27. Поліморфізм без явного вказівника

Ключевое слово this представляет собой неявно определенный указатель на сам объект. С его помощью метод класса определяет, с данными какого объекта ему предстоит работать. Каждый метод класса неявно содержит в качестве поля данных указатель:

**ИмяКласса \*this;**

При вызове метода ему передается неявный аргумент, содержащий адрес объекта, для которого эта функция вызывается.

class example {

  int m;

public:

  int readm() { return m; }  // return this->m

};

void f() {

  example aa, bb;

  int a = aa.readm();  // this указывает на aa

  int b = bb.readm(); // this указывает на bb

}

В первом случае функции readm() неявно передается указатель на объект aa, а во втором случае – bb.

Использование this необходимо в функциях, которые непосредственно работают с указателем на объект:

* this  – указатель на объект (адрес объекта)
* \*this  – разыменованый указатель (сам объект)

Указатель this удобно использовать, например, в конструкторах, когда имена передаваемых параметров совпадают с именами полей класса.

сlass date

{

  int day, month, year;

public:

  date(int day, int month, int year)

  {

    this->day = day; //поле класса = аргумент

    this->month = month;

    this->year = year;

  }

};

В случае если аргументы и поля класса имеют разные имена, указатель this при обращении к полям класса может быть опущен. Но в приведенном выше примере использование this указывает, что мы ходим обратиться именно к полю класса.

Внутри тела класса могут использоваться ссылки или указатели на тот же класс. Это позволяет строить рекурсивные структуры класса.

Указатель this может быть использован только для нестатического метода.

#include <iostream>

using namespace std;

class example {

private:

  char c1, c2;

public:

  void init(char b) { c1 = b; c2 = b + 1; }

  example &increment() { c1++; c2++; return(\*this); }

  example \*where() { return(this); }

  void print() { cout << c1 << " " << c2 << endl; }

};

int main() {

  example a, b;

  a.init('A');  // a.c1=’A’, a.c2=’B’

  b.init('B');  // b.c1=’B’, b.c2=’C’

  a.print();  // A B

  cout << "&a = " << a.where() << endl; // &a=########

  a.increment().print();  // B C

  b.increment().print();  // C D

  cin.get();

  return 0;

}

Результат выполнения:

