Алгоритмы и Алгоритмические Языки

Семинар #19:

1. Переопределение операций.
2. Операция присваивания. Семантики копирования.
3. Конструктор копирования vs операция присваивания.
4. Наблюдение за «Автоматикой» C++.

24.01.2025

Переопределение операций (operator overloading)

Перегрузка операций в C++

Перегружать можно:

1. Операции: `operator` *`op`*, где *`op`* ∈ {`+ - * / % ^ & | ~ ! = < > += -= *= /= %= ^= &= |= << >> >>= <<= == != <= >= <=> && || ++ -- , ->* -> () [] }`
2. Операцию неявного приведения типа: `operator` *`type`*.
3. Операции аллокации памяти:
   `operator new, operator new[]`
4. Операции освобождения памяти:
   `operator delete, operator delete[].`
5. Создание объекта на основе литерала:
   `operator “”`*`suffix`*.

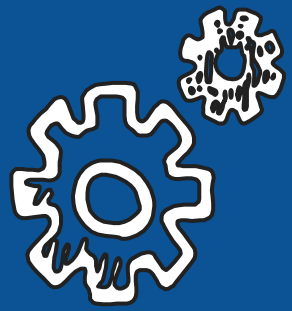
Перегрузка операций в C++

Перегруженный оператор – это функция или метод класса:

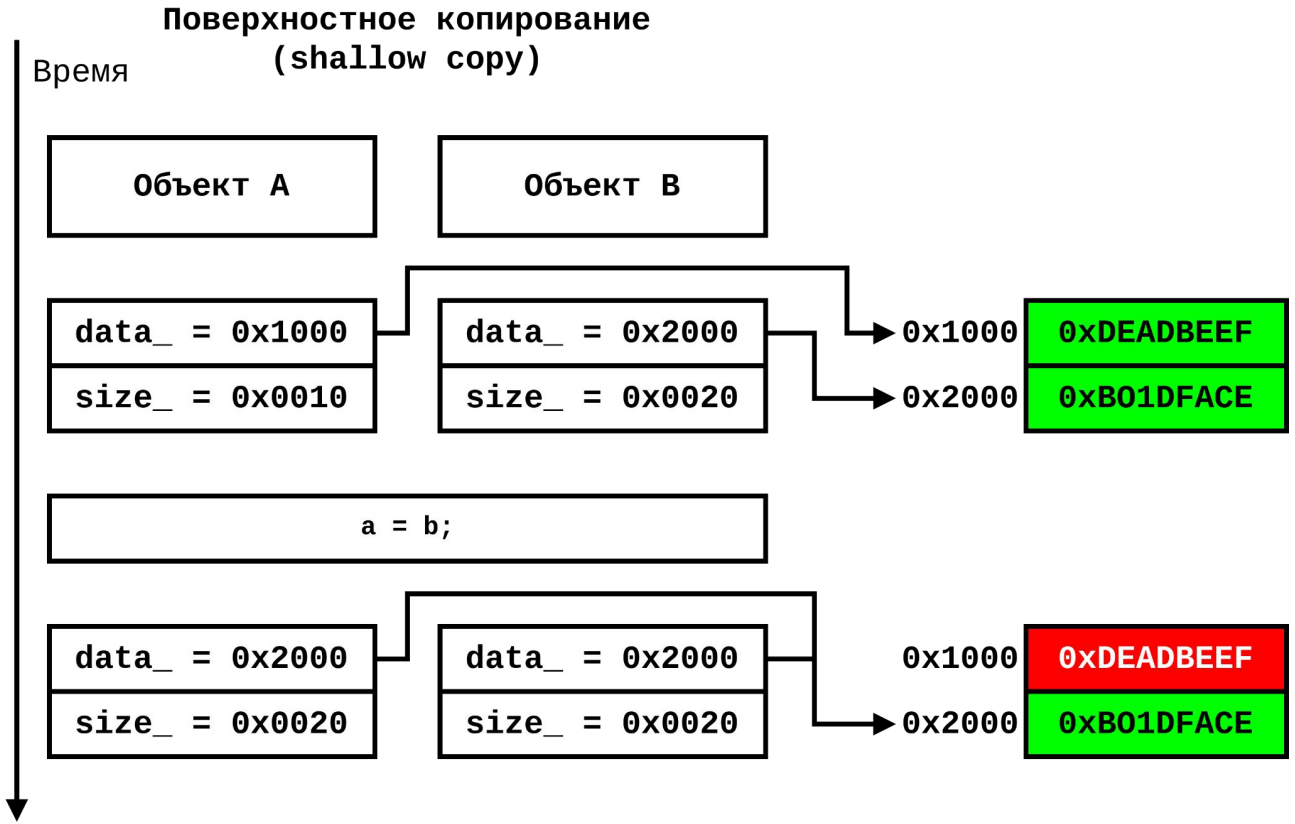
Expression	As member function	As non-member function
@a	(a).operator@ ( )	operator@ (a)
a@b	(a).operator@ (b)	operator@ (a, b)
a=b	(a).operator= (b)	cannot be non-member
a(b...)	(a).operator()(b...)	cannot be non-member
a[b...]	(a).operator[](b...)	cannot be non-member
a->	(a).operator-> ( )	cannot be non-member
a@	(a).operator@ (0)	operator@ (a, 0)

См. пример `19_operator_overloading`.

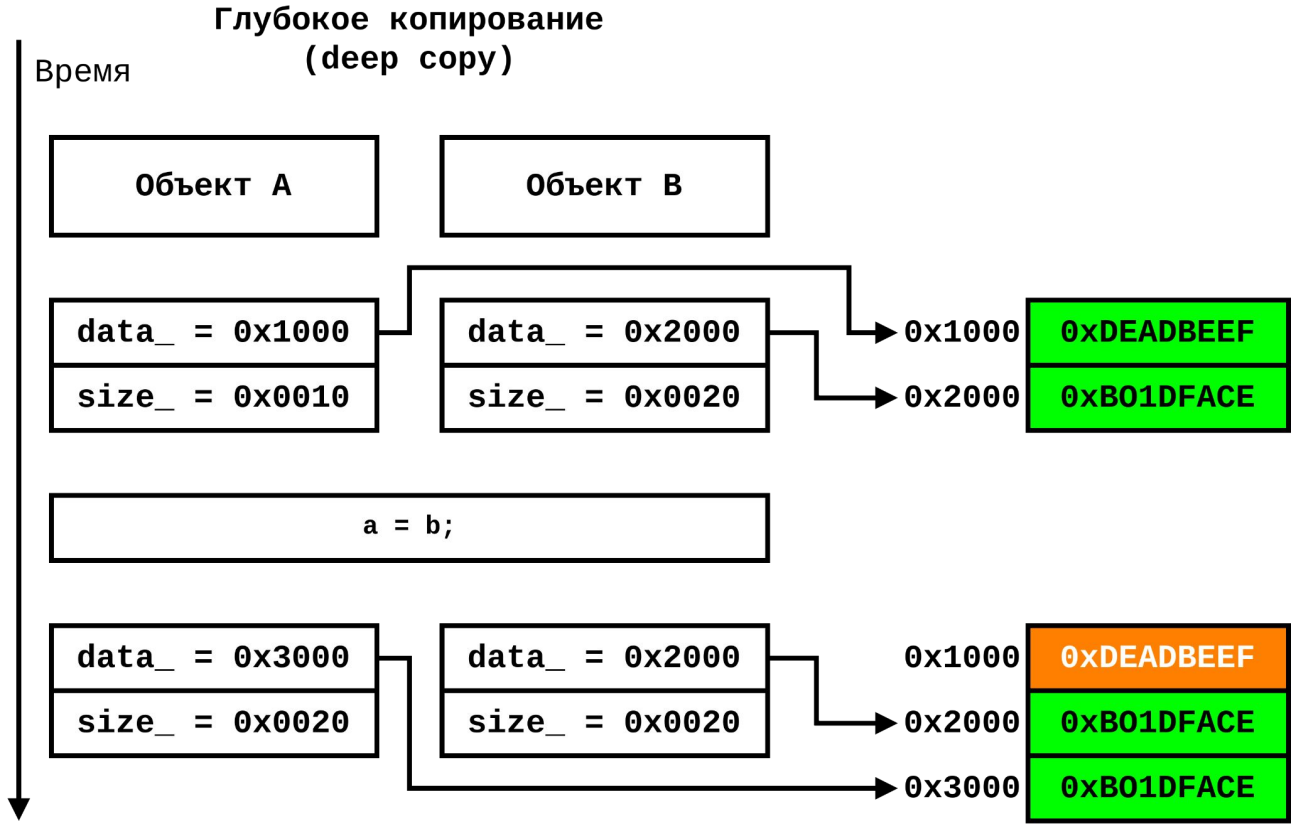
Операция присваивания, семантики копирования



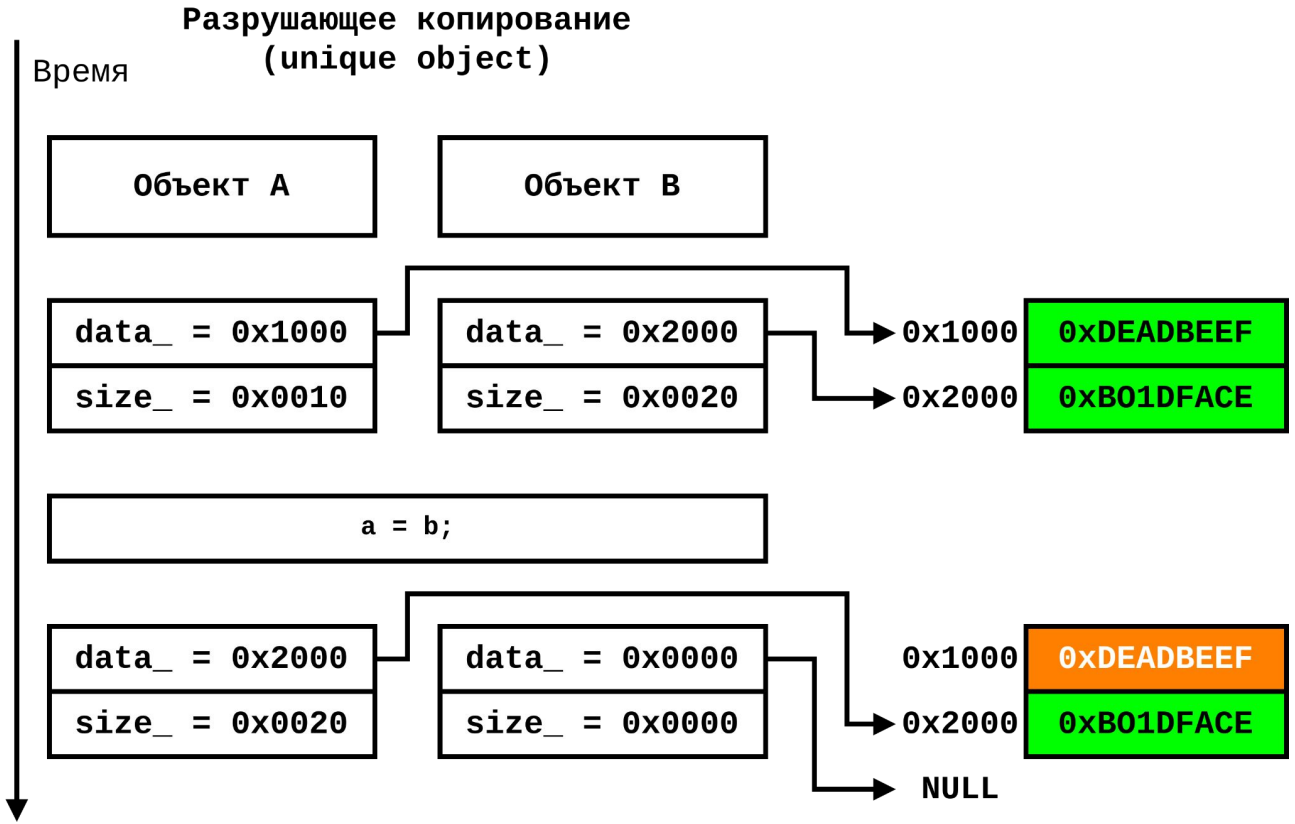
Поверхностное копирование



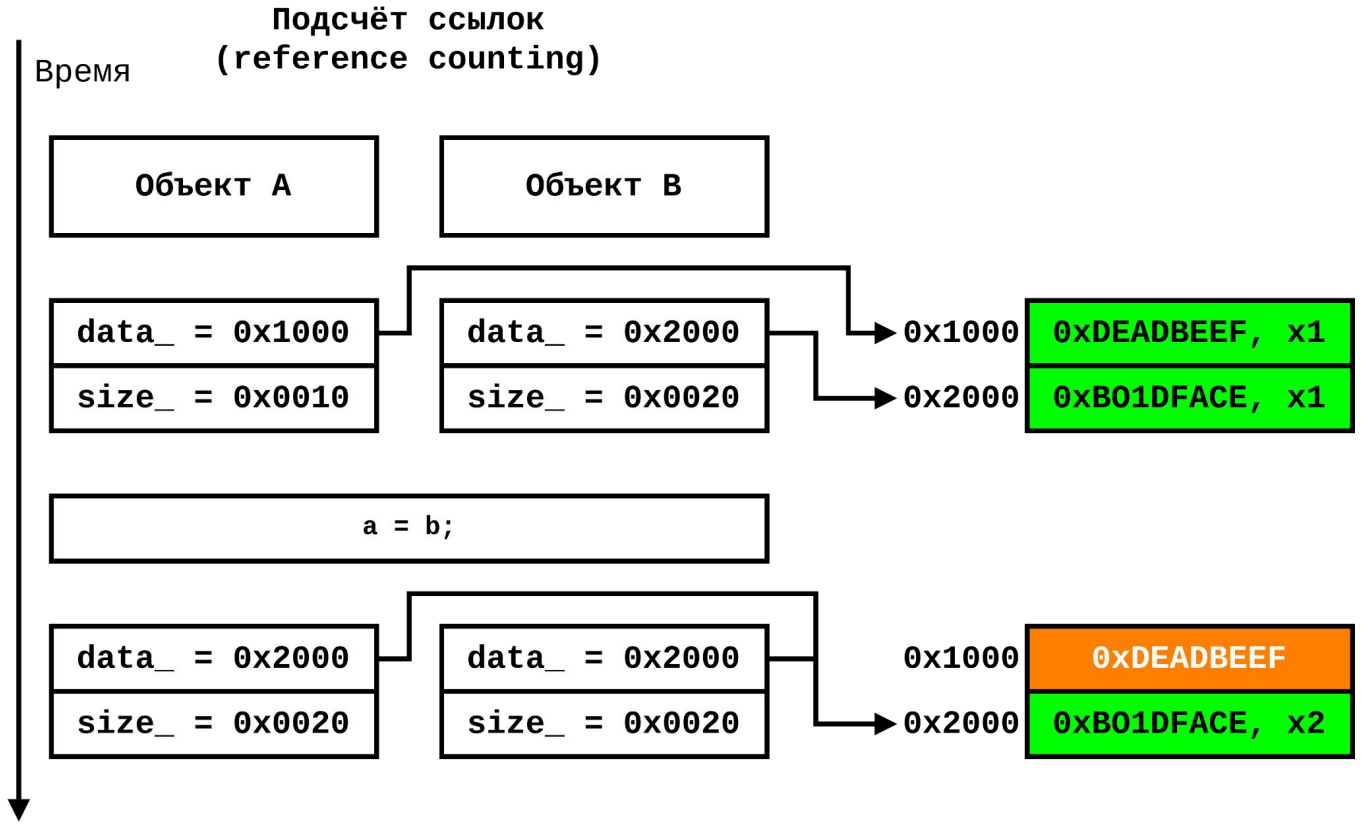
Глубокое копирование



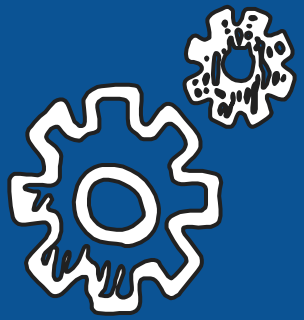
Уникальный объект



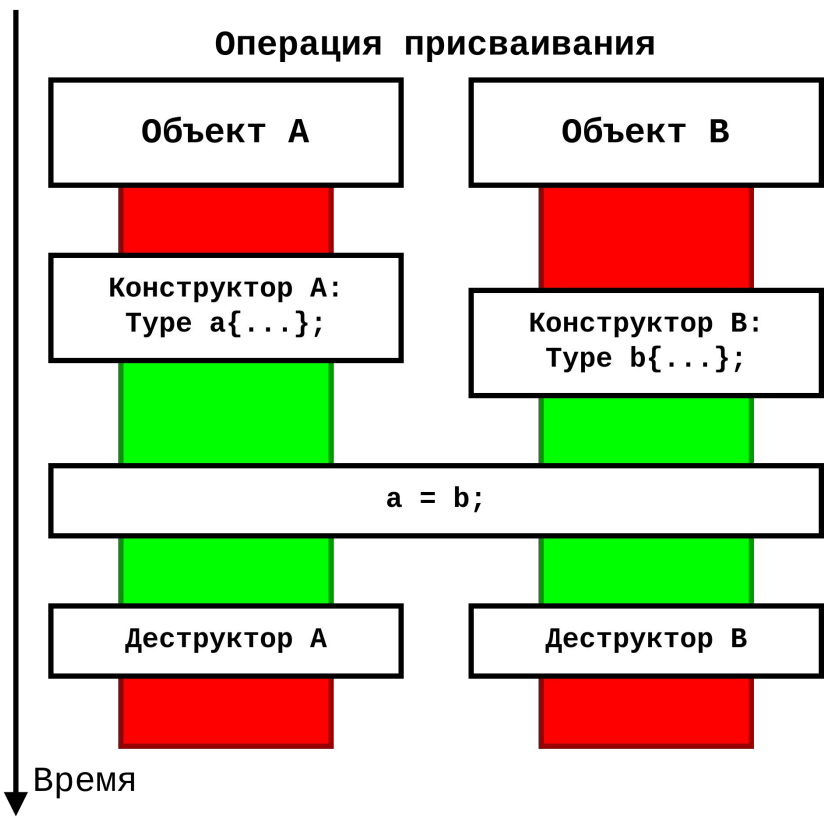
Подсчёт ссылок

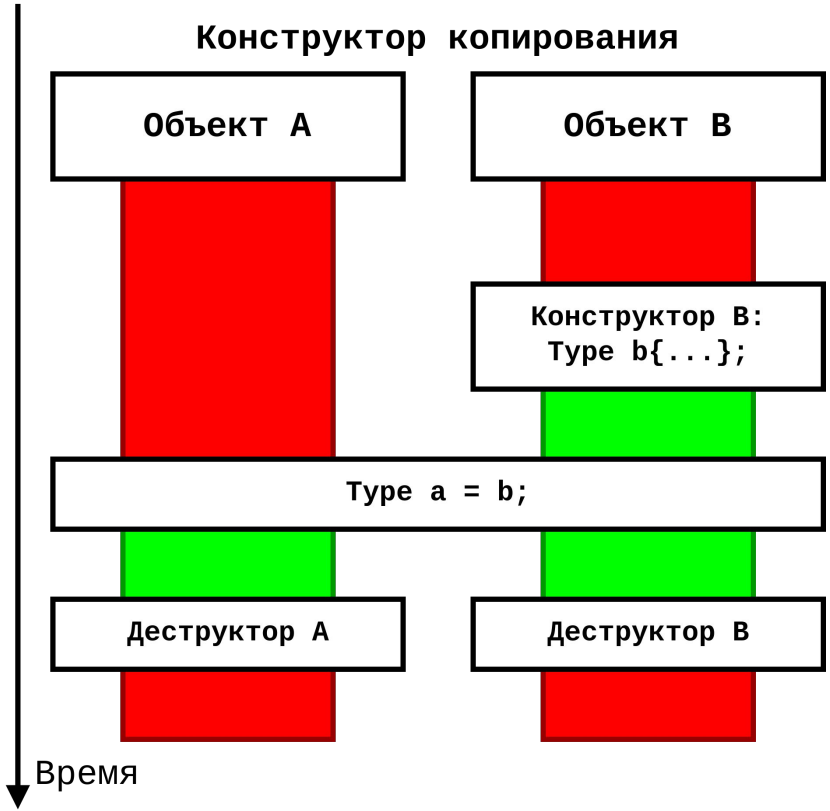


Конструктор копирования vs операция присваивания



Конструктор копирования и operator=





Наблюдение за «Автоматикой» C++

Внутренняя реализация операции +

```
// Производим добавление вектора к самому себе.
Vector copy = vec + vec;
```

```
Vector VectorArithmetics::operator+(const Vector& first, const Vector& second)
{
    Vector copy{first};

    copy += second;

    return copy;
}
```

Уровень оптимизаций по умолчанию:

```
int32_t main(int32_t argc, char** argv, char** envp)

00002c50    VectorArithmetics::Vector::operator+=(&var_2b8, &var_2a8);
00002cd6    int64_t rbx_1 = 0;
00002d1d    int32_t rbx_6;
```

Внутренняя реализация операции +

Выключенные оптимизации (-O0):

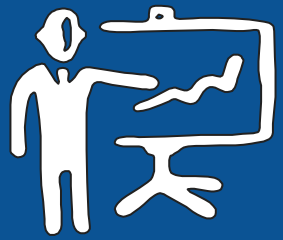
```
int64_t* VectorArithmetics::operator+(int64_t* arg1, int64_t* arg2, int64_t* arg3)

{
    VectorArithmetics::Vector::Vector(arg1, arg2);
    VectorArithmetics::Vector::operator+=(arg1, arg3);
    return arg1;
}
```

```
int32_t main(int32_t argc, char** argv, char** envp)

0000271a    void var_78;
0000271a    VectorArithmetics::operator+(&var_78, &var_68, &var_68);
0000271f    int64_t var_90 = 0;
000027dc    int32_t rbx;
```

Вопросы?



Красивые иконки взяты с сайта handdrawngoods.com