**Проектування і реалізація програм засобами об’єктно-орієнтованого програмування**

### Завдання № 1

### Основні завдання роботи

1. Реалізувати програмно клас, який реалізує роботу стека.
2. Розробити і реалізувати програму, що використовує клас стека для моделювання Т-подібного сортувального вузла на залізній дорозі:
   1. У програмі слід реалізувати поділ поїзда, що складається з вагонів двох типів, на два напрямки (на кожний напрямок формується поїзд з вагонів одного типу).
   2. Реалізувати у програмі можливість формування залізничного состава користувачем з клавіатури та випадковим чином до попередньої програми.

### Порядок виконання роботи

1. Скласти і реалізувати програмно опис класу стека, як організованої структури даних, що працює за принципом FCLS (first come last served — першим прийшов останнім пішов).
2. Елементами стека будуть об’єкти класу вагону. У класі вагону слід відобразити тип вагону та його порядковий номер у закритій частині класу. Задати конструктори і деструктори цього класу. Задати функції-члени, які будуть повертати значення закритих елементів.

Передбачити у класі стека функції-члени, які забезпечують:

— формування початкового елемента стека;

* формування наступних за початковим елементів стека, ураховуючи зв’язки між елементами;

додавання нового елемента в кінець стека;

* знищення елемента з кінця стека; якщо елемент, що знищується, є єдиним, передбачити існування порожнього стека;
* перевірку меж заповнення елементами стека;
* підрахування кількості елементів у стеку.

1. Передбачити у програмі функцію формування початкового об’єкта стека, що являє собою імітацію початкового залізничного складу, що складається з вагонів двох типів, з якого потім будуть братись елементи для сортування на склади з вагонів двох різних типів. Реалізувати можливість присвоєння вагонам типів випадковим чином та за допомогою клавіатури (користувачем).
2. Для моделювання роботи Т-подібного сортувального вузла, написати у програмі функцію, яка використовує початковий об’єкт класу стека для формування двох об’єктів типу стек, у кожний з яких заносяться вагони лише одного типу.
3. Передбачити у програмі функцію виведення елементів стека на екран.

Програма повинна містити меню, яке дозволить перевірити основні функції, створені у програмі.

**Завдання № 2**

### Основні завдання роботи

Розробити і реалізувати програмний додаток нарахування заробітної плати співробітникам фірми. Для збереження даних про співробітників використати масиви об’єктів, а для доступу до функцій — вказівники. Для цього:

1. Розробити і реалізувати програмно клас “Особа”.
2. Розробити і реалізувати програмно функції, які враховують такі вимоги:
   1. Кожному співробітнику заробітна плата нараховується, виходячи з тарифної сітки, у якій закладено такі позиції: “Спеціаліст”, “Спеціаліст 1 категорії”, “Спеціаліст 2 категорії”, “Спеціаліст вищої категорії”, “Начальник відділу”, “Директор”.
   2. Кожному співробітнику призначається додатково до ставки премія за принципом: якщо заробітна плата менша від середньої заробітної плати усіх співробітників, то розмір премії становить 50% від заробітної плати, в інших випадках — 30% від заробітної плати.
   3. Реалізувати програмно функцію, яка визначає суму, що видається на руки (заробітна плата плюс премія).
3. Продемонструвати роботу створеного класу і його функцій в основній частині програми.

### Порядок виконання роботи

1. Для реалізації програмного коду оголосити та визначити клас *особа*. У цьому класі слід оголосити закриті змінні, що зберігають такі дані: прізвище та ініціали співробітника, заробітну плату, премію, ставку, позицію з тарифної сітки.
2. Додати у клас відкриті конструктори. Один з конструкторів оголосити без параметрів, інший з параметрами для ініціалізації об'єктів класу. Конструктор не повинен ініціалізувати значення ставки, натомість він має викликати функцію, яка її визначає.
3. Додати у клас деструктор.
4. Додати у клас функцію визначення ставки відповідно до позиції у тарифній сітці.
5. Додати у клас функцію призначення премій. Врахувати, що премія призначається за принципом: якщо заробітна плата менша, ніж середня зарплата усіх співробітників, то розмір премії становить 50% від заробітної плати, в інших випадках — 30% від заробітної плати.
6. Додати функцію визначення заробітної плати.

9. Додати функцію виведення даних про співробітника.

В основній частині програми створити масив об’єктів. Доступ до елементів масиву показати через вказівник.

1. Програма повинна містити меню, яке дозволить перевірити основні функції, створені у програмі.

**Завдання № 3**

### Основні завдання роботи

Розробити та реалізувати програмний додаток, який оперує над множенням матриць та векторів. Для цього:

1. Розробити та реалізувати програмно клас *матриця*.
2. Розробити та реалізувати програмно клас *вектор*.

Реалізувати програмно перевантаження функції множення вектора на матрицю, матриці на вектор, матриці на матрицю, множення числа на матрицю, матриці на число.

Продемонструвати роботу створених класів та їх функцій в основній частині програми.

**Порядок виконання роботи**

1. Для реалізації програмного коду створити два класи: *матриця* і *вектор*. У кожному з класів задати конструктори і деструктори.
2. Як закриті елементи класів створити динамічні масиви, що зберігають відповідно матрицю або вектор та їх розмірність. Динамічний масив матриці має бути двовимірним, вектора — одновимірним.

Додати дружні функції множення вектора на матрицю і навпаки для цих двох класів. Як параметри передати об’єкти створених класів.

1. Перевантажити створену функцію для множення вектора на число і навпаки.
2. Перевантажити створену функцію для множення матриці на число і навпаки.
3. Перевантажити створену функцію для множення двох матриць (як об’єктів класу *матриця*).
4. Перевантажити створену функцію для множення двох векторів (як об’єктів класу *вектор*).
5. Під час реалізації перевантажених функцій перевірити можливість такого множення (не всі вектори і матриці можна перемножати).

В основній частині програми продемонструвати роботу перевантажених функцій.

**Завдання № 4**

### Основні завдання роботи

Реалізувати програмно клас, що оперує з комплексними числами. Для цого виконати такі завдання:

1. Розробити і реалізувати програмно клас комплексного числа.
2. Реалізувати перевантаження операцій:
   * додавання (+);
   * віднімання (-);
   * множення (\*);
   * множення на скалярне число (\*);
   * спряжене число (унарний -);
   * порівняння двох комплексних чисел (==, !=);
   * присвоєння комплексних чисел (=).

Перевантажте (на вибір) частину унарних і бінарних операторів як функції, що належать класу, а другу частину — як дружні функції.

1. Продемонструвати роботу створеного класу та його функцій в основній частині програми.

**Порядок виконання роботи**

1. Комплексне число складається з двох частин: дійсної і уявної. Нехай *a* = *A* + *Bi*, *c* = *C* + *Di*. Операції, виконувані з комплексними числами:

* додавання *a + c* = (*A* + C) + (*B* + *D*)*i*;
* віднімання *a - c* = (*A* - C) + (*B* - *D*)*i*;
* множення *a \* c* = (*A* \* *C* – *B* \* *D*) + (*A* \* *D* + *B* \* *C*)*i*;
* множення на число *X \* c* = (*X* \* *C)* + (*X* \* *D*)*i*;

спряжене число *ã* = *A* - *Bi*;

1. Скласти опис класу *комплексне число* і реалізувати його програмно. Клас комплексного числа має у закритій частині містити змінні для збереження дійсної і уявної частин.

8.Клас комплексного числа має містити конструктор і деструктор, а також усі функції-оператори, необхідні для реалізації:

* операції додавання (операція “+”),
* віднімання (операція “-” та “унарний мінус”),
* множення комплексних чисел, у тому числі множення комплексного на звичайне число і множення звичайного числа на комплексне (операція “\*”),
* порівняння двох комплексних чисел (операції “==”, “!=”),
* обчислення спряженого числа (оператор “унарний -”),
* присвоєння одного комплексного числа іншому

(оператор “=”).

1. Програма має містити меню, яке дозволить перевірити основні функції, створені у програмі.

Зокрема, у програмі потрібно надати користувачу можливість введення комплексних чисел у текстові поля у форматі “-3+7і”, а також можливість вибрати операцію, яку може виконати користувач з комплексними числами. Результат операції слід виводити у текстове поле у вигляді “(-1 + 4і) + (7 - 6і) = 6 - 2і”.

Продемонструвати роботу створеної програми.

### Завдання № 5

### Основні завдання роботи

Розробити та реалізувати програмно ієрархію класів для роботи з рахунком вкладника банку. Для цього слід виконати такі завдання:

1. Розробити та реалізувати програмно базовий клас

Rahunok.

1. Розробити та реалізувати програмно базовий клас

Vkladnyk.

1. Розробити та реалізувати програмно похідний клас

RahunokVkladnyka.

### У головній частині програми продемонструвати роботу створеної ієрархії класів.

**Порядок виконання роботи**

1. Створити у програмі клас *Rahunok*, який зберігатиме такі дані: унікальний номер рахунка, суму на рахунку (початкове значення задати при створенні рахунка), нарахування відсотків.
2. Додати до класу функції: конструктора з параметрами, відображення рахунка, встановлення додаткових відсотків, перерахування грошей з урахуванням відсотків.
3. Створити у програмі клас *Vkladnyk,* який зберігатиме такі дані: серію та номер паспорту, прізвище, ім’я та по батькові.
4. Додати до класу конструктора з параметрами, функцію відображення даних про вкладника.
5. Створити у програмі похідний клас *RahunokVkladnyka* від попередніх двох класів. Цей клас має містити дані про максимальну суму, яку можна зняти за добу, мінімальний залишок на рахунку після зняття грошей з рахунка, пароль, а також функції: конструктор з параметрами, функцію виведення інформації про рахунок вкладника, функцію зміни пароля, функцію вкладення грошей, функцію зняття грошей. Урахуйте необхідну політику конфіденційності та безпеки даних банку.
6. У головній частині програми продемонструйте роботу створеної ієрархії класів на прикладі такого об’єкта:

RahunokVkladnyka Geits(123456789, “Bill Geits”,

“EC123456”, 50000, 500, 500);

**Завдання № 6**

### Основні завдання роботи

Розробити та реалізувати програмно код, у якому демонструється робота віртуальної функції. Для цього виконати такі завдання:

1. Розробити і реалізувати програмно клас “Число”.
2. Розробити ієрархію класів: *число* (базовий), *матриця*

(похідний).

1. Розробити та реалізувати програмно віртуальну функцію, яка обчислює факторіал числа, заданого в базовому класі, і використовується похідним класом, що містить масив цілих чисел, факторіали яких слід підрахувати та вивести у таблицю.

**Порядок виконання роботи**

1. Створити у програмі клас *Chislo*. В цьому класі задати закриту змінну у вигляді цілого додатного числа (типу *long*).

Визначити конструктор і деструктор цього класу.

Визначити віртуальну функцію *factorial()*, яка має обчислювати і повертати факторіал числа, яке передається йому як параметр.

1. Створити похідний клас *Matrix*, який успадковує *Chislo*. Цей клас в розділі *public* має містити одномірний масив 100 цілих додатних чисел, а успадковуване ціле число (вводиться з клавіатури) від базового класу має сенс розмірності цього масиву.

Визначити конструктор і деструктор цього класу.

Віртуальну функцію у похідному класі не перевизначати.

Проаналізуйте отриманий результат.

1. Числа масиву вводяться з клавіатури користувачем. У програмі написати функцію обчислення факторіала кожного числа масиву тієї розмірності, яка зберігається у змінній базового класу.

Продемонструвати роботу створених функцій в основній частині програми. Використати вказівники на базові класи.ло

**Завдання № 7**

### Основні завдання роботи

Розробити і реалізувати програмно шаблонний клас для подання розріджених одновимірних масивів. Розмір логічного масиву передавати через аргумент конструктора. Для цього виконати такі завдання:

1. Розробити і реалізувати програмно структуру класу розрідженого одновимірного масиву.
2. Реалізувати зберігання даних будь-якого типу *Т* у створеному класі*.*
3. Передбачити у класі родовий конструктор за замовчуванням, родовий конструктор копіювання і операцію присвоєння.
4. Реалізувати у класі операцію індексування, що повертає посилання на знайдений елемент у масиві. Якщо елемент із заданим індексом не знайдений, то операція повинна створити новий елемент з цим індексом і розмістити його у масив.
5. В основній частині програми продемонструвати використання створеного класу.

**Порядок виконання роботи**

Згідно з умовами завдання потрібно розробити дуже примітивний клас з мінімальною функціональністю. У реальних додатках такий клас буде містити додатково інші, складніші методи, наприклад, видалення з фізичного масиву елемента із заданим індексом.

1. За умовами завдання кожний елемент фізичного масиву повинен містити два поля: логічний індекс елемента і його значення. Тому написання програмного коду потрібно почати з розроблення класу для подання одного елемента фізичного масиву. Цей клас має бути шаблонним.

Необхідно вирішити, яку структуру обрати для збереження фізичного масиву. Зазвичай використовують лінійні списки, бінарні дерева або структури із хешуванням індексів.

Лінійний список має найгірші показники за часом пошуку інформації із заданим ключем (індексом), однак є найбільш простим для програмування.

Можна використати контейнерний клас *list* із бібліотеки шаблонів *STL,* який реалізований у вигляді двозв’язного списку, кожний елемент якого містить посилання на попередній та наступні елементи. Для використання цього класу слід підключити заголовний файл <*list*>.

У класі *list* є конструктор за замовчуванням, який створює клас нульової довжини. У кінець списку можна додати елемент за допомогою функції *push\_back*(). Доступ до будь-якого елемента списку здійснюється через ітератор — змінну типу *list<T>::iterator.* Ітератор можна розглядати як вказівник на елемент списку. Він використовується для перегляду списку у прямому чи зворотному напрямках. У першому випадку використовується операція інкремента, у другому — декремента.

У класі *list* також є два методи, що допомагають переглянути список: *begin*() повертає вказівник на початковий елемент, *end*() повертає вказівник на елемент, що слідує за останнім. Поточне значення ітератора у циклі порівнюється із значенням, отриманим від *end*(), за допомогою операції “!=”, оскільки за довільного розміщення в пам’яті сусідніх елементів операція “<” для адрес елементів втрачає сенс.

*Приклад.* Використання класу *list*:

#include <iostream> #include <list>

using namespace std; int main() {

list <char> v1;

v1.push\_back(‘A’);

v1.push\_back(‘B’);

v1.push\_back(‘C’);

list<char>:: iterator i = v1.begin(); list <char>:: iterator n = v1.end();

for (i; i != n; ++i) cout << \*i << ‘ ‘;

//вміст комірки пам’яті, на яку вказує і

cout << endl; return 0;

}

1. Створити клас-шаблон для подання розрідженого масиву, передбачивши поле для зберігання фізичного масиву елементів типу *list*.
2. Додати у клас-шаблон перевантажену операцію індексування *operator*[]().
3. Додати функцію *show*(), яка виводитиме дані про елемент списку.
4. У головній частині програми створити об’єкти заданого класу для різних типів даних (наприклад, *double*, *list*).

Продемонструйте роботу створених функцій в основній частині програми для різних типів даних: цілочислового, з плаваючою крапкою, подвійної точності.

**Завдання № 8**

### Основні завдання роботи

1. Набрати та запустити на виконання програмний код. Проаналізувати його.

class Vect { public:

Vect(char);

~Vect() {

delete [] p; }

int& operator [] (int i) { return p[i]; } void Print();

private: int\* p; char size;

};

Vect::Vect(char n) : size(n)

{

p = new int[size]; if (!p) {

cerr << "Error of Vect constructor" << endl; return ;

}

for(int i=0;i<size;++i) p[i]=int();

}

void Vect::Print()

{

for (int i = 0; i < size; ++i) cout<< p[i]<< " "; cout<< endl;

}

int main() {

Vect a(3);

a[0]=0;

a[1]=1;

a[2]= 2;

a.Print(); Vect a1(200);

a1[10] = 5; a1.Print(); return 0;

}

1. Реалізувати програмно перехоплення виявлених помилок у коді.

Реалізувати програмно перехоплення перериваннь у конструкторі та деструкторі. Реалізуйте програмно оброблення виняткових ситуацій.

**Порядок виконання роботи**

У програмі із завдання реалізовано клас *Vect*, призначений для створення і використання одновимірних масивів типу *іnt* довільного розміру. Розмір масиву не перевищує число 256. Проект розроблявся для “заліза” з обмеженням ємності оперативної пам’яті. Для зберігання у класі *Vect* інформації про розмір масиву програміст використав поле *char size*, попокладаючись на те, що для типу *char* виділяється один байт, а діапазон можливих значень для восьмирозрядного двійкового числа становить 0...255.

Якщо протестувати програму, то на екрані з’явиться запис:

0 1 2

Error of Vect constructor

1. Виконати покрокове виконання програми. Воно покаже, що помилка з’являється при спробі виконати оператор *a1[10] =*

Повідомлення про помилку виводиться тоді, коли операція

*new* не могла виділити пам’ять і повернула нульове значення вказівника *р*.

Тип *char* – це скорочення типу *signed char*, у якому старший біт використовується для зображення знака числа. Тому діапазон поданих чисел для типу *char* становить 128...+127.

Як тоді інтерпретується десяткове число 200? Перетворивши його у шістнадцяткове, отримаємо число С8 у двійковому еквіваленті 11001000. Старший розряд цього числа дорівнює 1. Це означає, що комп’ютер сприйме його як - 1001000.

Нагадаймо, що від’ємні значення зберігаються у вигляді додаткового коду, який отримується інверсією усіх розрядів з наступним додаванням 1. Тому після оберненого перетворення слід відняти 1, а потім проінвертувати усі розряди. У результаті отримаємо число -56.

Отже, помилка міститься у реалізації класу. Отримавши від’ємне число як ємність запитуваної пам’яті, операція *new* повертає 0.

* 1. Потім слід виправити помилку, не змінюючи тип. Для цього необхідно змінити код конструктора класу, а також основну частину програми.
  2. Найважливіша вимога до деструктора: жодне із виключень, яке могло б з’явитись у процесі роботи деструктора, не повинне залишити його межі.

Щоб виконати цю вимогу, слід дотримуватись таких двох правил:

1. ніколи не генерувати виключення у тілі деструктора за допомогою *throw*;
2. якщо дії, що виконуються з деструктурованим об’єктом складні і пов’язані з викликом інших функцій, то рекомендується інкапсулювати усі ці дії в деякому методі, наприклад *Destroy*(), і викликати цей метод з використанням блока try/catch.
3. Додати оброблення виняткових ситуацій у деструктор заданого класу.

Продемонструвати роботу програми з оброблення виняткових ситуацій.

**Завдання № 9**

### Основні завдання роботи

Реалізувати програмний додаток відображення графічних залежностей за допомогою даних, наведених у файлі. Для цього виконати такі завдання:

1. Розробіть та реалізуйте програмно клас функції.

Розробіть структуру і створіть файл, у якому зберігаються дані про математичну функцію, представлену значеннями по осі абсцис і ординат.

1. Реалізуйте у програмі знаходження та відображення на графіку мінімального і максимального значень серед усіх з файлу.

Реалізуйте програмно функцію запису значень зі створеного файлу до нового файлу у певному форматі, за якого функція замінює пробіли на табуляцію, кожне число записує в нотації з фіксованою точкою, з трьома знаками після коми, кожне числове значення займає 10 позицій (ширина поля), цілі числа мають бути виведені у вигляді “+10.000”.

**Порядок виконання роботи**

1. Створити у програмі клас *функція.* Як закриті змінні використайте динамічні масиви, які визначають значення математичної функції по осі абсцис і ординат відповідно. Додайте у клас конструктор і деструктор, а також функції вставки і вилучення, які повинні зчитувати дані з файлу і відображати дані на екрані відповідним чином.
2. Створити на диску файл, який містить три стовпчики цифр. Перший — значення по осі абсцис, два інші — по осі ординат для двох різних математичних функцій. Числа у стовпцях розділені пробілами.
3. Створити у програмі функцію вилучення. Вона повинна зчитувати дані з файлу, при цьому враховуючи, що числа відокремлені один від одного пробілами, а також те, що перший стовпчик цифр — це значення по осі абсцис (тобто їх слід зберегти у масиві для абсцис), другий (або третій) стовпчики — значення по осі ординат (їх слід зберігати в іншому масиві, призначеному для значень ординат). Оскільки у файлі зберігається інформація для двох функцій, то слід в основній частині програми створити два об’єкти класу *функція*: один для збереження даних про одну, інший — про другу функцію з файлу.
4. Додати до програми дружню функцію визначення максимального і мінімального значення з файлу та виведення цих значень на екран.

Додайте у програму функцію вставки. Вона повинна відобразити на екрані графік відповідної функції на осях координат.

1. Додайте у програму функцію, що зчитує дані з файлу і записує їх до іншого таким чином: замінює пробіли на табуляцію, кожне число записує в нотації з фіксованою краточкоюпкою, з трьома знаками після коми, кожне число займає 10 позицій (ширина поля), цілі числа мають бути виведені у вигляді “+10.000”.

Продемонструвати роботу усіх функцій створених класів в основній частині програми, для цього оголосивши попередньо необхідні об’єкти.

**Перелік тем:**

1. Методи та засоби проектування класів і створення об’єктів в С++.
2. Проектування і реалізація програми з масивами об’єктів, вказівниками та посиланнями на них.
3. Проектування і реалізація програми з перевантаженням функцій.
4. Проектування і реалізація програми з перевантаженням операторів (бінарних і унарних).
5. Проектування і опрацювання програми з віртуальними функціями.

# Проектування і опрацювання програми з родовими функціями і родовими класами.

# Проектування і опрацювання програми з обробленням виняткових ситуацій.

1. Проектування і опрацювання програми з власними маніпуляторами і власними функціями введення-виведення (створення власних маніпуляторів)
2. Проектування шаблонів класів засобами С++.