Программирование на языке C++
Лекция 7

Информации о типах времени выполнения

Александр Смаль

Run-Time Type Information (RTTI)

В C++ этот механизм состоит из двух компонент:

1. оператор `typeid` и тип `std::type_info`,
2. оператор `dynamic_cast`.

Run-Time Type Information (RTTI)

В C++ этот механизм состоит из двух компонент:

1. оператор `typeid` и тип `std::type_info`,
2. оператор `dynamic_cast`.

Тип `type_info`

- Класс, объявленный в `<typeinfo>`.
- Содержит информацию о типе.
- Методы: `==`, `!=`, `name`, `before`.
- Нет публичных конструкторов и оператора присваивания.
- Можно получить ссылку на `type_info`, соответствующий значению или типу, при помощи оператора `typeid`.

Использование typeid и type_info

```
struct Unit {
    // наличие виртуальных методов необходимо
    virtual ~Unit() { }
};

struct Elf : Unit { };

int main() {
    Elf e;
    Unit & ur = e;
    Unit * up = &e;
    cout << typeid(ur) .name() << endl; // Elf
    cout << typeid(*up).name() << endl; // Elf
    cout << typeid(up) .name() << endl; // Unit *
    cout << typeid(Elf).name() << endl; // Elf
    cout << (typeid(ur) == typeid(Elf)); // 1
}
```

Преобразования в C++: dynamic_cast

Преобразования с проверкой типа времени выполнения.

```
Unit * u = (rand() % 2)? new Elf(): new Dwarf();
...
if (Elf * e = dynamic_cast<Elf *>(u))
    ...
else if (Dwarf * d = dynamic_cast<Dwarf *>(u))
    ...
```

Преобразования в C++: dynamic_cast

Преобразования с проверкой типа времени выполнения.

```
Unit * u = (rand() % 2)? new Elf(): new Dwarf();
...
if (Elf * e = dynamic_cast<Elf *>(u))
    ...
else if (Dwarf * d = dynamic_cast<Dwarf *>(u))
    ...
```

Особенности:

- Не заменяется преобразованием в стиле C.
- Требует наличие виртуальных функций (полиморфность).

Преобразования в C++: dynamic_cast

Преобразования с проверкой типа времени выполнения.

```
Unit * u = (rand() % 2)? new Elf(): new Dwarf();
...
if (Elf * e = dynamic_cast<Elf *>(u))
    ...
else if (Dwarf * d = dynamic_cast<Dwarf *>(u))
    ...
```

Особенности:

- Не заменяется преобразованием в стиле C.
- Требует наличие виртуальных функций (полиморфность).

Вопросы:

- Почему следует избегать RTTI?
- Что возвращает `dynamic_cast<void *>(u)`?

Пример обхода dynamic_cast: double dispatch

```
struct Rectangle; struct Circle;

struct Shape {
    virtual ~Shape() {}
    virtual bool intersect( Rectangle * r ) = 0;
    virtual bool intersect( Circle    * c ) = 0;
    virtual bool intersect( Shape     * s ) = 0;
};

struct Circle : Shape {
    bool intersect( Rectangle * r ) { ... }
    bool intersect( Circle    * c ) { ... }
    bool intersect( Shape     * s ) {
        return s->intersect(this);
    }
};
bool intersect(Shape * a, Shape * b) {
    return a->intersect(b);
}
```