

C++

Лекции

2024–2025

2024-09-13

Введение	2
Классы	2

2024-09-13

Введение

Препоод: Сергей Александрович

Оценка:

$$\text{Итог} = 0.5 \cdot (0.1 \cdot A + 0.2 \cdot \text{Дз1} + 0.35 \cdot \text{Дз2} + 0.35 \cdot \text{Дз3}) + 0.5 \cdot (0.3 \cdot \text{Кр} + 0.7 \cdot \text{Экз})$$

Будет:

- ООП
- Параллельное и конкурентное программирование
- Функциональное программирование
- Всякое

Классы

Классы — исторически первое отличие C++ от C

```
class Matrix {
    private:
        size_t n_rows_;
        size_t n_cols_;
        double *data_;

    public:
        Matrix(size_t n_rows, size_t n_cols);
        Matrix(const Matrix& other);
        Matrix() = delete; // Явно удаляем default-ный конструктор,
                           // хотя он, и так, не создается

        int rank() const;

        size_t n_rows() const { return n_rows_; }
}

int main() {
    Matrix m(10, 10);

    m.rank();
}
```

«Программы надо писать для людей»

Методы, реализованные внутри объявления класса, часто становятся inline-овыми.

Инкапсуляция — скрытие внутреннего состояния (private в классах). Позволяет:

1. меньше косячить в программах
2. отделять реализацию от интерфейса

Листинг 1. Инкапсуляция в Си

```

public.h:

typedef void* Matrix;
Matrix matrix_create();
int matrix_rank(Matrix m);

public.h:

Matrix matrix_create() { ... }
int matrix_rank(Matrix m) {
    struct MatrixData* = (MatrixData*)m;
    ...
}

```

const — после называния метода, значит метод не меняет экземпляр

```

Matrix m; // default-ный конструктор
Matrix m(1, 1); // конструктор
Matrix m(); // объявление функции
Matrix m{}; // default-ный конструктор
Matrix m2 = m; // конструктор копирования

```

Удалять, как создавали:

```

Matrix* pm = new Matrix(1, 1);
Matrix* a = new Matrix[100];

delete pm;
delete[] a;

```

New по уже выделенной памяти:

```

void* addr = malloc(...);
new (addr) Matrix(1, 1);
a->~Matrix();

```

Если у полей нет default-ного конструктора или поля константы или ссылки, то делать так:

```

class X {...};

X::X(int y) : a(y) { ... }

```

Поля инициализируются в том порядке, в котором указаны в классе

X&& — r-value

```

X::X(X&& other) {...}

```

Нельзя перегрузить оператор внутри класса, если первый аргумент другого типа

Правило трех:

- TODO
- TODO
- TODO

Правило пяти:

- .. правило трех
- TODO
- TODO

Не стоит бросать exception в деструкторе тк exception во время обработки exception-a — плохо

exception в конструкторе — можно

Хорошо делать exception только с типами, унаследованными от std::exception