МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ТАРАСА ШЕВЧЕНКА

ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Кафедра інформаційних систем та технологій

**Лабораторна робота №1**

з дисципліни **«Об’єктно орієнтоване програмування»**

**Тема: «Класи С++»**

*Виконав здобувач 1-ого курсу*

Денис ШЕВЧЕНКО

Група: ІР-11

Перевірив:

Викладач к.ф.-м.н., ас. Пономаренко Роман Миколайович

Дата:

Оцінка:

Київ - 2023

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

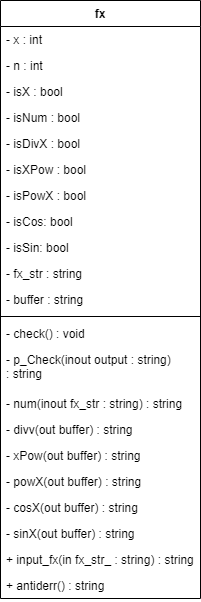
**Практична частина.**

**Варіант №18(В)**

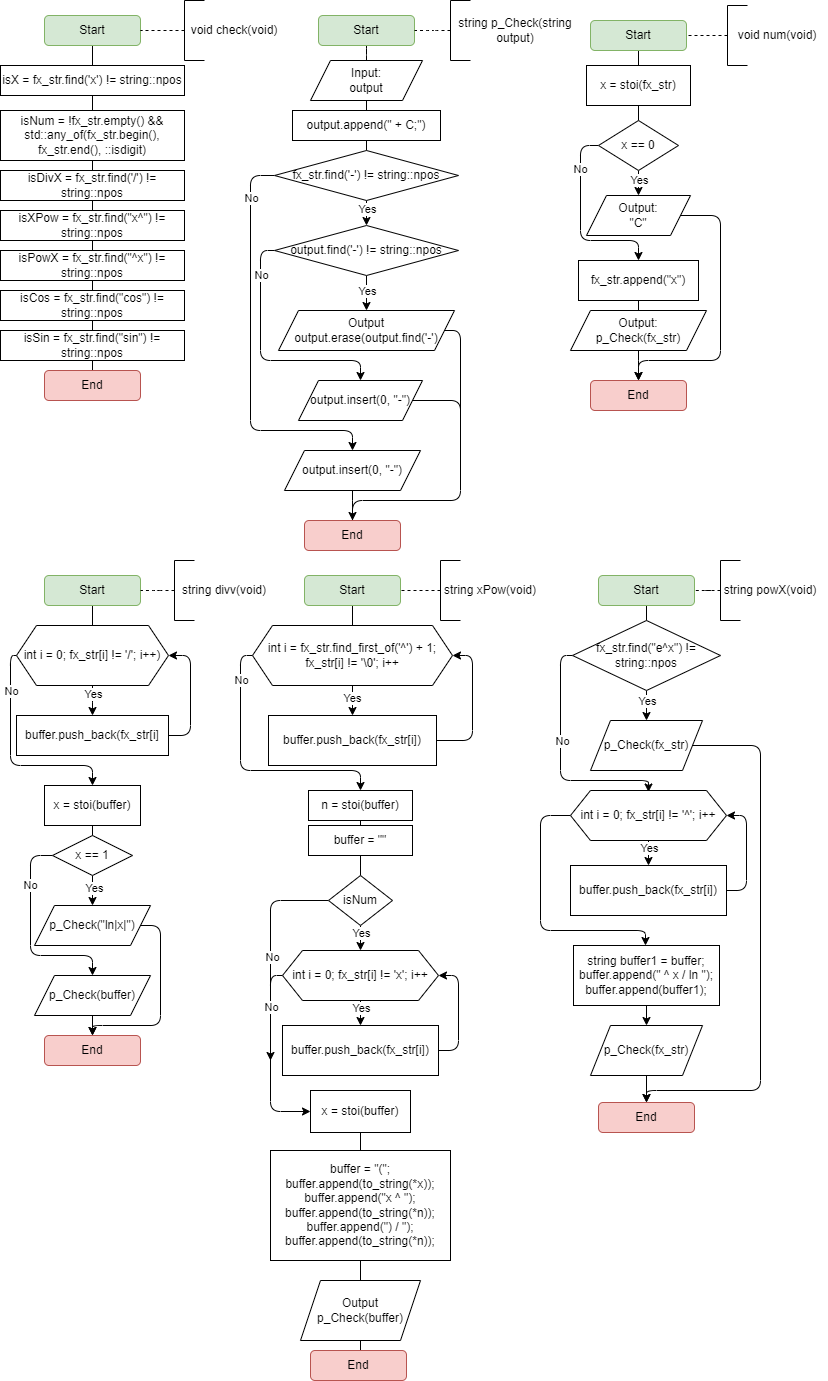
Изображение выглядит как текст

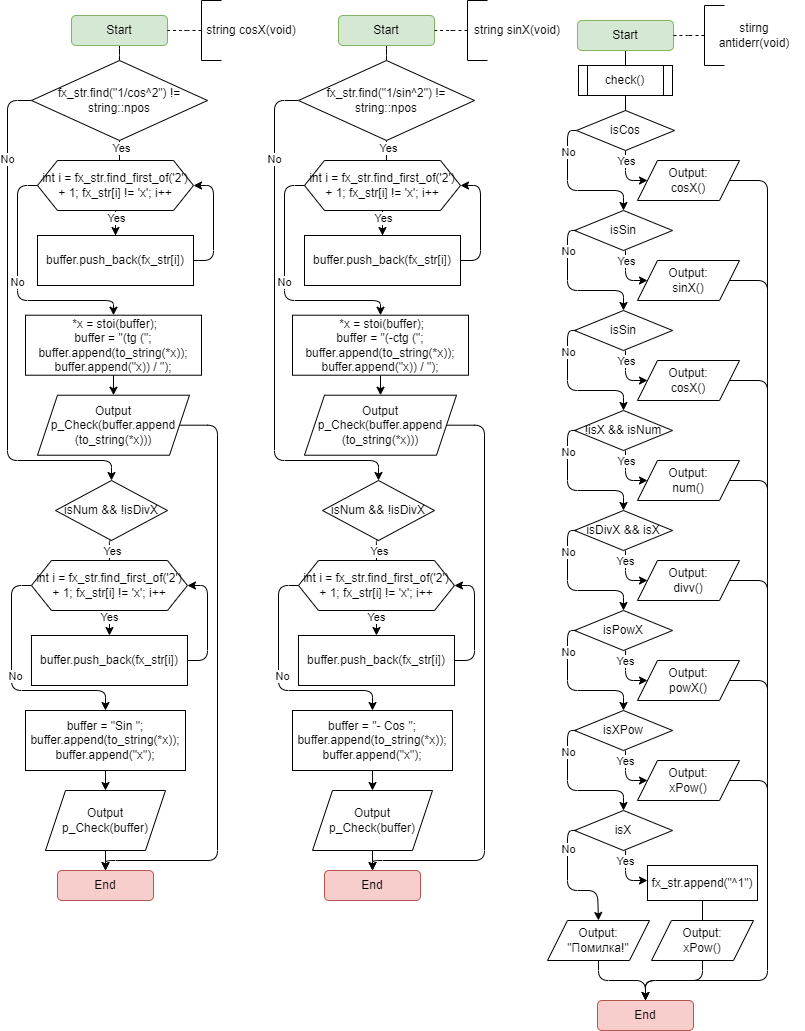
Автоматически созданное описание

**UML-Діаграма**

****

**Блок-схема алгоритму**

****



**Програма на С++**

#include <iostream>  
#include <string>  
#include <windows.h>  
  
using namespace std;  
  
class fx {  
private:  
 int \*x;  
 int \*n;  
 bool isX;  
 bool isNum;  
 bool isDivX;  
 bool isXPow;  
 bool isPowX;  
 bool isCos;  
 bool isSin;  
  
 string fx\_str;  
 string buffer;  
  
 void check(void){  
 isX = fx\_str.find('x') != string::npos;  
 isNum = !fx\_str.empty() && std::any\_of(fx\_str.begin(), fx\_str.end(), ::isdigit);  
 isDivX = fx\_str.find('/') != string::npos;  
 isXPow = fx\_str.find("x^") != string::npos;  
 isPowX = fx\_str.find("^x") != string::npos;  
 isCos = fx\_str.find("cos") != string::npos;  
 isSin = fx\_str.find("sin") != string::npos;  
 }  
  
 string p\_Check(string output) {  
 output.append(" + C;");  
 if (fx\_str.find('-') != string::npos) {  
 if (output.find('-') != string::npos)  
 return output.erase(output.find('-'), 1);  
 return output.insert(0, "-");  
 }  
 return output;  
 }  
  
 string num(void) {  
 \*x = stoi(fx\_str);  
 if (\*x == 0)  
 return "C;";  
 else {  
 fx\_str.append("x");  
 return p\_Check(fx\_str);  
 }  
 }  
  
 string divv(void) {  
 for (int i = 0; fx\_str[i] != '/'; i++)  
 buffer.push\_back(fx\_str[i]);  
 \*x = stoi(buffer);  
  
 if (\*x == 1)  
 return p\_Check("ln|x|");  
 else {  
 buffer.append(" \* ln|x|");  
 return p\_Check(buffer);  
 }  
 }  
  
 string xPow(void) {  
 for (int i = fx\_str.find\_first\_of('^') + 1; fx\_str[i] != '\0'; i++)  
 buffer.push\_back(fx\_str[i]);  
 \*n = stoi(buffer);  
 if (\*n == -1)  
 return "Сталася помилка!\nСтепінь не може доорівнювати -1.";  
 else {  
 \*n += 1;  
 buffer = "";  
 if (fx\_str.find\_first\_of('x') - 1 != string::npos) {  
 for (int i = 0; fx\_str[i] != 'x'; i++)  
 buffer.push\_back(fx\_str[i]);  
 \*x = stoi(buffer);  
 } else  
 \*x = 1;  
 if (!(\*x % \*n)) {  
 \*x /= \*n;  
 buffer = to\_string(\*x);  
 buffer.append("x ^ ");  
 buffer.append(to\_string(\*n));  
 return p\_Check(buffer);  
 } else {  
 buffer = "(";  
 buffer.append(to\_string(\*x));  
 buffer.append("x ^ ");  
 buffer.append(to\_string(\*n));  
 buffer.append(") / ");  
 buffer.append(to\_string(\*n));  
 return p\_Check(buffer);  
 }  
 }  
 }  
  
 string powX(void) {  
 if (fx\_str.find("e^x") != string::npos)  
 return p\_Check(fx\_str);  
 else {  
 for (int i = 0; fx\_str[i] != '^'; i++)  
 buffer.push\_back(fx\_str[i]);  
 string buffer1 = buffer;  
 buffer.append(" ^ x / ln ");  
 buffer.append(buffer1);  
 return p\_Check(buffer);  
 }  
 }  
  
 string cosX(void){  
 if (fx\_str.find("1/cos^2") != string::npos) {  
 for (int i = fx\_str.find\_first\_of('2') + 1; fx\_str[i] != 'x'; i++)  
 buffer.push\_back(fx\_str[i]);  
 if(buffer.empty())  
 return p\_Check("tg (x)");  
 \*x = stoi(buffer);  
 buffer = "(tg (";  
 buffer.append(to\_string(\*x));  
 buffer.append("x)) / ");  
 return p\_Check(buffer.append(to\_string(\*x)));  
 }  
 if (fx\_str.find('^') != string::npos)  
 return "Программа не можне знайти первісну від cos у степені!\nСпробуйте ще раз за табличними значеннями.";  
 if (isNum && !isDivX){  
 for (int i = fx\_str.find\_first\_of('s') + 1; fx\_str[i] != 'x'; i++)  
 buffer.push\_back(fx\_str[i]);  
 \*x = stoi(buffer);  
 }  
 else  
 \*x = 1;  
  
 buffer = "Sin ";  
 buffer.append(to\_string(\*x));  
 buffer.append("x");  
 return p\_Check(buffer);  
 }  
  
 string sinX(void){  
 if (fx\_str.find("1/sin^2") != string::npos) {  
 for (int i = fx\_str.find\_first\_of('2') + 1; fx\_str[i] != 'x'; i++)  
 buffer.push\_back(fx\_str[i]);  
 if(buffer.empty())  
 return p\_Check("-ctg (x)");  
 \*x = stoi(buffer);  
 buffer = "(-ctg (";  
 buffer.append(to\_string(\*x));  
 buffer.append("x)) / ");  
 return p\_Check(buffer.append(to\_string(\*x)));  
 }  
  
 if (fx\_str.find('^') != string::npos)  
 return "Программа не можне знайти первісну від cos у степені!\nСпробуйте ще раз за табличними значеннями.";  
 if (isNum && !isDivX){  
 for (int i = fx\_str.find\_first\_of('n') + 1; fx\_str[i] != 'x'; i++)  
 buffer.push\_back(fx\_str[i]);  
 \*x = stoi(buffer);  
 }  
 else  
 \*x = 1;  
  
 buffer = "- Cos ";  
 buffer.append(to\_string(\*x));  
 buffer.append("x");  
 return p\_Check(buffer);  
 }  
  
public:  
 fx(void) : x(new int(0)), n(new int(0)), isX(0), isDivX(0), fx\_str(""), buffer("") {}  
  
 ~fx(void) {  
 //cout << "Destructed!\n";  
 delete x;  
 delete n;  
 x = nullptr;  
 n = nullptr;  
 }  
  
 void input\_fx(void) {  
 cout << "Введіть функцію f(x): ";  
 cin >> fx\_str;  
 }  
  
 string antiderr(void) {  
 check();  
  
 if (isCos)  
 return cosX();  
 if (isSin)  
 return sinX();  
 if (!isX && isNum)  
 return num();  
 if (isDivX && isX)  
 return divv();  
 if (isPowX)  
 return powX();  
 if (isXPow)  
 return xPow();  
 if (isX) {  
 fx\_str.append("^1");  
 return xPow();  
 }  
 else  
 return "Сталася помилка при підрахунку первісної!";  
 }  
};  
  
int main() {  
 SetConsoleOutputCP(1251);  
 cout << "Студента першого курсу"  
 << "\nГрупи ІР-11"  
 << "\nШевченко Дениса"  
 << "\nВаріант №18(В)\n\n";  
 char y = 'y';  
 while (y == 'y') {  
 fx func;  
 func.input\_fx();  
 cout << "\nПервісна від вашої функції: \n" << func.antiderr();  
  
 cout << "\n\nБажаєте перезапустити?(y/n)";  
 cin >> y;  
 }  
 cout << "\nПрограма успішно завершена!\n";  
 return 0;  
}

**Приклади роботи програми.**

|  |  |  |
| --- | --- | --- |
|  |  |  |
|  |  |  |
|  |  |  |

**Відповіді на контрольні запитання.**

1. Зазначення **функції як вбудованої** за допомогою приписки inline рекомендує компілятору зробити копію цієї функції, щоб уникнути частих викликів цієї функції.
2. Механізм **перевантаження функції** полягає в створенні багатьох функцій з однаковими назвами, але різними вхідними даними. Тому подаючи різні типи або кількості даних в функцію ми можемо викликати зовсім різні функції.
3. **Nullptr** потрібен для занулення покажчика. Nullptr – це власний тип даних, та прирівнюванні до нього функцій або змінних ми позбавляємося подальших проблем з ними(наприклад перевантаження).
4. **Rvalue** посилання дещо схоже на звичайне посилання у С++, за винятком того, що rvalue можна прив’язати до деякого виразу, а lvalue – не константне, тож його прив’язати не вдасться. Rvalue позначається як &&.
5. **Клас** – визначений користувачем тип даних, який включає в себе набір змінних та спеціальних функцій. Вони можуть бути поділені на рівні доступу.
6. **Інкапсуляція** являє собою одночасно два терміни – мовна конструкція, що обмежує доступ до даних та механізм, що дозволяє передавати у закриті змінні деякі дані через методи обробки даних. Тобто це захист даних в межах класу, та передбачення методів для доступу до цих даних.
7. **Елементи-дані** – це данні, що належать до будь-якого класу. Вони можуть бути будь якими типами, структурами даних та навіть зберігати покажчики на той самий клас, якому належать. Але не можуть бути представниками цього класу.

**Функції-елементи** – це функції, які мають доступ до будь-яких інших функцій-елементів або елементів-даних та належать якомусь класу.

1. **Конструктор** – свого роду відкрита функція-елемент, що викликається при створенні класу та задає деякі дані за замовчуванням або за викликом. Він розпізнається по імені, яке збігається з ім’ям самого класу.

**Конструктор за замовчуванням** – це конструктор, який викликається без передачі аргументів під час створення об’єкту.

**Конструктор ініціалізації** – викликається, якщо передати будь-які аргументи під час створення об’єкту.

**Конструктор копіювання** – створює новий об’єкт класу на основі існуючого.

**Конструктор переміщення** – спрощення версія копіювання, яка дозволяє створювати об’єкти на основі тимчасових виразів, замінивши процедуру переміщенням покажчика.

**Конструктор перетворення** – конструктор, що перетворює скалярні величини на класові чи навпаки.

1. **Деструктор** – функція-елемент, що спрацьовує при знищенні об’єкту класу або його виходу за межі видимості програми. Позначається символом “~”. Деструктори необідні, якщо у класі є змінні динамітного типу. За допомогою деструктора вдається позбавитися витоків пам’яті.
2. **Створення об’єкта класу** в програмі може здійснювати звичайним оголошенням відповідної змінної, або динамічним розміщенням цього об’єкта в пам’яті. Існують випадки, коли можна створити динамічні масиви або матриці із об’єктів класів(за допомогою “\*”, або “\*\*” відповідно). В цілому, створення об’єктів класів майже не відрізняється від ініціалізації інших типів даних, адже об’єкти – це теж тип даних. Також, не слід забувати про видалення динамічно створених об’єктів за допомогою delete.
3. Коли метод, що належить класу, викликається, крім явно оголошених параметрів автоматично передається прихований (неявний) параметр **this**, у вигляді покажчика. В кожному методі, що належить класу, неявний параметр вказує саме на той об’єкт, для якого викликають цей метод. Тобто, за допомогою нього, ми можемо звертатися саме до елементів-даних об’єкта класу.

**Висновок.**

Під час цієї лабораторної роботи я ознайомився зі створенням свої класів, та відповідно, їх ініціалізації у вигляді об’єктів у програмі. У класах я передбачив конструктор за замовчуванням та деструктор. Також було створено елементи-дані й функції-елементи класу для вирішення поставленої задачі на прикладі мови програмування високого рівня С++.