### *Реферат на тему:*

### Програмування на С і С++. Вказівник this

Розробником мови програмування Сі++ є Бьєрн **Страуструп**. У своїй роботі він спирався на досвід розробників мов Сімула, Модула 2, абстрактних типів даних. Основні роботи велися в дослідницькому центрі компанії Bell Labs.

Безпосередній попередник Сі++ – мова Сі з класами – з'явився в 1979 році, а в 1997 році був прийнятий міжнародний стандарт Сі++, що фактично підвів підсумки його 20-літнього розвитку. Прийняття стандарту забезпечило однаковість усіх реалізацій мови Сі++. Не менш важливим результатом стандартизації стало те, що в процесі вироблення і утвердження стандарту мови було уточнено і доповнено ряд істотних можливостей.

На сьогодні стандарт затверджений Міжнародною організацією по стандартзації ISO. Його номер **ISO/IEC 14882**.

Мова Сі++ є універсальною мовою програмування, на додаток до якої розроблений набір різноманітних бібліотек. Тому, чесно говорячи, він дозволяє вирішити практично будь-яку задачу програмування. Проте, у силу різних причин (не завжди технічних) для якихось типів задач він вживається частіше, а для якихось – рідше.

Сі++ як спадкоємець мови Сі широко використовується в системному програмуванні. На ньому можна писати високоефективні програми, у тому числі операційні системи, драйвери і т.п. Мова Сі++ – одна з основних мов розробки трансляторів.

Оскільки системне програмне забезпечення часте буває написано мовою Сі чи Сі++, той і програмний інтерфейси до підсистем ОС теж часто пишуть на Сі++. Відповідно, ті програми, навіть і прикладні, котрі взаємодіють з операційними системами, написані мовою Сі++.

Розподілені системи, що функціонують на різних комп'ютерах, також розробляються мовою Сі++. Цьому сприяє те, що в широко розповсюджених компонентих моделей CORBA і COM є зручні інтерфейси мовою Сі++.

Обробка складних структур даних – тексту, бізнес-інформації, Internet-сторінок і т.п. – одна з найбільш розповсюджених можливостей **застосування** мови. У прикладному програмуванні, напевно, простіше назвати ті області, де мова **Сі++** застосовується мало.

Розробка графічного користувальницького інтерфейсу мовою Сі++ виконується, в основному, тоді, коли необхідно розробляти складні, нестандартні інтерфейси. Прості програми частіше пишуться на мовах Visual Basic, Java і т.п.

Програмування для Internet в основному виробляється на мовах Java, VBScript, Perl. У цілому треба сказати, що мова Сі++ у даний час є однією з найбільш розповсюджених мов програмування у світі.

 **Вказівник this**

    Ім'я **this** є службовим (ключовим) словом. Явно описати чи визначити вказівник **this** не можна. Відповідно до неявного визначення **this** є ***константним вказівником***, тобто змінювати його не можна, однак у кожної приналежної класу функції він указує саме на той об'єкт, для якого функція викликається. Говорять, що вказівник **this** є ***додатковим (схованим) параметром*** кожної нестатичної компонентної функції. Іншими словами, при вході в тіло приналежній класу функції вказівник **this** ініціалізується значенням адреси того об'єкта, для якого викликана функція. Об'єкт, що адресується вказівником **this**, стає доступним усередині приналежної класу функції саме за допомогою вказівника **this**. При роботі з компонентами класу усередині приналежної класу функції можна було б скрізь використовувати цей вказівник. Наприклад, зовсім правильним буде таке визначення класу:

struct ss

{ int si;

 char sc;

 ss(int in, char en) // Конструктор об'єктів класу.

 { this->si = in; this->sc = en; }

 void print(void) // Функція висновку відомостей про об'єкт.

 { cout << "\n si = " << this->si;

 cout << "\n sc = " << this->se; }

};

    При такому використанні вказівника **this** немає ніяких переваг, тому що дані конкретних об'єктів доступні в приналежних класу функціях і за допомогою імен даних класу. Однак у деяких випадках вказівник **this** корисний, а іноді просто незамінний. У наступному прикладі вказівник **this** дозволяє компілятору розібратися в ситуації, коли ім'я **Meaning** компонента класу збігається з ім'ям формального параметра приналежної класу функції:

//OOР11\_1.СРР - вказівник "this".

#include <iostream.h>

class cell // Клас "числовий елемент".

{

 int static Amount; // Загальна кількість елементів.

 int Number; // Порядковий номер елемента.

 double Meaning; // Значення елемента.

 public:

 //Конструктор:

 cell(double Meaning = 0.0)

 { // Змінюємо значення статичного компонента:

 Amount++;

 this -> Number = Amount;

 // Компонент і однойменний параметр:

 this -> Meaning = Meaning; }

 // Висновок зведень про об'єкт і кількість елементів:

 void display (void)

 {

 // Зайві this:

 cout << "\nNumber = " << this->Number;

 cout << " Amount = " << this->Amount;

 cout << " Meaning = " << this->Meaning;

 }

};

// Ініціалізація статичного компонента:

int cell::Amount = 0;

void main(void)

{ cell A; // Об'єкт зі значенням, що умовчується.

 A.display();

 cell B(200.0);

 cell C(300.0);

 B.display();

 C.display();

}

    Результат виконання програми:

 Number = 1 Amount = 1 Meaning = 0

 Number = 2 Amount = 3 Meaning = 200

 Number = 3 Amount = 3 Meaning = 300

Майже незамінним і дуже зручним вказівник **this** стає в тих випадках, коли ***в тілі приналежної класу функції потрібно явно задати адресу того об'єкта, для якого вона викликана***. Наприклад, якщо в класі потрібна функція, що поміщає адресу обраного об'єкта чи класу в масив яка включає конкретний об'єкт класу в список, те таку функцію складно написати без застосування вказівника **this**. Дійсно, при організації зв'язних списків, ланками яких повинні бути об'єкти класу, необхідно включати в зв'язку ланок вказівник саме на той об'єкт, що у даний момент обробляється. Це включення повинне виконати якась функція-компонент класу. Однак конкретне ім'я об'єкта, що включається, у момент написання цієї приналежної класу функції недоступно, тому що його набагато пізніше довільно вибирає програміст, використовуючи клас як тип даних. Можна передавати такої функції посилання чи вказівник на потрібний об'єкт, але набагато простіше використовувати вказівник **this**.

    Отже, повторимо, коли вказівник **this** використаний у функції, що належить класу, наприклад, з ім'ям **ZOB**, те він має за замовчуванням тип **ZOB \*const** і завжди дорівнює адресі того об'єкта, для якого викликана компонентна функція. Якщо в програмі для деякого класу **X** визначити об'єкт:

 X factor(5);

то при виклиці конструктора класу **X**, що створює об'єкт **factor**, значенням вказівника **this** буде **&factor**.