Об’єктно-орієнтоване програмування

ОО Програмування — це методологія програмування, що базується на представленні програми у вигляді сукупності ***об’єктів***, кожний з яких є екземпляром ***класу***, а класи створюють ***ієрархії***.

- Програма є об’єктно-орієнтованою при наявності всіх 3-х складових (об’єктів, класів, ієрархії).

Мова є об’єктно-орієнтованою, якщо:

- Підтримує об’єкти (інтерфейс у вигляді іменованих операцій та власні дані з обмеженим доступом);
- Об’єкти відносяться до відповідних типів (класів);
- Типи (класи) можуть успадковувати атрибути супертипів (відношення спадкування “is a” “це є “).

Складові частини ОО підходу

- Абстракція
- Інкапсуляція
- Модульність
- Ієрархія

Приклад

```
#include <iostream.h>
//Объявление класса
class complex_c1 {
 public:
    void assign(double r, double i); // функція-член або метод
    void  print() // функція-член або метод
       { cout << real << " + " << imag << "i ";}
 private:
    double  real, imag; // змінні-члени
 };
inline void complex_c1::assign(double r, double i = 0.0)
 {
    real = r;
    imag = i;
 }
```

Абстракція

- **Абстракція** — це відмова від зайвих деталей класу об`єктів та виділення важливих деталей їх структури та поведінки.
- Інтерфейс об’єкту містить важливі аспекти поведінки. Абстракція дозволяє реалізувати відношення між об’єктами: *клієнт-сервер* — один об’єкт (клієнт) використовує ресурси іншого об’єкта (сервера), на основі якої формується контрактна модель програми, тобто внутрішня структура об’єкта визначається контрактом.

Зовнішня взаємодія — контракт:

- 1. Кожна операція задається формальними параметрами та типом значення, що повертається;
- 2. Повний набір операцій — протокол (містить всі засоби) — *поліморфізм*;
- 3. Інваріант — деяка логічна умова, значення якої повинно зберігатися (передумова та постумова).

Приклад

```
#include <iostream.h>
//Объявление класса – абстрактного типа данных
class complex_c1 {
 public:
    void assign(double r, double i);
    void  print()
       { cout << real << " + " << imag << "i ";}
 private:
    double  real, imag;
 };
inline void complex_c1::assign(double r, double i = 0.0)
 {
    real = r;
    imag = i;
 }
```

Інкапсуляція

Інкапсуляція – це процес відокремлення одного елементу від інших елементів об’єкту, що визначають його структуру від поведінки. Призначена для того, щоб ізолювати контрактні обов’язки абстракції від їх реалізації ⇒ дозволяє створити уявність простоти програми.

Приклад:

```
#include <iostream.h>
//Объявление класса
class complex_c1 {
 public:
    void assign(double r, double i);
    void  print()
       { cout << real << " + " << imag << "i ";}
 private:
    double  real, imag;
 };
```

Модульність

Модульність — це властивість системи бути розкладеною на пов’язані між собою частини (модулі).

Модулі — об’єктні:

- виконують роль фізичних контейнерів, що містять визначення класів та об’єктів;
- розроблюються різними людьми;
- модулі можуть компілюватися окремо;
- в один модуль групуються логічно пов’язані класи та об’єкти.

Приклад

```
#include <iostream.h>
//Объявление клааса
class complex_c1 {
 public:
    void assign(double r, double i);
    void  print()
       { cout << real << " + " << imag << "i ";}
 private:
    double  real, imag;
 };
inline void complex_c1::assign(double r, double i = 0.0)
 {
    real = r;
    imag = i;
 }
```

Ієрархія

Ієрархія — це впорядкування абстракцій та розміщення їх по рівням.

Основні види ієрархії:

- 1. **Структура класів** “is a” (узагальнення-спеціалізація)
  - Одиночне успадкування — від одного класу;
  - Множинне успадкування — від багатьох (декількох класів).
- 2. **Структура об’єктів** “part of” (агрегація структури [включення])

Основні властивості ОО програмування

До основних принципів **ОО програмування** відносяться наступні:

1. *Абстракція* — виділення головного.
2. *Інкапсуляція* — можливість приховати внутрішні деталі під час опису загального інтерфейсу. *Ключові слова* private, public, protected визначають рівень доступу для забезпечення інкапсуляції (на рівні об'єктів);
3. *Успадкування* — передача властивостей від попередника (отримання властивостей з протоколу базового класу). Засіб отримання нових класів з існуючих;
4. *Поліморфізм* — умови, в яких вид має різні морфологічні форми. Окремі повідомлення можуть викликати різноманітні дії. Спосіб реалізації — перевантаження операцій та функцій – паралельний поліморфізм, коли тип невизначений, а його значення вказується пізніше (в C++ — базові вказівники та шаблони).

Основна концепція ООП

*Основна концепція ООП — **передача повідомлень** об'єктам.*

- ***Клас*** — ключове слово "class" — **абстрактний тип даних (АТД)** *(дозволяє керувати видимістю того, що є в основі виконання)..*
- ***Об'єкт*** — змінна типу ClassName, де ClassName — визначений раніше клас.
- ***Дані стану*** оголошуються в описі класу і називаються полями даних, даними-членами чи членами.

```
int main()
{
  complex_c1  b; // complex_c1 – АТД, и b - змінна
  complex_c1 * c; // complex_c1 * – вказівник на АТД, и c - змінна
  b.assign(3.3, 4.4); // передача повідомлення об'єкту
  c=new complex_c1;
  c->assign(5.5, 6.6); // неявний доступ до об'єкта через вказівник
  cout << "\nComplex from class definition: ";
  b.print();
  cout << "\nComplex from explicit class definition: ";
  c->print();
}
```

Висновок: основна концепція ООП

Основна концепція ООП — передача повідомлень об'єктам.

Абстрактні типи даних в C++ реалізуються за допомогою механізму **класів** (дозволяє керувати видимістю того, що є в основі виконання).

- **Клас** — ключове слово "Class" — абстрактний тип даних.
- **Об'єкт** — змінна типу ClassName, де ClassName — визначений раніше клас.
- **Дані стану** оголошуються в описі класу і називаються **полями даних**, **даними-членами** чи **змінними-членами**.

Основна концепція ООП

- *Повідомлення* – вказуються за допомогою прототипів функцій в описі класу. *Прототип функції* — це тип значення, що повертається, ім’я функції та список параметрів. *Список параметрів* включає типи параметрів та необов’язкові імена параметрів.
- *Метод* в C++ — це визначення (реалізація) функції. Прототипи функції та визначення називаються функціями-членами класу. *Члени класу* — це поля даних та функції-члени.
- *Підклас* — похідний клас. Його суперклас чи базовий клас називається базовим.