**Лабораторна робота №10.** **Розробка програм із використанням простого спадкування**

**Мета:** Набуття навичок в розробці програм, де використовуються дружні функції

**Порядок виконання роботи**

1. Повторити лекцію №15.
2. Ознайомитися з теоретичними відомостями.
3. Продовжити працювати над програмою минулої лабораторної роботи.
   1. Створити через просте спадкування похідний до *Person к*лас *Student*, в якому створити:
4. члени-дані : курс, спеціальність, поточна кількість балів по предметах фізика, математика, програмування;
5. три конструктори (порожній, ініціалізації, копіювання) та деструктор, в яких поставити виведення повідомлення для моніторингу їх роботи;
6. селектори та модифікатори до всіх членів похідного класу,
7. метод введення даних в інтерактивному режимі (використати успадкований від базового класу, якщо такого в базовому класі не було – записати),
8. метод виведення даних на консоль (використати успадкований від базового класу, якщо такого в базовому класі не було – записати);
9. дружні функції 1) виведення даних, попередньо введених в інтерактивному режимі, до файлу в двох режимах: формування нового файлу та дозапису до існуючого; 2) введення даних з файлу та заповнення масиву об’єктів..
   1. В головній програмі (функції main()):
10. включити виклик нових методів та функцій до відповідних позицій меню;
11. створити позицію меню "The state of the objects of the *Student* class" (Стан об'єктів класу *Student*), де виводити інформацію про кількість об’єктів та стан їхньої обробки;
12. створити екземпляр класу *Student* та заповнитийого даними та вивести на консоль.
13. Виведення до файлу та введення даних з файлу щодо студентів виконувати за відпрацьованими алгоритмами на ЛР№№3-4, а саме: інформація записується до файлу, де дані атрибутів відділяємо знаком ";", а кінець позначаємо знаком "#", після якого записуємо поточну дату у вигляді РРММДД (де РР – дві останні цифри року, ММ – номер місяця, ДД – номер дня). Відповідно, введені дані потім потрібно буде аналізувати та "розбирати" на складові, щоб присвоїти значення відповідним членам класу. На цьому етапі ім’я файлу надаєте літералом у форматі FS\_**<Прізвищеанглійською>.tхt.**
14. Для виведення даних класу *Student* створити позицію меню "Formation of file objects *Student* " (Формування файлу об’єктів *Student*), а для введення – "Read the *Student* object file" (Читання файлу об’єктів *Student*).
15. Встановити аналіз виклику нових функцій, зокрема, виконувати виведення в разі наявності заповнених об’єктів класу *Student*.
16. В першому рядку програми та заголовкового файлу повинні бути записаними в коментарі номер групи та прізвище, а також номер ЛР (через кому до попередньої).
17. Результати надсилати на електронну адресу викладача

[**t.i.lumpova@gmail.com**](mailto:t.i.lumpova@gmail.com)у вигляді заголовкового файлу та cpp-файлу з іменем у форматі

Для заголовкового файлу

**<Прізвище англійською>.h**

Наприклад, Ivanov.h

Для cpp-файлу:

**<Номер групи> <Номер лабораторної><Прізвище англійською>.cpp**

Наприклад, 31-01Ivanov.cpp.

Тему в заголовку листа записати

**ООП<Номер групи>-ЛР <Номер лабораторної>-<Прізвищеанглійською>**

**Строк відсилки ЛР ІПЗ-31 10.11.2020**

**ІПЗ-32 10.11.2020**

Всі запитання, що виникнуть, надсилайте на електронну адресу викладача, тему в заголовку листа записати

**ООП<Номер групи>-Запитання-<Прізвище англійською>**.

**Теоретичні відомості.**

**Підрахунок створених об'єктів**

Припустимо, нам необхідно здійснювати підрахунок об'єктів певного класу. Можна створити лічильник – статичний елемент даних, в якому зберігатиметься кількість об'єктів, присутніх у пам'яті. Збільшення значення лічильника здійснюємо в конструкторі, а зменшення – у деструкторі. Програма може мати такий вигляд:

#include <iostream>

#include <Windows.h>

**using** std::cout;

**using** std::endl;

**class** ObjectCount

{

**private**:

**static** **int** count;

**public**:

**static** **int** getCount()

{

**return** count;

}

ObjectCount()

{

count++;

}

~ObjectCount()

{

count--;

}

};

// Статичний елемент даних слід визначити й ініціалізувати поза межами класу:

**int** ObjectCount::count = 0;

**void** main()

{ system("color F0");

ObjectCount c1;

cout << c1.getCount() << endl; // 1

ObjectCount \*p1 = &c1; // копіюємо адресу, конструктор не викликається

cout << p1->getCount() << endl; // 1

ObjectCount \*p2 = **new** ObjectCount();

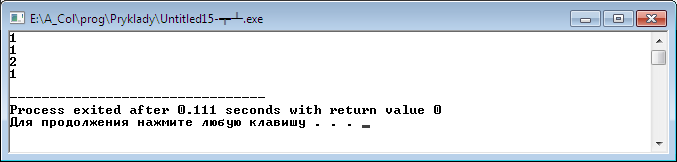
cout << p2->getCount() << endl; // 2

**delete** p2;

cout << p2->getCount() << endl; // 1

}

Результат роботи



Як видно з результатів виконання функції main(), функцію getCount() можна викликати для об'єкта навіть після того, як він був видалений з динамічної пам'яті. Це пов'язано з тим, що під час виклику статичної функції для компілятора важливим є лише тип вказівника, а не об'єкт, на який він указує. Більш коректним є виклик функції через ім'я класу:

**void** main()

{

ObjectCount c1;

cout << ObjectCount::getCount() << endl; // 1

ObjectCount \*p1 = &c1;

cout << ObjectCount::getCount() << endl; // 1

ObjectCount \*p2 = **new** ObjectCount();

cout << ObjectCount::getCount() << endl; // 2

**delete** p2;

cout << ObjectCount::getCount() << endl; // 1

}