

Kompilacja NianioLanga do efektywnych konstrukcji języka C

15.06.2018

- Język imperatywny bez wskaźników
- Niemutowalne zmienne
- Opcjonalne typowanie

Umożliwienie generowania efektywnego kodu wynikowego C

- Brak kopiowania
- Jeden właściciel w dowolnej chwili
- Zmienne kompilowane to prostych struktur języka C
- Zmienne na stosie tam, gdzie to możliwe

# Porównanie kodu wynikowego

`a[0] -> b++`

# Porównanie kodu wynikowego

a[0]->b++

```
move(&nl_2,get_const(0));  
move(&nl_1,get_arr(nl_0,  
    nl_2));
```

```
move(&nl_3,get_const("b"));  
move(&nl_3,get_hash(nl_1,  
    nl_3));
```

```
move(&nl_4,get_const(1));  
move(&nl_3,add(nl_3, nl_4));
```

```
move(&nl_5,get_const("b"));  
set_hash(&nl_1,nl_5,nl_3);  
set_arr(&nl_0,nl_2,nl_1);
```

# Porównanie kodu wynikowego

a[0]->b++

```
move(&nl_2,get_const(0));  
move(&nl_1,get_arr(nl_0,  
    nl_2));
```

```
move(&nl_3,get_const("b"));  
move(&nl_3,get_hash(nl_1,  
    nl_3));
```

```
move(&nl_4,get_const(1));  
move(&nl_3,add(nl_3, nl_4));
```

```
move(&nl_5,get_const("b"));  
set_hash(&nl_1,nl_5,nl_3);  
set_arr(&nl_0,nl_2,nl_1);
```

```
nl_int_2 = 0;  
nl_rec_ptr_1 =  
    type0get_ptr(&nl_arr_0,  
        nl_int_2);  
nl_int_ptr_3 =  
    &(nl_rec_ptr_1->b0field);
```

```
nl_int_4 = 1;  
(*nl_int_ptr_3) =  
    (*nl_int_ptr_3) +  
    nl_int_4;
```

- Przyspieszenie w stosunku do starych typów 10-100 razy
- Testy automatyczne dla nowych typów
- Skompilowanie kompilatora nim samym



# Demo