**Лабораторна робота №9.** **Використання дружніх функцій.**

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

**Порядок виконання роботи**

1. Повторити дружні функції та класи.
2. Ознайомитися з теоретичними відомостями.
3. Продовжити працювати над програмою минулої лабораторної роботи. Створити для класу *Person* дружні функції а) виведення даних, попередньо введених в інтерактивному режимі, до файлу в двох режимах: формування нового файлу та до запису до існуючого; б) введення даних з файлу та заповнення масиву об’єктів.
4. Виведення до файлу та введення даних з файлу 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 FР\_**<Прізвищеанглійською>.tхt.**
5. Для виведення даних класу *Person* створити позицію меню "Formation of file objects Perso*n* " (Формування файлу об’єктів *Person*), а для введення – "Read the Person object file" (Читання файлу об’єктів Person).
6. Встановити аналіз виклику нових функцій, зокрема, виконувати виведення в разі наявності заповнених об’єктів класу *Person*.
7. В першому рядку програми та заголовкового файлу повинні бути записаними в коментарі номер групи та прізвище, а також номер ЛР (через кому до попередньої).
8. 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 06.11.2020**

**ІПЗ-32 05.11.2020**

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

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

**Теоретичні відомості.**

**Дружні функції.**Клас може надавати особливі привілеї певним зовнішнім функціям або функціям-членам іншого класу. Ці функції одержали назви дружніх. Якщо функція або клас оголошена як дружні даному класу, то такі функції або функції-члени такого класу можуть здійснювати безпосередній доступ до всіх полів класу, для якого вони дружні. Дружні функції і класи можуть здійснювати прямий доступ до закритих полів класу без використання функцій-членів цього класу.

Ключове слово **friend**– специфікатор функції, який дає функції – не члену класу доступ до прихованих членів класу. Він використовується для того, щоб вийти за строгі рамки типізації і заховання даних в С++.

Одна з причин їх використання полягає втом, що деякі функції потребують привілейованого доступу більш, чим до одного класу. Друга причина – **friend-функція**передає всі параметри через список параметрів, і значення кожного з них підлегло перетворенню, сумісному з призначенням. Такі перетворення застосовуються до явно переданих аргументів-класів і тому особливо корисні у випадках перевантаження оператора.

Оголошення **friend**функції повинне з'являтися усередині оголошення класу, якому вона дружня. Імені функції передує ключове слово friend, і її оголошення може знаходиться як в public так і в private частини класу, що не вплине на значення. Функція-член одного класу може бути friend-функцією іншого класу. Це відбувається тоді, коли функція-член оголошена в friend класі з використанням оператора дозволу контексту для визначення імені функції дружнього класу. Якщо всі функції-члени одного класу є friend-функціями іншого класу, то це можна визначити записом:

friend class <ім’я класу>

class t1

{

friend void а(); // friend-функція

int b(); // функція-член

};

class t2

{

friend int t1::b();

// функція-член класу t1 має доступ до всіх прихованих полів класу t2

};

class t3

{

friend class t1;

// всі функції-члени класу t1 мають доступ до всіх полів класу t3

};

Розглянемо клас **matrix**і клас vector. Функція множення вектора на матрицю повинна мати доступ до **private-членів**обох класів. Ця функція буде **friend**для обох класів.

class matix;

class vect

{

int \*p;

int size;

friend vect mpy(const vect &,const matix &);

public:

};

class matrix

{

int \*\*base;

int row,column;

friend vect mpy(constr vect&,const matirx &);

};

vect mpy(const vect &v,const matrix &m)

{

if(v.size!=m.row) { exit(1); }

vect ans(column);

//.....

return ans;

}

Другорядне значення вимагає попереднього опису класу **matrix**. Воно необхідне тому, що функція mpy повинна з'являтися в обох класах, і використовує кожен клас як тип аргументу.

**Friend-функції**можна розглядати як частину загального інтерфейсу класу. Існує ряд ситуацій, в яких вони можуть бути альтернативою функціям-членам. Використання freind-функцій спірне, тому що вони порушують інкапсуляцію, що оточує private члени класів. Парадигма ООП стверджує, що об'єкти (у С++ вони - змінні класу) доступні їх public члени. Тільки функції-члени повинні мати доступ до прихованої реалізації абстрактного типу даних. Це ясний і строгий принцип проектування. Friend-функція знаходиться на сама його межі, оскільки має доступ до private членам, сама не будучи функцією-членом. З її допомогою можна організувати швидкий код для доступу до подробиць реалізації класу.

**Контрольні запитання**

**Контрольні питання**

1. Які особливості (щодо свого опису і застосування) мають конструктори?
2. Які особливості (щодо свого опису і застосування) мають деструктори?
3. Які властивості мають поля та методи, описані у private?
4. Які властивості мають поля та методи, описані у public?
5. Які властивості мають поля та методи, описані у protected?
6. Яка функція називається функцією-аксесором?
7. Що таке конструктор копіювання?
8. Які властивості мають поля, описані як static?
9. Які властивості мають поля, описані як const?
10. Яке призначення ключового слова this?
11. Як можна описати масив об’єктів?
12. Які особливості застосування деструктора до масиву об’єктів?
13. Які бувають різновиди класів?
14. Який оператор застосовується для розширення області видимості?
15. Які ви знаєте потоки введення/виведення? Як можна їх застосовувати?