Ячменьова С. О., здобувач,
Марчук Г. В., ст. викладач
Державний університет «Житомирська політехніка»

ВИКОРИСТАННЯ РОЗУМНИХ ПОКАЖЧИКІВ В С++

На сьогоднішній день одним із ефективних способів роботи з великою кількістю інформації в С++ є використання розумних покажчиків (Smart pointer), що розширюють функціональні можливості звичайного покажчика задля усунення проблем при роботі з пам'яттю, зокрема випадків спроби подвійного звільнення її комірок, тощо. Задля реалізації розумних покажчиків є необхідним створення абстрактного класу, який буде імітувати роботу звичайного покажчика, тому важливо зазначити.

Покажчиком є будь-яка змінна, яка містить адресу іншої в пам'яті і визначається як похідний тип даних. За видами використання розрізняють три види покажчиків: керовані покажчики (managed pointers), некеровані покажчики (unmanaged pointers), некеровані покажчики на функції (unmanaged function pointers) [1].

Традиційні або звичайні покажчики С++ є некерованими покажчиками, які вказують на об'єкти в нерегульованому об'ємі пам'яті, що виділяється для виконання програми [1]. Основними підвидами звичайних покажчиків є: покажчики на ціле і дійсне значення, на масив, символи, функції і структури, подвійні, нульові, постійні покажчики а також покажчики порожнечі і на константу [2].

У випадку, коли використання звичайного покажчика є недостатнім для вирішення програмних проблем застосовують розумні покажчики - класи, імітуючи роботу вказівника і розширюючи його можливості. Розрізняють три основних види розумних покажчиків: унікальний (unique), частковий (shared), слабкий (weak) [3].

Unique_ptr – це розумний покажчик, який володіє та керує іншим об'єктом через вказівник і позбавляється цього об'єкта, коли unique_ptr виходить за межі видимості [4]. Покажчик оголошується в С++ в просторі імен std за звернення std::unique_ptr.

```
unique_ptr<int> p1(new int(25));
    int* p2=p1.get();
    cout<<"*p1 = " << *p1 << endl << "*p2 = " << *p2 << endl;
```

Один з розумних покажчиків, який може вказувати на об'єкт, який вже має покажчика називається shared_ptr, оголошується в С++ в просторі імен std за звернення std::shared_ptr. Коли видаляється або перезаписується останній shared_ptr, то об'єкт знищується, а його пам'ять звільняється [5].

```
shared_ptr<float> p3(new float(2.8f));
shared_ptr<float> p5 = p3;
cccout << "*p3 = " << *p3 << endl << "*p5= " << *p5 << endl;
```

Розумний покажчик, який містить не володіючи ("слабке") посилання на об'єкт називають weak_ptr, викликають у std в С++ за допомогою звернення std::shared_ptr. Цей вказівник потрібно перетворити на shared_ptr, щоб отримати доступ до об'єкта [6].

```
weak_ptr<int> p6;
shared_ptr<int> p7(new int(5));
p6 = p7;
cout << "p6 " << *p6.lock().get() << endl;
```

Отже, використання розумних покажчиків в С++ - це поширена практика у сучасному світі, яка полегшує процес роботи з пам'яттю. Подальші дослідження будуть спрямовані на вдосконалення існуючих методів використання розумних вказівників.

Список використаних джерел

1. Покажчики. Частина 1. Загальні поняття. Типи покажчиків. Керовані та некеровані покажчики. Покажчики на функцію. Приклади використання [Електронний ресурс] // Bestprogish, BestBlog. – 2017. – Режим доступу до ресурсу:
https://www.bestprog.net/uk/2017/03/21/покажчики-частина-1-загальні-поняття/..
2. C Pointers. [Електронний ресурс] – Режим доступу до ресурсу: https://www.geeksforgeeks.org/c-pointers/.
3. Розумні покажчики. Класи покажчиків unique_ptr, shared_ptr, weak_pt. [Електронний ресурс] // Bestprogish, BestBlog. – 2017. – Режим доступу до ресурсу: https://www.bestprog.net/uk/2022/04/10/c-smart-pointers-pointer-classes-unique_ptr-shared_ptr-weak_ptr-ua/.
4. std:: unique_ptr. [Електронний ресурс] // cppreference.com. – 2023. – Режим доступу до ресурсу:
https://en.cppreference.com/w/cpp/memory/unique_ptr.
5. std:: shared_ptr. [Електронний ресурс] // cppreference.com. – 2023. – Режим доступу до ресурсу:
https://en.cppreference.com/w/cpp/memory/shared_ptr.
6. std:: weak_ptr. [Електронний ресурс] // cppreference.com. – 2023. – Режим доступу до ресурсу:
https://en.cppreference.com/w/cpp/memory/weak_ptr.