

Специфікатори доступу

Розглянемо наступну програму:

```
1 class DateClass // члени класу є закритими за замовчуванням
2 {
3     int m_day; // закрито по умовчання, доступ мають тільки інші члени класу
4     int m_month; // закрито по умовчання, доступ мають тільки інші члени класу
5     int m_year; // закрито по умовчання, доступ мають тільки інші члени класу
6 };
7
8 int main()
9 {
10     DateClass date;
11     date.m_day = 12; // помилка
12     date.m_month = 11; // помилка
13     date.m_year = 2018; // помилка
14
15     return 0;
16 }
```

Ви не зможете скомпілювати цю програму, так як всі члени класу є закритими за замовчуванням. Закриті члени (або ще «члени private») - це члени класу, доступ до яких мають лише інші члени цього ж класу. Оскільки функція main () не є членом DateClass, то вона і не має доступу до закритих членів об'єкта date.

Хоча члени класу є закритими за замовчуванням, ми можемо зробити їх відкритими, використовуючи ключове слово public :

```
1 class DateClass
2 {
3     public: // зверніть увагу тут на ключове слово public і двокрапку
4         int m_day; // відкрито, доступ має будь-який об'єкт
5         int m_month; // відкрито, доступ має будь-який об'єкт
6         int m_year; // відкрито, доступ має будь-який об'єкт
7 };
8
9 int main()
10 {
11     DateClass date;
12     date.m_day = 12; // ок, так як m_day має специфікатор доступу public
13     date.m_month = 11; // ок, так як m_month має специфікатор доступу public
14     date.m_year = 2018; // ок, так як m_year має специфікатор доступу public
15
16     return 0;
17 }
```

Оскільки тепер члени класу DateClass є відкритими, то до них можна отримати доступ безпосередньо з функції main ().

Ключове слово **public** разом з двокрапкою називається **специфікатором доступу**. Специфікатор доступу визначає, хто має доступ до членів цього специфікатора. Кожен з членів «набуває» рівень доступу в відповідність до специфікатором доступу (або, якщо він не вказаний, у відповідність зі специфікатором доступу за замовчуванням).

У C++ є 3 рівня доступу :

public : робить члени відкритими;

private : робить члени закритими;

protected: відкриває доступ до членів тільки для дружніх і дочірніх класів.

Класи можуть використовувати відразу кілька специфікаторів доступу для установки рівнів доступу для кожного зі своїх членів. Зазвичай змінні-члени є закритими, а методи - відкритими.

Правило: Встановлюйте специфікатор доступу private змінним-членам класу і специфікатор доступу public - методам класу.

Розглянемо приклад класу, який використовує специфікатор доступу private і public:

```
1 #include <iostream>
2
3 class DateClass // члени класу являються закритими по умовчанняю
4 {
5     int m_day; // закрито по умовчанняю, доступ мають тільки інші члени класу
6     int m_month; // закрито по умовчанняю, доступ мають тільки інші члени класу
7     int m_year; // закрито по умовчанняю, доступ мають тільки інші члени класу
8
9     public:
10    void setDate(int day, int month, int year) // відкрито, доступ має будь-який об'єкт
11    {
12        // Метод setDate() має доступ до закритих членів класу, так як сам є членом класу
13        m_day = day;
14        m_month = month;
15        m_year = year;
16    }
17
18    void print() // відкрито, доступ має будь-який об'єкт
19    {
20        std::cout << m_day << "/" << m_month << "/" << m_year;
21    }
22 };
23
24 int main()
25 {
26     DateClass date;
27     date.setDate(12, 11, 2018); // ок, так як setDate() має специфікатор доступу public
28     date.print(); // ок, так як print() має специфікатор доступу public
29
30     return 0;
31 }
```

Результат виконання програми:

12/11/2018

Зверніть увагу, хоч і не можемо отримати доступ до змінних-членам об'єкта

date безпосередньо з main () (так як вони є private за замовчуванням), ми можемо отримати доступ до них через відкриті методи setDate () і print ().

Відкриті члени класів складають відкритий (або ще «public») інтерфейс. Оскільки доступ до відкритих членів класу може здійснюватися ззовні класу, то відкритий інтерфейс і визначає, як програми, що використовують клас, будуть взаємодіяти з цим же класом.

Розглянемо наступну програму:

```
1 #include <iostream>
2
3 class DateClass // члени класу являються закритими по умовчанням
4 {
5     int m_day; // закрито по умовчанням, доступ мають тільки інші члени класу
6     int m_month; // закрито по умовчанням, доступ мають тільки інші члени класу
7     int m_year; // закрито по умовчанням, доступ мають тільки інші члени класу
8
9 public:
10    void setDate(int day, int month, int year)
11    {
12        m_day = day;
13        m_month = month;
14        m_year = year;
15    }
16
17    void print()
18    {
19        std::cout << m_day << "/" << m_month << "/" << m_year;
20    }
21
22    // Обратите внимание на этот дополнительный метод
23    void copyFrom(const DateClass &b)
24    {
25        // Мы имеем прямой доступ к закрытым членам объекта b
26        m_day = b.m_day;
27        m_month = b.m_month;
28        m_year = b.m_year;
29    }
30 };
31
32 int main()
33 {
34     DateClass date;
35     date.setDate(12, 11, 2018); // ок, так как setDate() имеет спецификатор доступа public
36
37     DateClass copy;
38     copy.copyFrom(date); // ок, так как copyFrom() имеет спецификатор доступа public
39     copy.print();
40
41     return 0;
42 }
```

Один нюанс в C ++, який часто ігнорують, про який забувають або неправильно розуміють, полягає в тому, що **контроль доступу працює на основі класу** , а не на основі об'єкта. Це означає, що коли метод має доступ до закритими членам класу, то він може звертатися до закритих членам будь-якого об'єкта цього класу.

В наведеному вище прикладі метод `copyFrom ()` є членом класу `DateClass`, що відкриває йому доступ до `private` членам класу `DateClass`. Це означає, що `copyFrom ()` може не тільки безпосередньо звертатися до закритих членам неявного об'єкта з яким працює (копія об'єкта), але і має прямий доступ до закритих членам об'єкта `b` класу `DateClass`!

Це корисно, коли потрібно скопіювати елементи з одного об'єкта класу в інший об'єкт того ж класу.

Завдання №2

а) Напишіть простий клас з ім'ям `Numbers`. Цей клас повинен мати:

- *три закриті змінні-члени типу `double`: `m_a`, `m_b` і `m_c`;*
- *відкритий метод з ім'ям `setValues ()`, який дозволить встановлювати значення для `m_a`, `m_b` і `m_c`;*
- *відкритий метод з ім'ям `print ()`, який буде виводити об'єкт класу `Numbers` в наступному форматі: `<m_a, m_b, m_c>`.*

Наступний код функції `main ()`:

```
1 int main()
2 {
3     Numbers point;
4     point.setValues(3.0, 4.0, 5.0);
5
6     point.print();
7
8     return 0;
9 }
```

Результат виконання:

```
<3, 4, 5>
```