7-MA’RUZA.

This ko’rsatkichi.

Sinfning zanjirlash metodlari

*this yashirin ko‘rsatkichi

Ko‘pincha, agar sinfda metodni chaqirsangiz, C++ uni qaysi obyekt deb hisoblaydi, degan savol ko’p uchrab turadi. Bu holatda C++ yashirin ***this** ko‘rsatkichi ishlatadi deb javob berish mumkin.

***this yashirin ko‘rsatkichi.** Quyida butun sonli qiymatni o‘z ichiga olgan, konstruktori va yordamchi funksiyalari bo‘lgan oddiy sinf (-listing).

```
#include <iostream>
using namespace std;
class Value
{
private:
    int m_number;
public:
    Value(int number)
    {
        setNumber(number);
    }

    void setNumber(int number)
    {
     m_number = number;
    }
    int getNumber()
```

```
    {
        return m_number;
    }
};

int main()
{
    Value V(3);
    V.setNumber(4);
   cout << V.getNumber() << '\n';

    return 0;
}
```

Dasturni bajarish natijasi:

4

V.setNumber (4) ni chaqirganda C++ tushunadiki, setNumber() funksiyasi boshqa obyektda ishlaydi va m_number aslida V.m_number. Hammasi qanday ishlashini batafsil ko‘rib chiqaylik.

Masalan, quyidagi qatorni oling:

V.setNumber(4);

Garchi bir qarashda bizda faqat bitta bahs bordek tuyulsa-da, aslida ikkita savol bor. Kompilyatsiya vaqtida V.setNumber(4) qatori kompilyator tomonidan quyidagilarga o‘zgartiriladi:

setNumber(&V, 4); //boshqa obyekt nuqta oldidan obyekt argumentiga aylantirildi

Bu endi faqat standart funksiya chaqiruvi va V obyekti (ilgari alohida obyekt bo‘lgan va nuqtadan oldin kelgan) endi funksiyaga argument sifatida manzilga uzatiladi.

Funksiya chaqiruvida hozir ikkita argument mavjud bo‘lgani uchun, metod mos ravishda o‘zgartirilishi kerak (shuning uchun ikkita argument kerak bo‘ladi). Shunday qilib, quyidagi metod:

```
void setNumber(int number)
{
m_number = number;
}
```

kompilyator tomonidan quyidagicha konvertatsiya qilinadi:

```
void setNumber(V* const this, int number)
{
this->m_number = number;
}
```

Oddiy metodni tuzishda kompilyator unga *this parametrni bilvosita qo‘shadi. *this ko‘rsatkich sinf metodini chaqiradigan obyektning manzilini o‘z ichiga olgan yashirin doimiy ko‘rsatkichdir.

Yana bitta tafsilot bor. Metod ichida sinfning barcha a'zolarini (funksiyalar va o‘zgaruvchilar) yangilashingiz kerak, shunda ular ushbu metodni chaqiradigan obyektga murojaat qilishadi. Buni har biriga **this->** prefiksini qo‘shish orqali qilish oson. Shunday qilib, setNumber() funksiyasining tanasida m_number (sinfning a'zo o‘zgaruvchisi) this->m_numberga aylanadi. Qachon *this boshqasining

manzilini ko‘rsatsa, this->m_number V.m_numberga ishora qiladi.

Yuqoridagilarni quyidagilarni umumlashtirib quyidagilarni aytish mumkin:

– V.setNumber(4) ga murojaat qilganda, kompilyator aslida setNumber(&another, 4) ni chaqiradi.

– setNumber() ichida *this ko‘rsatkich boshqasining manzilini o‘z ichiga oladi.

– setNumber() ichidagi har qanday a'zo o‘zgaruvchilar -> prefiksi orqali murojaat qilinadi.

– Shunday qilib, biz m_number = number deb aytganimizda, kompilyator aslida this-> m_number = number, ni bajaradi bu holda.

Muhim jihati shundaki, bularning barchasi bizdan (dasturchilar) yashiringan va bu qanday ishlashini eslaysizmi yoki yo‘qmi, muhim emas. Shuni yodda tutish kerakki, barcha oddiy sinf usullarida * metodikasi chaqiruvi bilan bog‘liq obyektni ko‘rsatuvchi ko‘rsatgich mavjud.

***this ko‘rsatkichi har doim joriy obyektga ishora qiladi.** Endigina dasturlashni o’rganishni boshlagan dasturchilar ba'zida nechta *this ko‘rsatkichlari borligi haqida chalkashib ketishadi. Har bir metodda parametr sifatida *this ko‘rsatkichi mavjud bo‘lib, u amalda bajarilayotgan obyektning manzilini ko‘rsatadi, masalan:

```
int main()
{

Value X(3); // * this = & X Value konstruktor ichida

    Value Y(4); // *this = &Y Value konstruktor ichida

    X.setNumber(5); // * this = & X setNumber metodi ichida

    Y.setNumber(6); // * this = & Y setNumber metodi ichida
    return 0;
}
```

E'tibor bering, *this ko‘rsatkich navbat bilan X yoki Y obyektlarining manzilini o‘z ichiga oladi, qaysi metod chaqirilishiga va hozirda bajarilishiga bog‘liq.

***this ko‘rsatkichiga oshkor ko‘rsatkich.** Ko‘pgina hollarda, *this ko‘rsatkichni aniq ko‘rsatish shart emas. Biroq, bu ba'zida foydali bo‘lishi mumkin. Misol uchun, agar parametr o‘zgaruvchisi bilan bir xil parametrga ega bo‘lgan konstruktor (yoki metod) bo‘lsa, uni *this ko‘rsatkich yordamida ajratib ko‘rsatish mumkin:

```
class Something
{
private:
    int data;

public:
    Something(int data)
```

```
    {
        this->data = data;
    }
};
```

Bu yerda konstruktor a'zo o‘zgaruvchi bilan bir xil nomdagi parametrni oladi. Bunday holda, ma'lumotlar parametrga va this->data a'zo o‘zgaruvchiga tegishli.

2. Sinfning zanjirlash metodlari

Ba'zida sinf metodi uchun ishlayotgan obyektni qaytarish qiymati sifatida qaytarish foydalidir. Bu yerda asosiy nuqta - bitta obyekt ustida ishlayotganda bir nechta metodlarni bir-biriga bog‘lab qo‘yishdir. Aslida biz buni uzoq vaqtdan beri ishlatamiz. Masalan, ma'lumotlarni cout bilan qismlarga ajratganimizda:

cout << "Assalomu alaykum, " << userName;

Bunday holda, cout - bu obyekt, << operatori - bu shu obyektda ishlaydigan metod. Kompilyator yuqoridagi fragmentni quyidagicha ishlatadi:

(cout << " Assalomu alaykum, ") << userName;

Birinchidan, << operatori konsolga “Assalomu alaykum”ni chop etish uchun cout va " Assalomu alaykum" satrini ishlatadi. Biroq, bu ifodaning bir qismi bo‘lgani uchun << operatori ham qiymatni (yoki bo‘shliqni) qaytarishi kerak. Agar << operatori bekor qilsa, quyidagilar olinadi:

(void)<< userName;

Bu hech qanday ma'noga ega emas (kompilyator xato qiladi). Biroq, buning o‘rniga, << operatori * this ko‘rsatkichini qaytaradi, bu kontekstda shunchaki cout. Shunday qilib, birinchi << operatorini qayta ishlagandan so‘ng, biz:

(cout) << userName;

Natijada foydalanuvchi nomi (userName) chiqadi.

Shunday qilib, biz obyektni (bu holda, cout) bir marta ko‘rsatishimiz kerak va har bir funksiya chaqiruvi bu obyektni keyingi funksiyaga o‘tkazadi, bu bizga bir nechta metodlarni birlashtirishga imkon beradi.

O‘zimiz bu xatti-harakatni amalga oshirishimiz mumkin. Quyidagi sinfni ko‘rib chiqaylik (-listing).

Agar 7 ni qo‘shib, 5 ni ayirib, hammasini 3 ga ko‘paytirmoqchi bo‘lsangiz, quyidagilarni bajarishingiz kerak.

```
#include <iostream>
using namespace std;
class Mathem
{
private:
    int m_value;

public:
    Mathem()
    {
        m_value = 0;
    }

    void add (int value)
```

```
    {
        m_value += value;
    }
    void sub(int value)
    {
        m_value -= value;
    }
    void multiply(int value)
    {
        m_value *= value;
    }

    int getValue()
    {
        return m_value;
    }
};
```

```
int main()
{
   Mathem operation;
   operation.add(7); // void
   operation.sub(5); // void
   operation.multiply(3); // void

 cout << operation.getValue() << '\n';
   return 0;
}
```

Dastur natijasi:

6

Ammo, agar har bir funksiya *this ko‘rsatkichni qaytarsa, biz bu metod chaqiruvlarini birgalikda *zanjirlashimiz* mumkin. Masalan:

add(), sub() va multiply() endi *this ko‘rsatkichini qaytaradi, shuning uchun quyidagilar to‘g‘ri bo‘ladi:

```
#include <iostream>
using namespace std;
class Mathematic
{
private:
    int m_value;
public:
    Mathematic()
    {
        m_value = 0;
    }
```

```
Mathematic& add(int value)
{
    m_value += value;
    return *this;
}
Mathematic& sub(int value)
{
    m_value -= value;
    return *this;
}
Mathematic& multiply(int value)
{
    m_value *= value;
    return *this;
}

int getValue()
```

```
    {
        return m_value;
    }
};
int main()
{
    Mathematic operation;
    operation.add(7).sub(5).multiply(3);
    cout << operation.getValue() << '\n';
    return 0;
}
```

Dastur natijasi:

6

Bu dasturda (-listingga qarang) bitta operatorga uchta alohida qatorni joylashtirildi. Endi buni batafsil ko‘rib chiqaylik:

- Operation.add(7) birinchi bo‘lib chaqiriladi, bu m_value maydoniga 7 ni qo‘shadi.
- Keyin add() operation obyektiga havola bo‘lgan this* ko‘rsatkichni qaytaradi.
- So‘ngra operatorion.sub(5) ga chaqiruv m_value dan 5 ni olib tashlaydi va operationni qaytaradi.
- multiply(3) m_value ni 3 ga ko‘paytiradi va allaqachon e'tiborga olinmagan operation ni qaytaradi.
- Biroq, har bir funksiya operation ni o‘zgartirganligi sababli, operation obyektining m_value qiymati endi ((0 + 7) - 5) * 3) o‘z ichiga oladi, demak natija 6.