        str.replace(start_pos, from.length(), to);
        start_pos += to.length();
    }
    return str;
}
```

shunting_yard.cpp

```
//
//  shunting_yard.hpp
//  Lab1
//
//  Created by Kushka Misha on 9/13/17.
//  Copyright © 2017 Kushka Misha. All rights reserved.
//

#ifndef shunting_yard_hpp
#define shunting_yard_hpp

#include <iostream>
#include <string>
#include <list>
#include <map>
```

```
#include <cmath>
#include <stdio.h>
using namespace std;

list<string> ShuntingAlgorithm(string, double);
double ReversePolish(list<string>);
string ReplaceAll(string, const string&, const string&);
double f(string, double);

#endif /* shunting_yard_hpp */
```

class_prototype.cpp

```
//
//  class_prototype.cpp
//  Lab1
//
//  Created by Kushka Misha on 9/13/17.
//  Copyright © 2017 Kushka Misha. All rights reserved.
//

class AlgebraicExpression {
    string fx = "";
    double x = 0;
public:
    AlgebraicExpression();
    void GetFunction();
    void GetX();
    void ClearString();
    void SolveFunction();
    ~AlgebraicExpression();
};
```

main.cpp

```
#include "shunting_yard.hpp"
#include "class_prototype.hpp"

void AuthorInfo();

int main() {
    AuthorInfo();

    AlgebraicExpression *ptr;

    char quit = 'y';
    while (quit != 'n') {
        ptr = new AlgebraicExpression;
        ptr->SolveFunction();
        delete ptr;

        // Break the cycle.
        cout << "\nTry again? (y/n)\n> ";
        cin >> quit;
    }

    return 0;
}

AlgebraicExpression::AlgebraicExpression() {
```

```
    // Get function and x-value for it.
    cout << "Enter algebraic expression with one variable\n> ";
    getline(cin, fx);
    cout << fx << endl;
//    cin >> fx;
    this->ClearString();
}
AlgebraicExpression::~AlgebraicExpression() {
    // Class destructor.
    cout << "\nMust delete something..." << endl;
    fx = "";
    x = 0;
}

void AlgebraicExpression::GetFunction() {
    cout << "F(x) = " << fx << endl;
}
void AlgebraicExpression::GetX() {
    cout << "x = " << x << endl;
}
void AlgebraicExpression::SolveFunction() {
    cout << "Enter value of x\n> ";
    cin >> x;

    double answer = f(fx, x);
    cout << "Answer = " << answer << endl;
}
void AlgebraicExpression::ClearString() {
    // Delete all spaces in the string.
    fx.erase(remove(fx.begin(), fx.end(), ' '), fx.end());
}

void AuthorInfo() {
    cout << "\
    ----------------------\n\
    | Kushka Misha, IP-61 |\n\
    | Level: 3            |\n\
    | Variant: 15         |\n\
    ----------------------\n\n";
}
```

6. ПРИКЛАДИ РОБОТИ ПРОГРАМИ

```
----------------------
| Kushka Misha, IP-61 |
| Level: 3            |
| Variant: 15         |
----------------------

Enter algebraic expression with one variable
> (2x-1)^2
(2x-1)^2
Enter value of x
> 2
Answer = 9

Must delete something...

Try again? (y/n)
> n
Program ended with exit code: 0
```

7. ВИСНОВКИ

У даній лабораторній роботі я навчився працювати з класами в мові програмування C++ та познайомився з їх деякими особливостями. Дізнався про перевантаження функцій та nullptr.