Тема
Створення та використання класів

Мета
Навчитися створювати класи, використовувати конструктори для ініціалізації об’єктів, опанувати принципи створення функцій-членів. Навчитися використовувати різні типи доступу до полів та методів класів.

Теоретичні відомості
Ідея класів має на меті дати інструментарій для відображення будови об’єктів реального світу - оскільки кожен предмет або процес має набір характеристик (відмінних рис) іншими словами, володіє певними властивостями і поведінкою. Програми часто призначені для моделювання предметів, процесів і явищ реального світу, тому в мові програмування зручно мати адекватний інструмент для представлення цих моделей.

Клас є типом даних, який визначається користувачем. У класі задаються властивості і поведінка будь-якого предмету або процесу у вигляді полів даних (аналогічно до того як це є в структурах) і функцій для роботи з ними. Створюваний тип даних володіє практично тими ж властивостями, що і стандартні типи.

Конкретні величини типу даних «клас» називаються екземплярами класу, або об'єктами.

Об’єднання даних і функцій їх обробки з одночасним приховуванням непотрібної для використання цих даних інформації називається інкапсуляцією (encapsulation). Інкапсуляція підвищує ступінь абстракції програми: дані класу і реалізація його функцій знаходяться нижче рівня абстракції, і для написання програми з використанням вже готових класів інформації про них (дані і реалізацію функцій) не потрібно. Крім того, інкапсуляція дозволяє змінити реалізацію класу без модифікації основної частини програми, якщо інтерфейс залишився тим самим (наприклад, при необхідності змінити спосіб зберігання даних з масиву на стек). Простота модифікації, як уже неодноразово зазначалося, є дуже важливим критерієм якості програми.

Опис класу в першому наближенні виглядає так:

```
class <ім'я> {
[private:]
    <Опис прихованих елементів>
public:
        <Опис доступних елементів>
}; //Опис закінчується крапкою з комою.
```

Замість ключового слова class може бути також struct. З точки зору C++ між class та struct практично немає різниці. Невеликі відмінності наступні:

1. Для class дані-члени, функції-члени та батьківські класи за замовчуванням приватні (private).
2. Для struct дані-члени, функції-члени та батьківські класи за замовчуванням публічні (public).

Специфікатор доступу private і public керують видимістю елементів класу. Елементи, описані після службового слова private, доступні тільки всередині класу. Цей вид доступу прийнятий у класі за замовчуванням. Інтерфейс класу описується після специфікатора public. Дія будь-якого специфікатора поширюється до наступного специфікатора або до кінця класу. Можна задавати кілька секцій private і public, їх порядок значення не має.

Поля класу:

- можуть мати будь-який тип, крім типу цього ж класу (але можуть бути вказівниками або посиланнями на цей клас);

- можуть бути описані з модифікатором const, при цьому вони ініціалізуються тільки один раз (за допомогою конструктора) і не можуть змінюватися;
- можуть бути описані з модифікатором static (розглядається в наступних лабораторних).

Починаючи із C++11 допускається ініціалізація полів при описі класу.

Конструктори

Конструктор призначений для ініціалізації об’єкту і викликається автоматично при його створенні. Автоматичний виклик конструктора дозволяє уникнути помилок, пов’язаних з використанням неініціалізованих змінних. Нижче наведені основні властивості конструкторів:

- Конструктор не повертає жодного 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).
- Конструктор без параметрів називається конструктором за замовчуванням.
- Параметри конструктора можуть мати будь-який тип, крім цього ж класу. Можна задавати значення параметрів за замовчуванням. Їх може містити тільки один з конструкторів.
- Якщо програміст не вказав жодного конструктора, компілятор створює його автоматично. Такий конструктор викликає конструктори за замовчуванням для полів класу і конструктори за замовчуванням базових класів. У разі, коли клас містить константи або посилання, при спробі створення об'єкту класу буде видана помилка, оскільки їх необхідно ініціалізувати конкретними значеннями, а конструктор за замовчуванням цього робити не вміє.
- Починаючи із C++11 конструктори можна наслідувати за допомогою ключового слова using. Але при тому наслідуються усі конструктори батьківського класу.
- Конструктори не можна описувати з модифікаторами const, virtual і static. Але можна описувати із модифікатором noexcept, що позначатиме, що він ніколи не генерує виключень.
- Конструктори глобальних об’єктів викликаються до виклику функції main. Локальні об’єкти створюються, як тільки стає активною область їх дії. Конструктор запускається і при створенні тимчасового об'єкта (наприклад, при передачі об’єкта з функції).
- Конструктор викликається, якщо в програмі зустрілася будь-яка із синтаксичних конструкцій:

```
// Список параметрів може бути порожнім або відсутнім (якщо є конструктор за замовчуванням)
ім’я_класу ім’я_об’єкту[(список параметрів)];

// Створюється об’єкт без імені (список може бути порожнім але присутнім)
ім’я_класу(список параметрів);

// Створюється об’єкт, викликаючи конструктор із параметром такого ж типу, як
//  результат виразу
ім’я_класу ім’я_об’єкту = вираз;
```

Завдання

1. Створити клас відповідно до варіанту.
2. При створенні класу повинен бути дотриманий принцип інкапсуляції.
3. Створити конструктор за замовчуванням та хоча б два інших конструктори для початкової ініціалізації об’єкта.
4. Створити функції-члени для задавання та зчитування значень полів (getters/setters)
5. Створити інші функції члени згідно з варіантом.
6. Продемонструвати можливості класу написавши для нього модульні тести для Google Test.
7. У звіті до лабораторної намалювати UML-діаграму класу, яка відповідає варіанту.
8. Звіт має містити тексти програм, тестів та результат виконання тестів.

Варіанти завдань

Варіант	Завдання
1	Клас CFraction– звичайний дріб. Клас повинен містити функції-члени, які реалізовують: • Додавання • Віднімання • Множення • Ділення • Скорочення дробу • Обертання дробу
2	Клас CComplex – комплексне число. Клас повинен містити функції-члени, які реалізовують: • Додавання • Віднімання • Множення • Піднесення до n-го степеня • Отримання кореня n-го степеня • Представлення в тригонометричній формі
3	Клас CBigNumber – велике ціле число (розміром 128 біти – як два long long числа). Клас повинен містити функції-члени, які реалізовують: • Додавання • Віднімання • Зчитування із двох значень типу long long (старші та молодші 64 біти) • Отримання значень у вигляді пари значень типу long long
4	Клас CFixedPointNumber – число із фіксованою крапкою з точністю до сотих (після крапки). Клас повинен містити функції-члени, які реалізовують: • Додавання • Віднімання • Множення • Ділення • Зчитування із значення типу double • Отримання значення типу double
5	Класс CPoint – точка в просторі Клас повинен містити функції-члени, які реалізовують: • Зсув у просторі на задані значення координат • Збільшення всіх координат в певну кількість разів (задається параметром) • Відстань до початку координат • Відстань до іншої точки (задається параметром) • Зчитування із сферичних координат • Отримання значень сферичних координат
6	Клас CVector – вектор в просторі. Клас повинен містити функції-члени, які реалізовують: • Додавання векторів • Віднімання • Скалярний добуток • Векторний добуток • Добуток вектора на скаляр • Обчислення довжини вектора
7	Клас CSphere – куля у просторі. Клас повинен містити функції-члени, які реалізовують: • Знаходження площі кулі

	• Знаходження об’єму кулі • Перенесення кулі в просторі на задану різницю координат • Збільшення радіусу кулі у задану кількість разів
8	Клас CTriangle – трикутник на площині (задаються довжини трьох сторін). Клас повинен містити функції-члени, які реалізовують: • Знаходження площі трикутника • Знаходження трьох кутів • Знаходження периметра • Знаходження трьох висот • Збільшення одразу всіх трьох сторін трикутника на константу • Перевірка чи трикутник є прямокутний
9	Клас CRectangle – прямокутник на площині (задаються довжини сторін) Клас повинен містити функції-члени, які реалізовують: • Знаходження площі прямокутника • Знаходження периметру прямокутника • Знаходження співвідношення сторін • Обертання прямокутника (зміна сторін) • Збільшення одразу всіх сторін на константу
10	Клас CPolynom2 – лінійний двочлен вигляду (ax+c). Клас повинен містити функції-члени, які реалізовують: • Знаходження значення виразу для заданого x • Знаходження значення похідної в заданій точці x. • Знаходження визначеного інтегралу на заданому проміжку • Знаходження кореня рівняння • Додавання двох поліномів • Віднімання двох поліномів
11	Клас CPolynom3 – квадратичний тричлен (ax^2+bx+c). Клас повинен містити функції-члени, які реалізовують: • Знаходження значення виразу для заданого x • Знаходження значення похідної в заданій точці x. • Знаходження визначеного інтегралу на заданому проміжку • Знаходження коренів рівняння • Додавання двох поліномів • Віднімання двох поліномів
12	Клас CPolynom4 – кубічний чотиричлен (ax^3+bx^2+cx+d). Клас повинен містити функції-члени, які реалізовують: • Знаходження значення виразу для заданого x • Знаходження значення похідної в заданій точці x. • Знаходження визначеного інтегралу на заданому проміжку • Знаходження коренів рівняння • Додавання двох поліномів • Віднімання двох поліномів
13	Клас CBitField – бітове поле (для 32 бітів). Клас повинен містити функції-члени, які реалізовують: • Повернення значення біта по номеру • Встановлення значення біта по номеру • Кількість встановлених бітів • Побітове І • Побітове АБО • Побітове Виключне АБО
14	Клас CTime – час. Реалізовує роботу з часом – години, хвилини, секунди. Години та хвилини мають бути цілочисельні, а секунди – дійсне число. Клас повинен містити функції-члени, які реалізовують:

	• Збільшення часу на задану кількість секунд (дійсними числами) • Збільшення часу у задану кількість разів (дійсними числами) • Додавання двох об’єктів типу CTime. Результат – час, який дорівнює сумі. • Різниця в секундах з іншим часом заданим через параметр • Порівняння з іншим часом заданим через параметр (функція порівняння має вертати -1, 0 чи 1) • Приведення часу до секунд (години*3600 + хвилини*60 + секунди)
15	Клас CDate – дата. Реалізовує роботу із датою – рік, місяць, день. Клас повинен містити функції-члени, які реалізовують: • Різницю в днях між двома датами • Збільшення дати на задану кількість днів • Зменшення дати на задану кількість днів • Повернення кількості днів у місяці.