Лабораторна робота №3. ЗАСОБИ ОПРАЦЮВАННЯ ВИНЯТКОВИХ СИТУАЦІЙ В С++, ПЕРЕВАНТАЖЕННЯ ОПЕРАТОРІВ ДЛЯ КЛАСУ. РОБОТА З КОНСТРУКТОРАМИ ТА ПЕРЕВАНТАЖЕННЯ МЕТОДІВ КЛАСУ

МЕТА РОБОТИ: навчитись перехоплювати і обробляти різного роду виняткові ситуації, які виникають в процесі розроблення програм з використанням об’єктно-орієнтованого підходу, освоїти поняття конструктора та його призначення. Навчитись перевантажувати методи класу та оператори, здобути навики роботи з ними.

3.1. Програма роботи

3.1.1. Отримати завдання.

3.1.2. Написати програми відповідних класів, основну та відповідні допоміжні функції, згідно з вказівками до виконання роботи.

3.1.3. Підготувати власні коректні вхідні дані (вказати їх формат і значення) і проаналізувати їх.

3.1.4. Оформити електронний звіт про роботу та захистити її.

3.2. Вказівки до виконання роботи

3.2.1. Студент, згідно з індивідуальним номером, вибирає своє завдання з розд. 3.5 і записує його до звіту.

3.2.2. Оголошення класу (структури), основну та відповідні допоміжні функції необхідно запрограмувати так, як це показано у розд. 3.4.

3.2.3. Власних вхідних даних необхідно підготувати не менше двох комплектів. Їхні значення мають бути коректними, знаходитися в розумних межах і відповідати тим умовам, які стосуються індивідуального завдання.

3.2.4. Звіт має містити такі розділи:

* мету роботи та завдання з записаною умовою задачі;
* коди всіх використовуваних .h і .ccp файлів, а також пояснення до них;
* результати реалізації програми;
* діаграму класів та діаграму варіантів використання з поясненням;
* висновки, в яких наводиться призначення програми, обмеження на її застосування і можливі варіанти удосконалення, якщо такі є.

3.3. Теоретичні відомості

**Засоби опрацювання виняткових ситуацій**

***Оброблення винятків*** – це системні засоби, за допомогою яких програма

може справитися з помилками часу виконання. Керування С++ – механізмом

оброблення винятків тримається на трьох ключових словах: **try**, **catch** і **throw**. Оператори **try**, **throw** та **catch** є пов’язаними між собою. Код програми, який може викидати значення (об’єкт) за допомогою оператора **throw**, необхідно розмістити у блоці **try**. Безпосередньо за блоком **try** розміщується один або декілька **catch** – обробників. При викиданні винятка спрацює тільки один з обробників, тип якого відповідає типу викинутого значення. Нижче наведено схему розміщення операторів керування винятками:

try {

// блок генерування виняткової ситуації (throw вираз)

}

catch (type1 arg) {

1 – й блок перехоплення виняткової ситуації

}

catch (type2 arg) {

2 – й блок перехоплення виняткової ситуації

}

catch (typeN arg) {

N – й блок перехоплення виняткової ситуації

}

Функції **exit()** і **abort()** забезпечують завершення роботи програми. Для використання цих функцій потрібно підключити до програми заголовок <**cstdlib**>. Функцію exit() використовують для зупинки програми під час виникнення непоправної помилки, яка робить подальше її виконання безглуздим або небезпечним. Її прототип має такий вигляд: **void** **exit**(**int** *status*);

Функція **abort**() викликає негайне завершення роботи програми, проте вона не повертає операційній системі ніякої інформації про статус завершення роботи програми і не виконує стандартної ("правильної") послідовності дій під час зупинки програми. Її прототип має такий вигляд: **void** **abort**();

**Перехоплення винятків базового класу.** Якщо потрібно перехоплювати винятки як базового, так і похідного типів, то у catch – послідовності catch – настанову для похідного типу необхідно помістити перед catch – настановою для базового типу. Інакше catch – вираз для базового класу перехоплюватиме крім "своїх" і винятки всіх похідних класів. Розглянемо, наприклад, такий код програми: демонстрація механізму перехоплення винятків базових і похідних типів:

class bClass {

};

class dClass: public bClass {

};

void main() {

                 dClass derived;

  try { throw derived; }

  catch(bClass ObjB) { cout << "Перехоплення винятку базового класу" << endl; }

catch(dClass ObjD) { cout << "Це перехоплення не відбудеться" << endl; }

}

  Оскільки тут об'єкт derived – це об'єкт класу dClass, який виведено з базового класу bClass, то виняток типу derived завжди перехоплюватиметься першим catch – виразом; друга ж catch – настанова при цьому ніколи не виконається. Одні компілятори відреагують на такий стан речей застережним повідомленням, інші можуть видати повідомлення про помилку. У будь – якому випадку, щоб виправити ситуацію, достатньо поміняти порядок слідування цих **catch** – настанов на протилежний.

**Повторне генерування винятку.** Для того, щоби повторно згенерувати винятки в його обробнику, необхідно використовувати throw – настанову без вказання типу винятку. У цьому випадку поточний виняток передається в зовнішню try/catch – послідовність. Найчастіше причиною для такого виконання настанови throw слугує прагнення мати доступ до одного винятку декільком обробникам. Наприклад, перший обробник винятків повідомлятиме про один аспект винятку, а другий – про щось інше. Винятки можна повторно згенерувати тільки в catch – блоці (або в будь – якій функції, яка викликається з цього блоку). При повторному генеруванні винятку сам виняток не перехоплюватиметься тією тією ж **catch**–настановою. Він пошириться на найближчу try/catch – послідовність. Повторне генерування винятку char \* продемонстровано у наведеному нижче коді програми:

void Xhandler() {

try {

 throw "Привіт";  // Генерує винятки типу char \*

}

catch(char \*) {  // Перехоплює винятки типу char \*

 cout << "Перехоплення винятку у функції Xhandler" << endl;

throw; //Повторне генерування винятку типу char \*, яке перехоплюватиметься поза функцією Xhandler.

}

}

void main() {

cout << "Початок" << endl;

try {

Xhandler();

}

catch(char \*)

{

cout << "Перехоплення винятку у функції main()" << endl;

}

}

***Внаслідок виконання ця програма відображає на екрані такі результати:***

Початок.

Перехоплення винятку у функції Xhandler.

Перехоплення винятку у функції main().

Кінець програми

**Перевантаження операторів для класу**

Перевантаження операцій здійснюється за допомогою методів спеціального виду (функцій – операцій) і підлягає наступним правилам:

* при перевантаженні операцій зберігаються кількість аргументів, пріоритети операцій і правила асоціації (справа наліво і зліва направо), що використовуються в стандартних типах;
* для стандартних типів даних перевизначити операції неможливо;
* функції-операції не можуть мати аргументів по замовчуванню;
* функції-операції не наслідуються (за виключенням =);
* функції-операції не можуть визначатися як static.

Функцію – операцію можна визначити трьома способами:

* вона може бути методом класу;
* вона може бути дружньою функцією;
* вона може бути звичайною функцією.

У двох останніх випадках функція повинна приймати хоча б один аргумент, який має тип класу, вказівник або посилання на клас.

Функція – операція містить ключове слово operator, за яким слідє знак перевизначеної операції:

тип operator операція (список параметрів) { тіло функції }

**Перевантаження унарних операцій:**

Унарні операції – це такі, які мають тільки один операнд (операнд – це змінна, на яку діє операція).

Прикладом унарних операцій є операціїї інкремента та декремента «++» та «--» а також унарний мінус, наприклад -35.

Для демонстрації перевантаження унарного оператора показано унарні оператори інкремента та декремента у префіксних, та постфіксних формах для класу Point:

class Point

{

public:

// Оголошення префіксної та постфіксної форми інкремента.

Point& operator++(); // Prefix increment operator.

Point operator++(int); // Postfix increment operator.

// Оголошення префіксної та постфіксної форми декремента .

Point& operator--(); // Prefix decrement operator.

Point operator--(int); // Postfix decrement operator.

// Визначення конструктора

Point() { \_x = \_y = 0; }

// Визначення доступу

int x() { return \_x; }

int y() { return \_y; }

private:

int \_x, \_y;

};

// Оголошення унарної операції “++” – префіксна форма

Point& Point::operator++()

{

\_x++;

\_y++;

return \*this;

}

// Оголошення унарної операції “++” – постфіксна форма

Point Point::operator++(int)

{

Point temp = \*this;

++\*this;

return temp;

}

// Оголошення унарної операції “--” – префіксна форма

Point& Point::operator--()

{

\_x--;

\_y--;

return \*this;

}

// Оголошення унарної операції “--” – постфіксна форма

Point Point::operator--(int)

{

Point temp = \*this;

--\*this;

return temp;

}

int main()

{

}

**Перевантаження бінарних операцій.** Бінарні операції можуть бути перевантажені таким же чином, як і унарні. Перевантаження бінарної операції як члена класу є нестатичною операторною функцією з одним параметром (другим, перший є вказівником this). Наприклад:

class A {

int x;

public:

A(int y = 0 : x(y)) {}

A operator-(const A&) const;

};

A A::operator-(const A&a) const

{return A(x-a.x);}

void main()

{

A a(1), b(2), c;

c = b-a;

}

**Робота з конструкторами та перевантаження методів класу**

Конструктор призначений для ініціалізації об’єкта і викликається автоматично при його створенні. Нижче перераховано основні властивості конструкторів:

* конструктор не повертає значення, навіть типу **void.** Неможливо отримати вказівник на конструктор;
* клас може мати декілька конструкторів з різними параметрами для різних видів ініціалізації (при цьому використовується механізм перевантаження);
* параметри конструктора можуть мати будь – який тип, окрім цього ж класу. Можна створювати параметри по замовчуванню, причому їх може містити тільки один із конструкторів;
* якщо програміст не вказав жодного із конструкторів, компілятор створює його автоматично. Такий конструктор викликає конструктори по замовчуванню базових класів;
* конструктори не наслідуються;
* конструктори не можна описувати з модифікаторами const, virtual, static;
* конструктори глобальних об’єктів викликаються до функції main. Локальні об’єкти створюються, як тільки стає активною їх область дії;
* конструктор викликається автоматично при створенні об’єкта.

**Перевантаженням** (overloading) називається практика надання декількох значень оператору або функції. Вибір конкретного значення залежить від типу аргументів, що отримує оператор чи функція, або від їх кількості. Наприклад:

class A{  
    public:  
        void foo(int x) { cout << "1\n";}  
        void foo(double x) { cout << "2\n";}  
        void foo(int x, int y) { cout << "3\n";}  
 };  
 int main()  
 {  
    A a;  
    a.foo(1); //виклик методу 1  
    a.foo(1.0); //виклик методу 2  
    a.foo(1, 2); //виклик методу 3  
 }

3.4. Зразок виконання роботи

**Код програми:**

Рис. 1. Вигляд вікна в результаті компіляції програми

3.5. Індивідуальні завданняНачало формы

1. Визначити клас СКЛАД з масивом об’єктів ТОВАР. Товар характеризується назвою, кількістю та ціною. Визначити методи роботи з цими полями даних та перевантажити операції (++,--,=) для збільшення, зменшення к-сті товару та заміни одного товару на інший. Перевантажити метод пошуку товару за різними критеріями. Передбачити обробку виняткових ситуацій при встановленні та читанні числових полів класу ТОВАР.
2. Реалізувати клас СЛОВНИК, який містить набір українських слів та їх переклад англійською мовою. Перевантажити операції для занесення слів у словник та їх видалення (+, -), перевантажити операції порівняння(<, >=) двох слів (об’єктів класу СЛОВО) та перевантажити метод у класі СЛОВНИК для сортування словника по алфавіту (за українським перекладом або за англійським). Описати методи для виводу та пошуку слів. Передбачити обробку виняткових ситуацій при спробі звернення то поля-вказівника, який не ініціалізований.
3. Створити клас КАЛЕНДАР, який містить: день, місяць, рік та подію. Визначити необхідні конструктори, методи, деструктор. Перевантажити операцію (=) для встановлення дати, операції (+=, -=) для додавання та віднімання днів, місяців, років до заданої дати, операції (==, !=, >, >=, <, <=) для порівняння дат. Перевантажити метод пошуку події за різними критеріями. Передбачити обробку виняткових ситуацій при встановленні та читанні числових полів класу КАЛЕНДАР.
4. Визначити клас ЗАПИСНИК та ЗАПИС, який містить прізвище та ім’я, номер телефону, день народження. Визначити методи роботи з цими полями даних та перевантажити операції для занесення інформації у записник та видалення (+, -). Перевантажити метод пошуку за прізвищем та за датою народження. Відсортувати записник за датами народження (перевантажит оператор = ).Передбачити обробку виняткових ситуацій при встановленні та читанні полів класу ЗАПИСНИК.
5. Створити клас ЛІКИ, які містять: назву, кількість одиниць товару, вартість. Визначити вартість ліків за рецептом. Перевантажити операції (+=, -=) для зміни кількості одиниць товару, (+, -) для зміни вартості товару. Первантажити методи вартості ліків за рецептом та пошуку ліків за різними критеріями. Передбачити обробку виняткових ситуацій при встановленні та читанні числових полів класу ЛІКИ.
6. Створити клас ДІЙСНИХ ЧИСЕЛ. Визначити необхідні конструктори, методи, деструктор. Передбачити в класі методи знаходження найбільшого та найменшого дійсного числа. Перевантажити операцію (+) для додавання чисел, (-) для віднімання, (\*) для добутку, (/) для ділення. Перевантажити метод пошуку дійсного числа за такими критеріями: найбільше та найменше число. Передбачити обробку виняткових ситуацій при встановленні та читанні значень полів цього класу.
7. Створити клас ОСОБА, яка містить: прізвище, ім’я, по батькові, стать, дату народження. Перевантажити операцію (>>) потокового введення та (<<) виведення об’єктів. Перевантажити метод пошуку особи за різними критеріями. Передбачити обробку виняткових ситуацій при встановленні та читанні числових полів класу ОСОБА.
8. Створити клас АВТОМОБІЛЬ, який має марку, колір, потужність, об’єм двигуна. Перевантажити операцію (+=) для нарощування потужності, (>>) потокову операцію введення та (<<) виведення об’єктів. Перевантажити метод пошуку автомобіля за різними критеріями. Передбачити обробку виняткових ситуацій при встановленні та читанні числових полів класу АВТОМОБІЛЬ.
9. Створити клас РОЗРАХУНКОВА КНИГА ЕЛЕКТОЕНЕРГІЇ, яка містить: вартість кіловат – години, дату та сплачену суму. Перевантажити операцію (++) для збільшення значення показників на 1 та (-=) для зменшення значення показників на задану величину. Перевантажити метод пошуку показників за датою. Передбачити обробку виняткових ситуацій при встановленні та читанні значень полів цього класу.
10. Створити клас КОЛО, який містить: площу, довжину кола, член класу – радіус R. Перевантажити потокову операцію введення (>>) та (<<) виведення об’єктів. Перевантажити методи розрахунку площі та виведення даних про фігуру. Передбачити обробку виняткових ситуацій при встановленні та читанні числових полів класу КОЛО.
11. Створити клас ПРЯМОКУТНИК, який містить: довжини сторін, діагоналей, периметр, площу, члени класу – довжини сторін a і b. Перевантажити операцію (++) для пропорційного збільшення довжини сторін, (--) для зменшення довжини сторін. Перевантажити методи розрахунку площі та виведення даних про фігуру. Передбачити обробку виняткових ситуацій при встановленні та читанні значень полів цього класу.
12. Створити клас ЗАРПЛАТА, який містить дані про співробітників: прізвище, ім’я, по батькові, номер телефону, посаду та зарплату. Перевантажити операцію (+=), якщо видано премію, (-), якщо співробітник звільнився та (+), якщо прийнято нового співробітника. Перевантажити метод пошуку співробітника за різними критеріями. Передбачити обробку виняткових ситуацій при встановленні та читанні числових полів класу ЗАРПЛАТА.
13. Створити клас БІБЛІОТЕКА, яка містить: назву книги, автора, рік видання та видавництво. Перевантажити операції (+, -) для зміни кількості книг у бібліотеці. Перевантажити метод пошуку книги за прізвищем автора, назвою книги, видавництвом, роком видання. Передбачити обробку виняткових ситуацій при встановленні та читанні числових полів класу БІБЛІОТЕКА.
14. Створити клас КВАДРАТ, який містить: діагональ, периметр, площу, члени класу – довжина сторони. Перевантажити потокову операцію введення (>>) та (<<) виведення об’єктів. Перевантажити методи розрахунку площі і виведення даних про фігуру. Передбачити обробку виняткових ситуацій при встановленні та читанні числових полів класу КВАДРАТ.
15. Створити клас ТЕЛЕФОННА КНИГА, яка містить дані про абонентів. Перевантажити операцію (+) для збільшення абонентів, операцію (=) для встановлення номеру телефона. Перевантажити метод пошуку номера телефону за прізвищем абонента. Передбачити обробку виняткових ситуацій при встановленні та читанні числових полів класу ТЕЛЕФОННА КНИГА.

3.6. Контрольні запитання

1. Що таке механізм перевантаження?
2. Назвіть причини необхідності перевантаження конструкторів?
3. Як можна перевантажити метод класу?
4. Який за замовчуванням доступ до елементів структури?
5. Чи може бути конструктор класу бути порожнім, обґрунтуйте свою думку.
6. Що таке деструктор класу, коли він викликається?
7. Опишіть створення перевантажених методів та конструкторів класу.
8. Якщо змінити порядок слідування параметрів у методі, чи необхідно створювати відповідний перевантажений метод? Чому?
9. Чи може бути у класі перевантажений деструктор? Чому?
10. Які основні принципи ООП Ви знаєте? Що вони означають?
11. Що таке механізм перевантаження оператора?
12. Назвіть причини необхідності перевантаження операторів?
13. Як можна перевантажити оператор для класу?
14. Які оператори перевантажувати не можна?
15. Яка різниця між перевантаженням бінарних та унарних операторів?
16. Опишіть синтаксис перевантаження унарних операторів класу.
17. У чому полягає специфіка перевантаження операторів зсуву
18. У яких випадках перевантаження операцій визначається як дружня функція до класу?
19. Коли застосовують виключення і які для цього засоби Ви знаєте?
20. Що таке обробник виняткових ситуацій (виключень)?
21. Які є форми представлення типу обробника виключень?
22. Як повторно згенерувати виключення, якщо обробник не може повністю обробити дане виключення?
23. Яка функція викликається, якщо виключення згенеровано, але не перехвачено?
24. Яка користь від віртуальних функцій при обробці виняткових ситуацій в класах виключень?
25. Що таке специфікація виключень?
26. Чи можна виключення використовувати в конструкторах і деструкторах? Обґрунтуйте.