Kategorie wartości wyrażeń

dr inż. Ireneusz Szcześniak

jesień 2018 roku

Wartość wyrażenia

Wyrażeniem jest:

- literał: `3.14`
- zmienna: `x`
- operator z operandami: `x + y`
- wywołanie funkcji: `foo(x)`

Wartość wyrażenia jest wynikiem opracowania wyrażenia.

Wartość wyrażenia ma:

- typ (np. int, bool, klasa A) - znany w czasie kompilacji,
- wartość typu (np. 5, false, A()) - znana w czasie uruchomienia,
- kategorię (l-wartość lub r-wartość) - znana w czasie kompilacji.

Historia: CPL, C, C++98

Dwie kategorie wartości wyrażenia po raz pierwszy wprowadzono w języku CPL:

- lvalue ("left of assignment" value),
- rvalue ("right of assignment" value).

W języku C wartości są "lvalue" (locator value), albo "non-lvalue". Wartości "lvalue" posiadają miejsce w pamięci, np. zmienne.

C++98 przyjęło definicję z C, ale nazwało "non-lvalue" jako "rvalue".

Po polsku mówimy l-wartość i r-wartość.

Kategoria wartości wyrażenia

Dwie podstawowe kategorie to: l-wartość i r-wartość. Wyrażenie kategorii l-wartość nazywamy l-wartością, a wyrażenie kategorii r-wartość nazywamy r-wartością.

Kategoria wartości wyrażenia określa, jakie operacje możemy dalej wykonać na opracowanym wyrażeniu. Niektóre operacje możemy wykonać wyłącznie na l-wartości (np. `&`), inne wyłącznie na r-wartości (np. `+` dla typu `int`).

Przykładowe operacje na wyrażeniu `x`:

- przypisanie: `x = 1`
- inicjalizacja referencji: `<typ referencyjny> y = x`
- pobranie adresu: `&x`
- wyłuskanie: `*x`
- inkrementacja: `++x, x++`

Nowoczesny C++: zmiany, zmiany, zmiany

W nowoczesnym C++ wprowadzono nowe kategorie wartości wyrażeń (pr-wartość, gl-wartość, x-wartość), jednak podstawą ciągle są l-wartości i r-wartości.

Aby precyzyjnie posługiwać się nowoczesnym C++ należy opanować podstawy kategorii wartości wyrażeń. Na przykład, poniższe kryptyczne zdanie jest częścią definicji l-wartości w C++11!

"Even if the variable's type is *rvalue reference*, the expression consisting of its name is an *lvalue expression*."

Kategoria l-wartość w C++

Jeżeli kategoria wartości wyrażenia to l-wartość, to można pobrać adres, czyli można wykonać: `&(wyrażenie)`.

Przykłady l-wartości:

- nazwa zmiennej: `x`
- nazwa funkcji: `foo`
- literał ciągu znaków: `"Hello␣World!";`
- `++i`

Kategoria r-wartość w C++

Jeżeli wyrażenie nie jest l-wartością, to jest r-wartością. Nie można pobrać adresu r-wartości.

Przykłady r-wartości:

- literał: `1`
- `std::string("Hello␣World!");`
- `i++`
- wywołanie funkcji: `foo()`, jeżeli `int foo();`

Konwersja z l-wartości na r-wartość

L-wartość może być poddana standardowej konwersji do r-wartości.

Na przykład, operator `+` dla typu `int` wymaga r-wartości jako swoich operandów, dlatego poniższy przykład jest poprawny, bo l-wartości, którymi są wyrażenia `x` i `y`, zostaną poddane konwersji do r-wartości.

```
int x, y;
x + y;
```

Nie ma niejawnej konwersji z r-wartości do l-wartości.

Przykład: operator ++

Operator inkrementacji (++) jako argumentu wymaga l-wartości. Wyrażenie z operatorem ++ jest:

- l-wartością dla operatora prefixowego, np. `++i`, bo operator wykonuje działanie na obiekcie `i`, a potem zwraca l-referencję na ten obiekt,
- r-wartością dla operatora sufixowego, np. `i++`, bo operator tworzy kopię obiektu `i`, wykonuje działanie na obiekcie `i`, a potem zwraca stworzoną kopię obiektu.

Dlatego `++++i` kompiluje się, a `i++++` już nie.

Obiekty tymczasowe

Obiekt tymczasowy (ang. temporary) jest tworzony przy:

- opracowywaniu wyrażenia: `1 + 2, string("T") + "4";`
- przekazywaniu argumentu wywołania funkcji:
  ```
  void foo(const A &);
  foo(A());
  ```
- zwracaniu wartości funkcji: `string x = foo();`
- rzucaniu wyjątku: `throw A();`

Obiekty tymczasowe w C++98 były zawsze kopiowane.

Obiekty tymczasowe w C++11 mogą być *przenoszone*, bo wyrażenia, w których są tworzone (np. jako argument wywołania funkcji) są r-wartością, którą można wskazać r-referencją.

Funkcje a kategorie wartości wyrażeń

Funkcja o nazwie `foo` może być użyta w wyrażeniu na dwa sposoby:

- użycie tylko nazwy:
  - przykładowe wyrażenie: `foo`,
  - w tym przypadku `foo` jest l-wartością, bo na nazwę,
  - możemy pobrać adres l-wartości, więc `&foo` jest OK,
- wywołanie funkcji:
  - przykładowe wyrażenie: `foo(argumenty)`,
  - kategoria wartości wyrażenia wywołania funkcji zależy od typu wartości zwracanej przez funkcję:
    - jeżeli typem jest l-referencja, to wyrażenie jest l-wartością, np.: jeżeli `int &foo();`, to `foo()` jest l-wartością,
    - jeżeli typ nie jest referencyjny, to wyrażenie jest r-wartością: np.: jeżeli `int foo();`, to `foo()` jest r-wartością.

Typy niepełne a kategorie wartości wyrażeń

Jeżeli typ nie został w pełni zdefiniowany, bo został jedynie zadeklarowany, albo częściowo zdefiniowany jako klasa abstrakcyjna, to typ taki nazywamy typem niepełnym (ang. incomplete type).

Wyrażenia typu niepełnego mogą mieć tylko kategorię l-wartość.

```
class B;

B &
boo()
{
}

int main()
{
  B &br = boo();
  // B(); - błąd, bo r-wartość
}
```

Podsumowanie

- Kategoria wartości odnosi się do wyrażenia, a nie wartości pewnego typu (np. klasy A albo typu `int`).
- Wartość pewnego typu (np. klasy A albo typu `int`) nie ma kategorii wartości.
- Kategoria wartości wyrażenia określa, jakie operacje można dalej wykonać na wyrażeniu.
- Każde wyrażenie jest l-wartością albo r-wartością.

Dziękuję za uwagę.