Тема
Шаблони функцій та класів.

Мета
Навчитись створювати шаблони функцій та класів, використовувати різні види параметрів шаблонів, створювати екземпляри шаблонів.

Теоретичні відомості
Шаблони є основою для узагальненого програмування на C++. Як строго типізована мова, C++ вимагає, щоб усі змінні мали певний тип, або явно оголошений програмістом, або виведений компілятором. Однак багато структур даних і алгоритмів виглядають однаково незалежно від того, з яким типом вони працюють. Шаблони дозволяють визначати операції класу чи функції, а користувачу дозволяють вказувати, з якими конкретними типами ці операції мають працювати.

По суті шаблон визначає:

- родину класів (шаблон класу)
- родину функцій (шаблон функцій)
- псевдонім до родини типів (шаблон-псевдонім) – починаючи із C++11
- родину змінних (шаблон змінних) – починаючи із C++14

Шаблони мають один або кілька параметрів. Існує три види параметрів:

1. параметри шаблону, що позначають тип (type template parameters);
2. параметри шаблону, що позначають не тип (non-type template parameters);
3. шаблонні параметри шаблонів (template template parameters).

Визначення та використання шаблонів
Шаблон — це конструкція, яка ґенерує звичайний тип, функцію або змінну (починаючи із C++11) під час компіляції на основі аргументів, які користувач надає для параметрів шаблону. Наприклад, ви можете визначити шаблони функцій так:

```
template <typename T>
T GetMinimum(T left, T right)
{
    return left < right ? left : right;
}

template <class T>
T GetMaximum(T left, T right)
{
    return left < right ? right : left;
}
```

Рис. 1. Визначення шаблонів функцій

Наведений вище код описує шаблони функцій та містить один параметр типу T. Функція повертає значення цього типу, а також має два параметри (left і right) також цього типу. Параметр шаблону може мати будь-яку назву, хоча часто використовують одну велику літеру. Ключове слово typename означає, що цей параметр позначає тип. Замість typename також може бути ключове клово class. В даному контексті вони мають однакове значення. Коли функція викликається, компілятор замінить кожен екземпляр T назвою конкретного типу, який або вказано користувачем, або виведений компілятором. Процес генерації класу, функції чи змінної з шаблону називається інстанціюванням шаблону. GetMinimum<int> є екземпляром шаблону GetMinimum<T>.

В іншому місці програми користувач може оголосити екземпляр шаблону, який спеціалізується типом int. Припустимо, що get_a() і get_b() є функціями, які повертають int:

```
int a = get_a();
int b = get_b();
int minimum = GetMinimum<int>(a, b);
int maximum = GetMaximum<int>(a, b);
```

Рис. 2. Інстанціювання шаблону функцій із явним позначенням типу

Однак компілятор може вивести тип T з аргументів a і b, тому можна викликати функції як звичайні:

```
int minimum = GetMinimum(a, b);
int maximum = GetMaximum(a, b);
```

Рис. 3. Інстанціювання шаблону із автоматичним виведенням типу

Коли компілятор стикається з такими викликом, він ґенерує нову функцію, у якій кожне входження T у шаблоні замінюється на int:

```
int GetMinimum(int left, int right)
{
    return left < right ? left : right;
}

int GetMaximum(int left, int right)
{
    return left < right ? right : left;
}
```

Рис. 4. Зґенеровані функції із шаблонів

Правила того, як компілятор виконує дедукцію типу в шаблонах функцій, базуються на правилах для звичайних функцій.

Параметри шаблону, що позначають тип

Такі параметри оголошуються за допомогою ключевих слів typename або class. Кілька параметрів відділяються символом кома (,). Кількість параметрів не обмежується. Параметри можуть мати значення за замовчуванням – позначаються через символ ‘=’ та назву типу за замовчуванням:

```
template <typename TFirst, typename TSecond, typename TThird = int>
class Foo {};
```

Рис. 5. Шаблон класу із трьома параметрами, що позначають тип

Будь-який вбудований чи визначений користувачем тип може бути використаний, як параметр шаблону.

Починаючи із C++11 можна використовувати оператор три крапки (...), щоб визначити шаблон, який приймає нуль або довільну кількість параметрів на позначення типу.

```
template<typename... Types> class class_template_with_any_number_of_type_parameters;

class_template_with_any_number_of_type_parameters<> instance1;
class_template_with_any_number_of_type_parameters<int> instance2;
class_template_with_any_number_of_type_parameters<float, bool> instance3;
```

Рис. 6. Шаблон класу, який приймає довільну кількість параметрів, що позначають тип

Параметри шаблону, що позначають не тип

Шаблони у C++ підтримують також памаметри, що позначають не тип. Такі параметри ще називаються параметрами-значеннями.

Наприклад, можна вказати константне ціле значення для позначення розміру масиву, як в прикладі нижче:

```
template<typename TElementType, size_t Size>
class CMyArray
{
public:
    CMyArray() { ... }

private:
    TElementType m_data[Size];
};
```

Рис. 7. Шаблон класу із параметром, що позначає ціле число

Зверніть увагу, що значення Size типу size_t передається у шаблон як константне (const, constexpr) та має бути відоме під час компіляції. Наприклад:

```
CMyArray<MyClass*, 10> arr;
```

Рис. 8. Інстанціація шаблону класу CMyArray для створення змінної arr.

Інші види значень, як-от вказівники та посилання, можуть бути параметрами шаблону. Наприклад, можна передати вказівник на функцію чи функційний об’єкт, щоб підмінити певні операції у коді шаблону.

Шаблони як параметри шаблонів

Шаблон може бути параметром шаблону. Наприклад, можемо мати функцію, що створює різні типи контейнерів:

```
#include <vector>
#include <list>

template<template<typename TElementType> typename ContainerType>
ContainerType<int> GetData() {
      ContainerType<int> result;
      result.emplace_back(1);
      result.emplace_back(2);
      return result;
}

int main() {
      auto myVector = GetData<std::vector>();
      auto myList = GetData<std::list>();

      return 0;
}
```

Рис. 9. Приклад шаблону функції, що має параметром інший шаблон

Завдання

1. Змінити реалізацію класу із лабораторної роботи №8 відповідно до варіанту, зробивши його шаблоном класу.
2. Переконатись, що у класі є визначений оператор порівняння (==).
3. Переконатись у правильності роботи різних інстанціацій шаблону відповідно до варіанту.
4. Продемострувати роботу шаблону класу таким чином (на вибір):
    a. За допомогою тестів Google Test.
    b. За допомогою інтерактивної програми із демострацією її роботи.
5. Написати шаблон функції FindElementInArray(), яка буде приймати масив елементів будь-якого типу та елемент, який потрібно знайти в цьому масиві. Функція повинна повернути індекс першого входження елемента в масив, або розмір масиву, якщо елемент не знайдено. Протестувати функцію на будь-якому вбудованому типі, а також розробленому відповідно до п.1 класі.
6. Оформити звіт до лабораторної роботи.

Варіанти завдань

Варіант	Завдання
1	Шаблон класу CMatrix<typename T, size_t ColsCount, size_t RowsCount> – двовимірна матриця значень типу T. ColsCount – кількість колонок, RowsCount – кількість рядків. Протестувати шаблон на таких типах: • int • double • CFraction клас (розроблений на лабораторній роботі №3)
2	Шаблон класу CStaticArray<typename T, size_t Size> – одновимірний масив значень типу T. Size – розмір масиву. Протестувати шаблон на таких типах: • usigned int • float • CComplex клас (розроблений на лабораторній роботі №3)
3	Шаблон класу CSet<typename T> – множина значень типу T. Протестувати шаблон на таких типах: • long • long double • CBigNumber клас (розроблений на лабораторній роботі №3)
4	Шаблон класу CMultiSet<typename T> – множина значень типу T, які можуть повторюватись. Протестувати шаблон на таких типах: • short int • double • CFixedPointNumber клас (розроблений на лабораторній роботі №3)
5	Шаблон класу CDictionary<typename TKey, typename TValue> – асоціативний масив із ключем типу TKey та значенням типу TValue. Протестувати шаблон на таких типах: • int, std::string • std::string, int • char *, CPoint (розроблений на лабораторній роботі №3)
6	Шаблон класу CString<typename T> – стрічка символів (масив елементів символьного типу T). Пам’ять під елементи масиву повинна виділятися динамічно. Протестувати шаблон на таких типах: • char • short int • int
7	Шаблон класу CSingleLinkedList<typename T> – однозв'язний список об’єктів типу T. Пам’ять під елементи списку повинна виділятися динамічно. Протестувати шаблон на таких типах: • int • double • CSphere клас (розроблений на лабораторній роботі №3)
8	Шаблон класу CDoubleLinkedList<typename T> – двозв'язний список об’єктів типу T. Пам’ять під елементи списку повинна виділятися динамічно. Протестувати шаблон на таких типах: • unsigned int

	• float • CTriangle клас (розроблений на лабораторній роботі №3)
9	Шаблон класу CStack<typename T> – стек об’єктів типу T. Пам’ять під елементи стеку повинна виділятися динамічно. Протестувати шаблон на таких типах: • unsigned char • long long • CRectangle клас (розроблений на лабораторній роботі №3)
10	Шаблон класу CQueue<typename T> – однобічна черга об’єктів типу T. Пам’ять під елементи черги повинна виділятися динамічно. Протестувати шаблон на таких типах: • unsigned short • unsigned long long • CPolynom2 клас (розроблений на лабораторній роботі №3)
11	Шаблон класу CDeque<typename T> – двобічна черга об’єктів типу T. Пам’ять під елементи черги повинна виділятися динамічно. Протестувати шаблон на таких типах: • bool • float • CPolynom3 клас (розроблений на лабораторній роботі №3)
12	Шаблон класу CBitArray<size_t BitsCount> – одновимірний масив бітів. Кількість бітів задається параметром шаблона. Для збереження бітів використовувати масив 8-бітних цілих чисел. Розмір масиву має визначатись на етапі компіляції. Протестувати шаблон на таких значеннях BitsCount: • 1 • 9 • 256
13	Шаблон класу CTable<typename T> – таблиця із колонками та рядками. Містить опис колонок та рядки із даними типу T. Пам’ять під елементи масиву повинна виділятися динамічно. Протестувати шаблон на таких типах: • int • bool • CBitField клас (розроблений на лабораторній роботі №3)
14	Шаблон класу CEncryptor<size_t ShiftCount> – клас для шифрування/дешифрування тексту простим зсувом символів по алфавіту на ShiftCount позицій. Пам’ять під текст повинна виділятись динамічно. Протестувати шаблон на таких значеннях ShiftCount: • 1 • 10 • 52
15	Шаблон класу CStaticString<size_t MaxSize, typename T> – стрічка символів (масив елементів символьного типу T) фіксованого максимального розміру MaxSize. Розмір масиву має визначатись на етапі компіляції. Протестувати шаблон на таких розмірах та типах: • 256, char • 30, int • 1, bool

Контрольні питання

1. Які сутності може ґенерувати шаблон у C++?
2. Які є види параметрів шаблону?
3. Що таке параметри, що позначають тип? Який синтаксис їх опису?
4. Що таке параметри, що позначають не тип? Який синтаксис їх опису?
5. Що таке шаблонні параметри? Який синтаксис їх опису?
6. Що таке інстанціація шаблону?