Лекція 5. Дружні функції

У C++ 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, треба включити її прототип в public-розділ оголошення класу і передувати його ключовим словом **friend**. Наприклад, в цьому фрагменті коду функція frnd() оголошується «другом» класу Cat.

```
class Cat {
// . . .
public:
friend void frnd(Cat obj);
// . . .
};
```

Ключове слово friend надає функції, яка не є членом класу, доступ до його закритих членів.

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 жодного 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);  // виводить число 7
}
```

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.

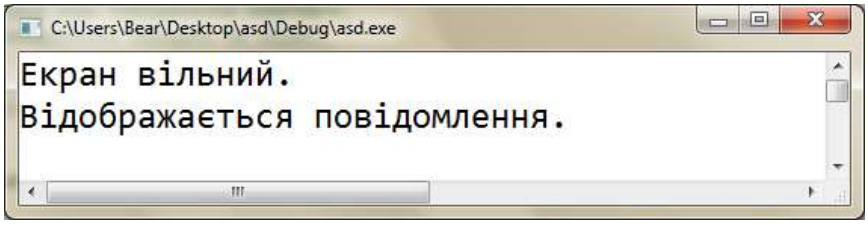
Третя причина використання «функцій-друзів» полягає в тому, що в деяких випадках два (або більше) класи можуть містити члени, які знаходяться у взаємному зв'язку з іншими частинами програми. Наприклад, у нас є два різних класи, які при виникненні певних подій відображають на екрані «спливаючі» повідомлення. 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); // idle – бездіяльність (англ.)
};

void C1::set_status(int state)
{
status = state;
}
// клас C2 визначається після оголошення дружньої функції у класі C1
// тому необхідно попередньо оголосити клас C2
 void C2::set_status(int state)
{
status = state;
}
// Функція idle () - "друг" для класів C1 і C2.
int idle(C1 a, C2 b) // idle – бездіяльність (англ.)
```

```
{
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:



У випадку, коли функція оголошується «другом» для кількох класів. можна зіткнутися з однією проблемою. Наприклад, «дружня» функція оголошується як у класі A так й у класі B і має в якості параметрів об’єкти класів A і B, а сам клас B визначається після оголошення «дружньої» функції у класі A. При компіляції з'являється повідомлення про помилку в рядку, де ця «дружня» функція оголошується у класі A. Для того, щоб уникнути такої помилки, необхідно попередньо оголосити клас B.

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().

Зверніть увагу на те, що в цій програмі використовується випереджальне оголошення для класу C2. Щоб створити випереджальне оголошення для класу, досить використовувати формат, представлений в цій програмі:

```
class C2; // випереджальне оголошення
```

«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 MyFunc; // зазделегідь оголошуємо клас, якиий стане дружнім
class Cat
{
 string name;
 int tail;
public:
 Cat(string Name, int Tail)
 {
       name = Name;
       tail = Tail;
 }
 friend MyFunc;  // дружній клас
};
//Всі функції класу MyFunc мають доступ до всіх членів класу Cat
class MyFunc
{
public:
 void AboutName(Cat cat)
 {
       cout << "Кота кличуть " << cat.name << endl;
 }
 void AboutTail(Cat cat)
 {
       cout << "Кіт " << cat.name << " має хвіст ";
       cout << cat.tail << " см." << endl;
```

```
 }
};
int main()
{
 setlocale(0, "ukr");
 Cat c("Томас", 25);
 MyFunc mf;
 mf.AboutName(c); // виклик функції AboutName
 mf.AboutTail(c); // виклик функції AboutTail
}
```

Результат роботи програми: