ЛАБОРАТОРНА РОБОТА №2
ДРУЖНІ ФУНКЦІЇ С ++

Мета роботи – Отримати практичні навички дружніх функцій для класів на С ++.

Основні теоретичні відомості

У С++ існує можливість дозволити доступ до закритих членів класу функціям, які не є членами цього класу. Для цього достатньо оголосити ці функції «дружніми» (або «друзями») стосовно розглянутого класу. Щоб зробити функцію «другом» класу, включіть її прототип в public-розділ оголошення класу і передуватимете його ключовим словом **friend**. Наприклад, в цьому фрагменті коду функція frnd () оголошується «другом» класу cl.

```
class cl {
// . . .
public:
friend void frnd(cl ob);
// . . .
};
```

Ключове слово friend надає функції, яка не є членом класу, доступ до його закритим членам.

Функція може бути «другом» декількох класів.

Розглянемо короткий приклад, в якому функція-«друг» використовується для доступу до закритих членів класу myclass.

```
// Демонстрація використання функції-"друга".
class myclass {
int a, b;
public:
myclass(int i, int j) { a=i; b=j; }
friend int sum(myclass x); // Функція sum () - "друг" класу
myclass.
};
// Зверніть увагу на те, що функція sum () не є членом  класу
int sum(myclass x)
{
/* Оскільки функція sum () - "друг" класу myclass,
вона має право на прямий доступ до його членів-даних a й b. */
return x.a + x.b;
}
```

```
int main ()
{
myclass n (3, 4);
cout << sum(n);
}
```

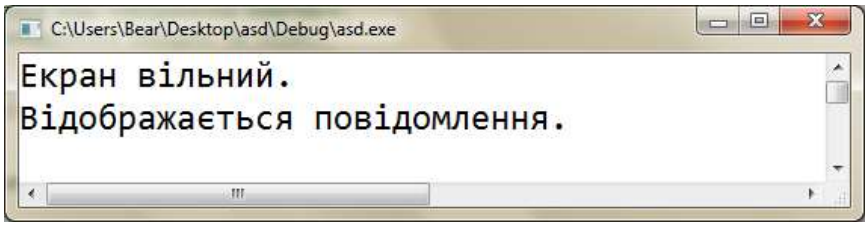
Незважаючи на те що в даному прикладі ми не отримаємо ніякої користі з оголошення функції sum() «другом», а не членом класу myclass, існують певні обставини, при яких статус функції – «друга» має велике значення. По-перше, функції – «друзі» можуть бути корисні для перевантаження операторів певних типів. По-друге, функції – «друзі» спрощують створення деяких функцій вводу-виводу. Про це мова попереду.

Третя причина використання функцій – «друзів» полягає в тому, що в деяких випадках два (або більше) класи можуть містити члени, які знаходяться у взаємному зв'язку з іншими частинами програми. Наприклад, у нас є два різних класи, які при виникненні певних подій відображають на екрані «спливаючі» повідомлення. Інші частини програми, які призначені для виведення даних на екран, повинні знати, чи є «спливаюче» повідомлення активним, щоб випадково не перезаписати його. У кожному класі можна створити функцію-член, що повертає значення, за яким можна судити про те, чи активно повідомлення чи ні; однак перевірка цієї умови потребує додаткових витрат (тобто двох викликів функцій замість одного). Якщо статус «спливаючого» повідомлення необхідно перевіряти часто, ці додаткові витрати можуть виявитися просто неприйнятними. Однак за допомогою функції, «дружній» для обох класів, можна безпосередньо перевіряти статус кожного об'єкта, викликаючи тільки одну функцію, яка буде мати доступ до обох класів. У подібних ситуаціях функція-«друг» дозволяє написати більш ефективний код. Ця ідея ілюструється на прикладі наступної програми.

```
// Використання функції-"друга".
const int IDLE=0;
const int INUSE=1;

class C2; // випереджальне оголошення
```

```
class C1 {
int status; // IDLE якщо повідомлення неактивно,
// INUSE якщо повідомлення виведено на екран.
public:
void set_status(int state);
friend int idle(C1 a, C2 b);
};

class C2 {
int status; // IDLE якщо повідомлення неактивно,
// INUSE якщо повідомлення виведено на екран.
public:
void set_status(int state);
friend int idle(C1 a, C2 b);
};

void C1::set_status(int state)
{
status = state;
}

void C2::set_status(int state)
{
status = state;
}
// Функція idle () - "друг" для класів C1 і C2.
int idle(C1 a, C2 b)
{
if(a.status || b.status) return 0;
else return 1;
}

int main()
{
setlocale(0,"Ukr");
C1 x;
C2 y;
x.set_status(IDLE);
y.set_status(IDLE);
if(idle(x, y)) cout << "Екран вільний."<<endl;
else cout << "Відображається повідомлення."<<endl;
x.set_status(INUSE);
if(idle(x, y)) cout << "Екран вільний."<<endl;
else cout << "Відображається повідомлення."<<endl
}
```

При виконанні програма генерує такі результати:



Оскільки функція idle () є «другом» як для класу C1, так і для класу C2, вона має доступ до закритого члену status, визначеному в обох класах. Таким чином, стан об'єкта кожного класу одночасно можна перевірити одним зверненням до функції idle().

Випереджальне оголошення призначене для оголошення імені класового типу до визначення самого класу.

Зверніть увагу на те, що в цій програмі використовується випереджальне оголошення для класу C2. Його необхідність обумовлена тим, що оголошення функції idle() в класі C1 використовує посилання на клас C2 до його оголошення. Щоб створити випереджальне оголошення для класу, досить використовувати формат, представлений в цій програмі.

«Друг» одного класу може бути членом іншого класу. Перепишемо попередню програму так, щоб функція idle() стала членом класу C1. Зверніть увагу на використання оператора дозволу області видимості (або оператора дозволу контексту) при оголошенні функції idle() в якості "друга" класу C2.

```
//Функція може бути членом одного класу і одночасно
//"другом" іншого
const int IDLE=0;
const int INUSE=1;
class C2; // випереджальне оголошення
class C1 {
int status; // IDLE якщо повідомлення неактивно,
// INUSE якщо повідомлення виведено на екран.
public:
void set_status(int state);
int idle(C2 b); //тепер це член класу C1
};

class C2 {
int status; // IDLE якщо повідомлення неактивно,
// INUSE якщо повідомлення виведено на екран.
public:
void set_status(int state);
```

```
friend int C1::idle(C2 b); // функція-"друг"
};

void C1::set_status(int state)
{
status = state;
}

void C2::set_status(int state)
{
status = state;
}
// Функція idle () - член класу C1 і "друг" класу C2.
int C1::idle(C2 b)
{
if(status || b.status) return 0;
else return 1;
}


int main()
{
setlocale(0,"Ukr");
C1 x;
C2 y;
x.set_status(IDLE);
y.set_status(IDLE);
if(x.idle(y)) cout << "Екран вільний."<<endl;
else cout << "Відображається повідомлення."<<endl;
x.set_status(INUSE);
if(x.idle(y)) cout << "Екран вільний."<<endl;
else cout << "Відображається повідомлення."<<endl;
}
```

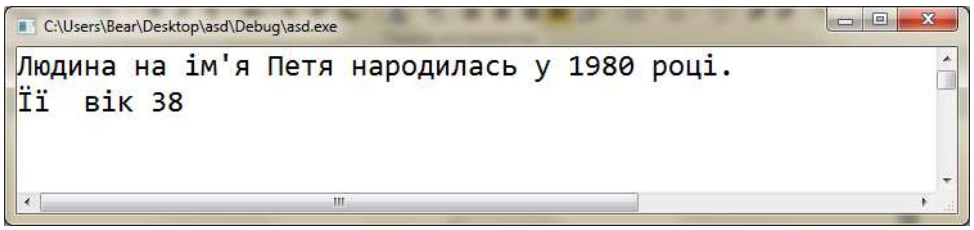
Оскільки функція idle() є членом класу C1, вона має прямий доступ до змінної status об'єктів типу C1. Отже, в якості параметра необхідно передавати функції idle () тільки об'єкти типу C2.

Взагалі, цілий клас A може бути «другом» класа B , тоді всі функії-члени класу A є «друзями» класа B. Розглянемо приклад, в якому один клас є «дружнім» до іншого.

```
class MyMethods; // зазделегідь оголошуємо клас, якиий
                 // стане дружнім

class Human
{
protected:
```

```
 string name;
 int year;
public:

 Human(string Name, int Year)
 {
      name = Name;
      year=Year;

 }
friend     MyMethods;  // дружній клас

};

//Всі функції класу MyMethods мають доступ до всіх членів
//класу Human
class MyMethods
{
public:
void GetInfo(Human human)
 {
      cout<<"Людина на ім'я "<<human.name<<" народилась у
"<<human.year<<" році."<<endl;
 }
void GetAge(Human human)
 {
      cout<<"Її  вік "<<2018-human.year<<endl;
 }
};
int main()
{
 Human h("Петя", 1980);
 MyMethods mm;
 mm.GetInfo(h); // виклик функції GetInfo
 mm.GetAge(h);  // виклик функції GetAge
}
```

Результат роботи програми:



Завдання до лабораторної роботи

1. Визначити користувальницький клас у відповідності з варіантом завдання.
2. Створити дружню функцію у відповідності з варіантом завдання.
3. Створити звіт.

Варіанти завдання

Варіант 1.

Створити клас «комплексне число», такий, що:

a) його екземпляр містить дійсну і уявну частину - змінні з плаваючою точкою.

b) його конструктор без параметра створює екземпляр зі значенням 0.0 + i0.0, а конструктор з параметрами створює екземпляр з відповідною дійсною та уявною частиною.

a) його методи дозволяють отримувати і присвоювати значення дійсної та уявної частини.

c) .

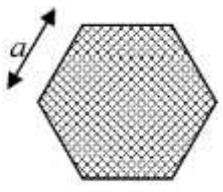
d) функція print () виводить на екран значення екземпляра у вигляді (0.00 ± i0.00).

e) дружня функція frd в якості параметрів отримує екземпляри двох «комплексних чисел», і повертає нове «комплексне число», дійсна і уявна частини якого дорівнюють сумам дійсних і уявних частин цих двох «комплексних чисел».

Варіант 2.

Створити клас «шайба» (правильний шестикутник), такий, що:

$$S = \frac{3\sqrt{3}}{2}a^2$$



а) його екземпляр містить розмір сторони a.

b) його конструктор без параметра створює екземпляр зі значенням 0, а конструктор з параметрами створює екземпляр з відповідним значенням a.

c) його методи дозволяють отримувати і присвоювати значення a і обчислювати значення площі.

d) функція print_sqrt () виводить на екран значення сторони і площі цієї фігури.

e) дружня функція frd в якості параметрів отримує екземпляри двох шайб і повертає нову шайбу, площа якої дорівнює сумі площ цих двох шайб. Значення сторони нової шайби має бути перераховане.

Варіант 3.

Створити клас «квадрат», такий, що:

a) його екземпляр містить розмір сторони квадрата.

b) його конструктор без параметра створює екземпляр зі значенням 0, а конструктор з параметрами створює екземпляр з відповідним значенням сторони.

c) його методи дозволяють отримувати і присвоювати значення сторони і обчислювати значення площі.

d) функція print_sqr () виводить на екран значення його сторони і площі.

e) дружня функція frd в якості параметрів отримує екземпляри двох квадратів і повертає новий квадрат, площа якого дорівнює сумі площ цих двох квадратів. Значення сторони нового квадрата має бути перераховане.

Варіант 4.

Створити клас «трикутник», такий, що:

a) його екземпляр містить розміри трьох сторін трикутника.

b) його конструктор без параметра створює екземпляр зі значенням 0, а конструктор з параметрами створює екземпляр з відповідним значенням сторін.

c) його методи дозволяють отримувати і присвоювати значення сторін і обчислювати значення площі (наприклад, за формулою Герона), $s = \sqrt{p(p-a)(p-b)(p-c)}, p = (a+b+c)/2$.

d) функція print_sqr () виводить на екран значення його сторін і площі.

e) дружня функція frd в якості параметрів отримує екземпляри двох трикутників і повертає новий трикутник, сторони якого дорівнюють сумам сторін цих двох трикутників.

Варіант 5

Створити клас «циліндр», такий, що:

a) його екземпляр містить розмір радіуса і висоти.

b) його конструктор без параметра створює екземпляр зі значенням 0, а конструктор з параметрами створює екземпляр з відповідним значенням радіуса і висоти.

c) його методи дозволяють отримувати і присвоювати значення радіуса і висоти та обчислювати значення об'єму.

d) функція print_val () виводить на екран значення його радіуса, висоти об’єму.

e) дружня функція frd в якості параметрів отримує екземпляри двох циліндрів і повертає новий циліндр, об’єм якого дорівнює сумі об’ємів цих двох циліндрів, а радіус дорівнює максимальному серед цих двох циліндрів.

Варіант 6

Створити клас «куля», такий, що:

a) його екземпляр містить розмір радіусу.

b) його конструктор без параметра створює екземпляр зі значенням 0, а конструктор з параметрами створює екземпляр з відповідним значенням радіусу.

c) його методи дозволяють отримувати і присвоювати значення радіусу і обчислювати значення об'єму.

d) функція print_val () виводить на екран значення його радіусу і об'єму.

e) дружня функція frd в якості параметрів отримує екземпляри двох куль і повертає нову кулю, об’єм якої дорівнює сумі об’ємів цих двох куль. Значення радіусу нової кулі має бути перераховане.

Варіант 7.

Створити клас «конус», такий, що:

a) його екземпляр містить розмір радіуса і висоти.

b) його конструктор без параметра створює екземпляр зі значенням 0, а конструктор з параметрами створює екземпляр з відповідним значенням радіуса і висоти.

c) його методи дозволяють отримувати і присвоювати значення радіуса, висоти та обчислювати значення об'єму.

d) функція print_val () виводить на екран значення його радіуса, висоти і об’єму.

e) дружня функція frd в якості параметрів отримує екземпляри двох конусів і повертає новий конус, висота якого дорівнює сумі висот цих двох конусів, а радіус – найбільший з двох.

Варіант 8.

Створити клас «трапеція», такий, що:

a) його екземпляр містить розмір двох основ і висоти.

b) його конструктор без параметра створює екземпляр зі значенням 0, а конструктор з параметрами створює екземпляр з відповідним значенням підстав і висоти.

c) його методи дозволяють отримувати і присвоювати значення основ і висоти та обчислювати значення площі.

d) функція print_sqrt () виводить на екран значення її основ, висоти та площі.

e) дружня функція frd в якості параметрів отримує екземпляри двох трапецій і повертає нову трапецію, висота якої дорівнює сумі висот цих двох трапецій, а основи – сумі основ цих двох трапецій.

Варіант 9.

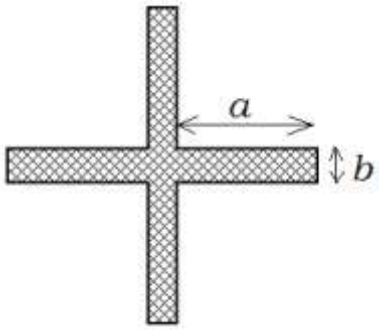
Створити клас «хрест», такий, що:

a) його екземпляр містить два розміри a і b.

b) його конструктор без параметра створює екземпляр зі значенням 0, а конструктор з параметрами створює екземпляр з відповідним значенням a і b.

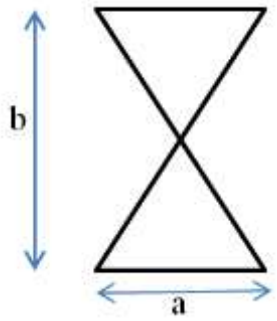
c) його методи дозволяють отримувати і присвоювати значення a, b та обчислювати значення площі.

d) функція print_sqrt () виводить на екран значення a, b та площі цієї фігури.



e) дружня функція frd в якості параметрів отримує екземпляри двох хрестів і повертає новий хрест, два розміри сторін a і b якого дорівнюють сумам сторін цих двох хрестів.

Варіант 10. Створити клас «пісочний годинник»:



a) його екземпляр містить розміри a і b.

b) його конструктор без параметра створює екземпляр зі значенням 0, а конструктор з параметрами створює екземпляр з відповідним значенням a і b.

c) його методи дозволяють отримувати і присвоювати значення a і обчислювати значення площі.

d) функція print_sqrt () виводить на екран значення a, b та площі цієї фігури.

e) дружня функція frd в якості параметрів отримує екземпляри двох «годинників» і повертає новий «годинник», два розміри сторін a і b якого дорівнюють сумам сторін цих двох «годинників».

Контрольні запитання

1. Чи може функція бути «дружньою» до багатьох класів?
2. Чи може бути один клас «другом» іншого?
3. Чи може «друг» одного класу може бути членом іншого класу?
4. Назвіть обставини, при яких статус функції – «друга» має велике значення.
5. Для чого потрібно попередне оголошення класу?