Лабораторна робота №6.

Тема: *Параметризовані типи. Шаблони функції. Шаблони класів. Ітератори.*

Методичні вказівки

1. Параметризовані типи.

Шаблони (англ. *template*) – засіб мовиС++, який призначений для кодування узагальнених алгоритмів, без прив'язки до деяких параметрів: типу даних, розміру буфера та стандартного значення. В С++ можливе створення шаблону функції і шаблону класу. Шаблони також називають родовими чи параметризованими типами, дозволяють створювати (конструювати) *сімейства родинних функцій і класів*. Мета введення шаблонів - автоматизація створення функцій та класів, що можуть обробляти різнотипні дані. На відміну від механізму перевантаження, коли для кожного набору формальних параметрів визначається своя функція та клас, шаблон сімейства функцій такласів визначається один раз, але це визначення параметризуется.

2. Шаблони функцій.

Узагальнена функція визначає універсальну сукупність операцій, застосовних до різних типів даних. Тип даних, з якими працює функція, передається як параметр. Узагальнена функція оголошується за допомогою ключового слова template. Визначення шаблонної функції виглядає в такий спосіб.

```
template <typename T>
тип_значення_що_повертається ім'я_функції(список_параметрів)
{
// Тіло функції
}
```

Тут параметр T задає тип даних, з яким працює функція. Цей параметр можна використовувати й усередині функції, однак при створенні конкретної версії узагальненої функції компілятор автоматично підставить замість нього фактичний тип. За правилами узагальнений тип задається за допомогою ключового слова `typename`, хоча замість нього можна застосовувати традиційне ключове слово `class`

3. Шаблони класів.

Крім узагальнених функцій можна визначити узагальнені класи. При цьому створюється клас, у якому визначені всі алгоритми, але фактичний тип даних задається як параметр при створенні об'єкта.

Узагальнені класи виявляються корисними, якщо логіка класу не залежить від типу даних. Наприклад, до черг, що складаються з цілих чисел або символів, можна застосовувати той самий алгоритм. Оголошення узагальненого класу має наступний вид.

```
template < typename T> class ім'я_класу
{
..
}
```

Тут параметр *T* задає тип даних, що уточнюється при створенні екземпляра класу. При необхідності можна визначити декілька узагальнених типів, використовуючи список імен, розділених комами. Конкретний екземпляр узагальненого класу створюється за допомогою наступної синтаксичної конструкції.

```
ім'я_класу <тип> ім'я_об'єкта;
```

Тут параметр тип задає тип даних, якими оперує клас. Функції — члени узагальненого класу автоматично стають узагальненими. Для їхнього оголошення не обов'язково використовувати ключове слово ***template***.

4. Ітератори.

Ітератор — це об'єкт, що перебирає всі елементи (переходить від одного елемента до іншого). Він може обійти всі елементи контейнера. Ітератор представляє визначену позицію в контейнері. За звичаєм для ітераторів визначаються наступні фундаментальні операції.

- Операція * повертає елемент, що стоїть в поточній позиції. Якщо цей елемент має члени, то за допомогою операції -> можна одержати доступ до них безпосередньо з ітератора.
- Операція ++ переміщає ітератор уперед на наступний елемент. Більшість ітераторів також дозволяють повернення до попереднього елементу за допомогою операції --.
- Операції == і != повертають результат перевірки, чи представляють два ітератори ту саму позицію.
- Операція = присвоює ітератор (позицію елемента, на яку він посилається).

Завдання до лабораторної роботи:

Розробити та реалізувати класи згідно варіантів. Передбачити введення початкових даних: з клавіатури, файлу та використовуючи датчик випадкових чисел. Написати тестові програми для декількох типів.

Завдання 1. Варіанти задач.**Шаблони функцій.**

Задача 1.1. Написати функцію-шаблон послідовного пошуку в масиві по ключу. Функція повертає індекс останнього, знайденого елемента в масиві, рівного ключу. Написати специфікацію функції-шаблон для типу **char*.**

Задача 1.2. Написати функцію-шаблон, що знаходить максимальне значення в масиві та їх кількість.

Задача 1.3. Написати функцію-шаблон, що переставляє елементи в масиві. Написати специфікацію функції-шаблон для типу **char*.**

Задача 1.4. Написати функцію-шаблон, що обчислює середнє значення в масиві. Написати специфікацію функції-шаблон для типу **char*.**

Задача 1.5. Написати функцію-шаблон, що обчислює мінімальне значення в масиві. Написати специфікацію функції-шаблон для типу **char*.**

Задача 1.6. Написати родову функцію у вигляді функції-шаблон. Функція міняє місцями два аргументи. Написати специфікацію функції-шаблон для типу **char*.**

Задача 1.7. Описати функції-шаблони для знаходження мінімального та максимального елемента пари чисел. Написати специфікацію функції-шаблон для типу **char*.**

Завдання 2. Варіанти задач. **Шаблони функцій 2.**

Задача 2.1. Написати функцію-шаблон функцію впорядкування методом швидкого впорядкування. Написати специфікацію функції-шаблон для типу **char*.**

Задача 2.2. Написати функцію-шаблон функцію впорядкування методом Шелла. Написати специфікацію функції-шаблон для типу **char*.**

Задача 2.3. Написати функцію-шаблон функцію впорядкування методом «Вставки». Написати специфікацію функції-шаблон для типу **char*.**

Задача 2.4. Написати функцію-шаблон функцію впорядкування методом «Вилучення». Написати специфікацію функції-шаблон для типу **char*.**

Задача 2.5. Написати функцію-шаблон бінарного пошуку. Якщо дані, по яких потрібно провести пошук, відсортовані, то можна використовувати бінарний пошук. Тому, спочатку треба написати функцію впорядкування. Написати специфікацію функції-шаблон для типу **char*.**

Завдання 3. Варіанти задач. **Шаблони класів.**

Задача 3.1. Створити параметризований масив з конструкторами, деструктором і перевантаженими операторами [], =, +, +=,-,-=.
Задача 3.2. Створити параметризованний стек.
Задача 3.3. Створити параметризованную черга.
Задача 3.4. Створити параметризованную циклічну чергу.
Задача 3.5. Створити параметризованний клас - матриця. Визначені конструктори, деструктор і перевантажені оператори =, [], +, +=.
Задача 3.6. Створити параметризованний клас бінарного дерева. З методами - додати елемент у дерево, проходження по дереву в спадному й у висхідному порядку. Здійснити пошук по дереву.
Задача 3.7. Створити параметризованний клас однозв'язного списку.

Завдання 4. Варіанти задач. ***Ітератори.***

Задача 4.1. Побудувати клас, що описує бінарне дерево стандартної форми. Побудувати клас ітератор, що дозволяє обходити дерево. Написати програму, яка проходить по бінарному дереву використовуючи ітератор.
Задача 4.2. Побудувати клас, що описує лінійний однозв'язний список. Побудувати клас ітератор, що дозволяє проходити список. Написати програму, яка тестує створений ітератор.
Задача 4.3. Побудувати клас, що описує лінійний двох зв'язний список. Побудувати клас ітератор, що дозволяє проходити список.
Задача 4.4. Написати програму, яка тестує створений ітератор. Побудувати клас, що описує масив. Побудувати клас ітератор, що дозволяє проходити масив. Написати програму, яка тестує створений ітератор.
Задача 4.5. Описати шаблон класу **List,** який визначає однозв'язний список для елементів будь-якого типу. Побудувати клас ітератор, що дозволяє проходити список. Написати програму тестування цього шаблону цілих і дійсних чисел.