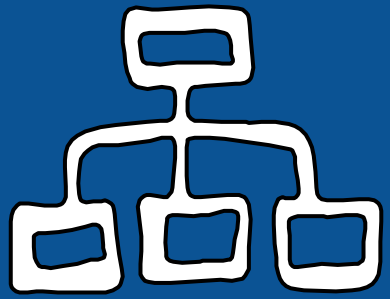
Алгоритмы и Алгоритмические Языки

Семинар #18:

1. Система сборки Make и шаблон Makefile.
2. Модификаторы доступа `public` и `private`.
3. Автоматическое управление временем жизни: конструктор и деструктор.
4. Регистрация в контесте «Активность».

21.01.2025

Система сборки Make. Шаблон Makefile.

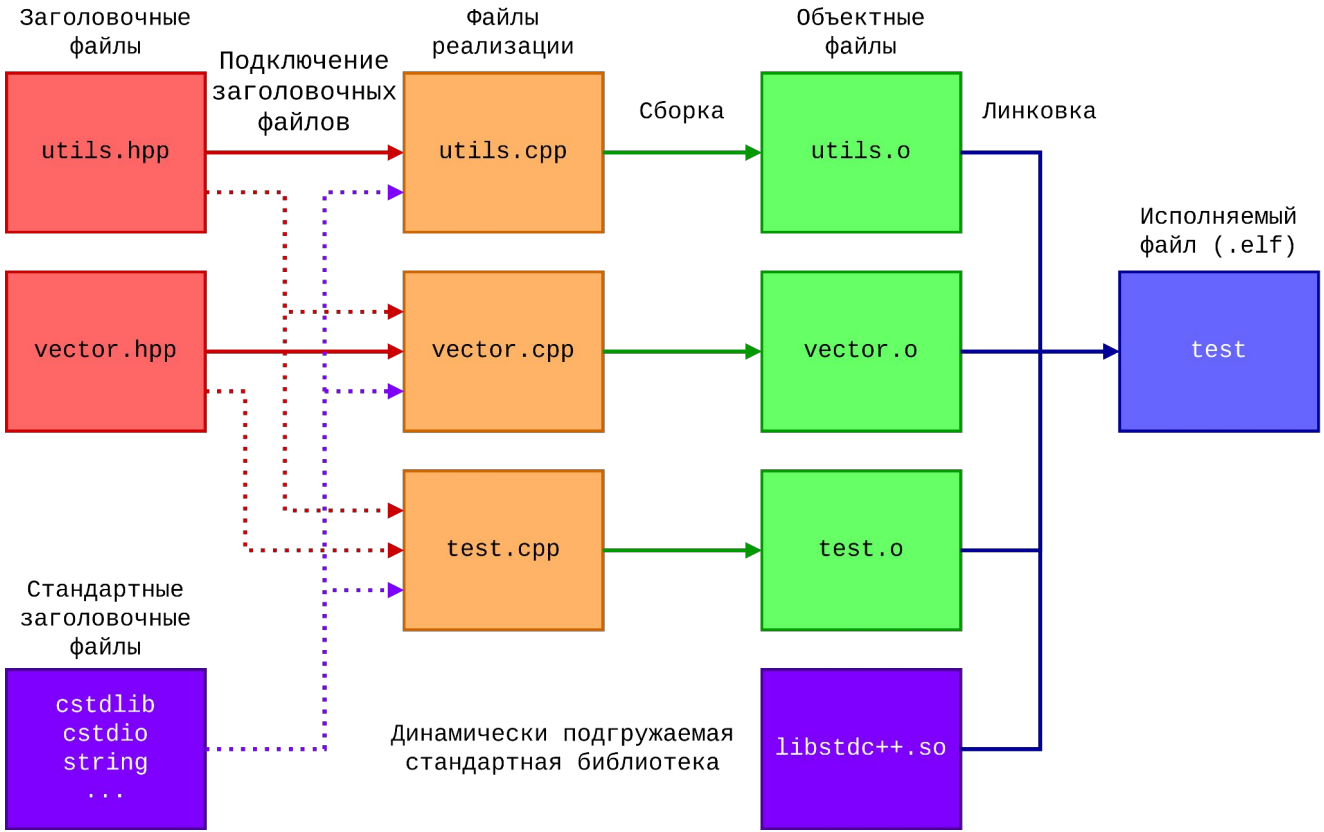


Более простой пример на C++

```
❯ cd examples/18_classes
❯ tree .
.
├── build
│   ├── test
│   ├── test.o
│   ├── utils.o
│   └── vector.o
├── include
│   ├── utils.hpp
│   └── vector.hpp
...
```

```
❯ cd examples/18_classes
❯ tree .
...
├── Makefile
└── src
    ├── test.cpp
    ├── utils.cpp
    └── vector.cpp
```

Модель сборки проекта



Сборка проекта вручную

```
❯ make
Building object file build/test.o
g++ -c src/test.cpp -std=c++17 -Wall -Wextra -Werror -I include -o
build/test.o
Building object file build/utils.o
g++ -c src/utils.cpp <CXXFLAGS> -I include -o build/utils.o
Building object file build/vector.o
g++ -c src/vector.cpp <CXXFLAGS> -I include -o build/vector.o
Linking executable build/test
g++ build/test.o build/utils.o build/vector.o <LDFLAGS> -o build/test
```

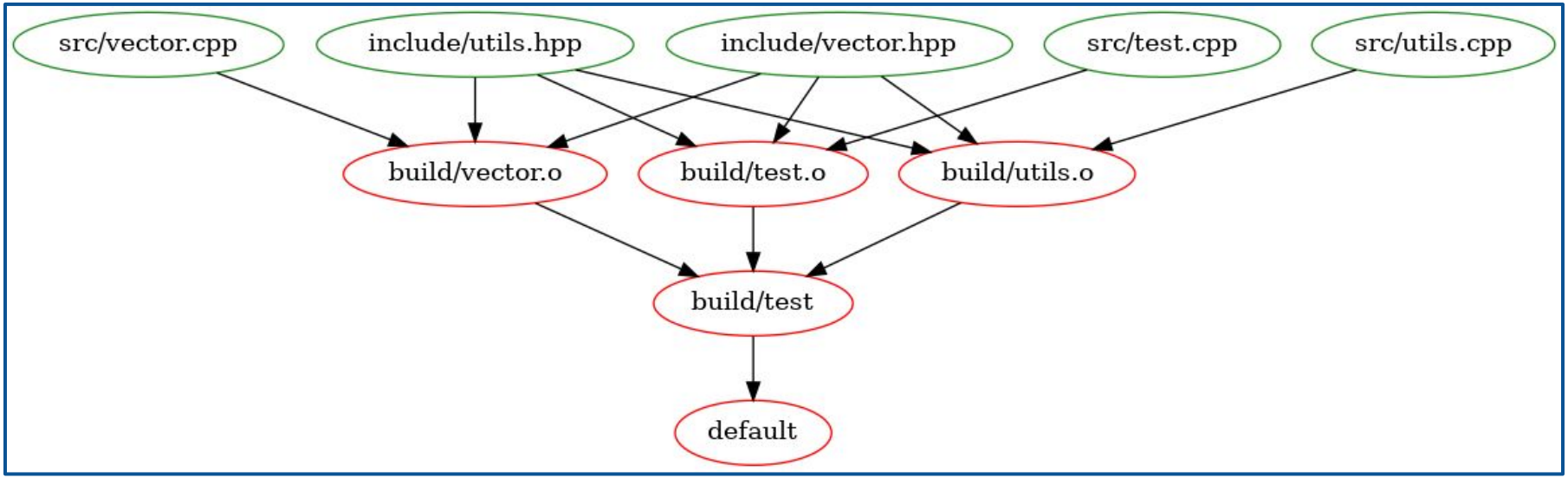
Система сборки Make

```
build/hello: hello.c
    @mkdir -p build
    gcc hello.c -Wall -Werror -o build/hello

run: build/hello
    @./build/hello

clean:
    @rm -rf build

.PHONY: run clean
```

Шаблон Makefile для проекта на C++

См. шаблон Makefile.

Граф зависимостей между целями в системе сборки:



Модификаторы доступа private и public.

Модификаторы доступа public и private

Модификаторы доступа:

- **`private`** – поля и методы доступны только другим методам.
- **`public`** – поля и методы доступны всем.

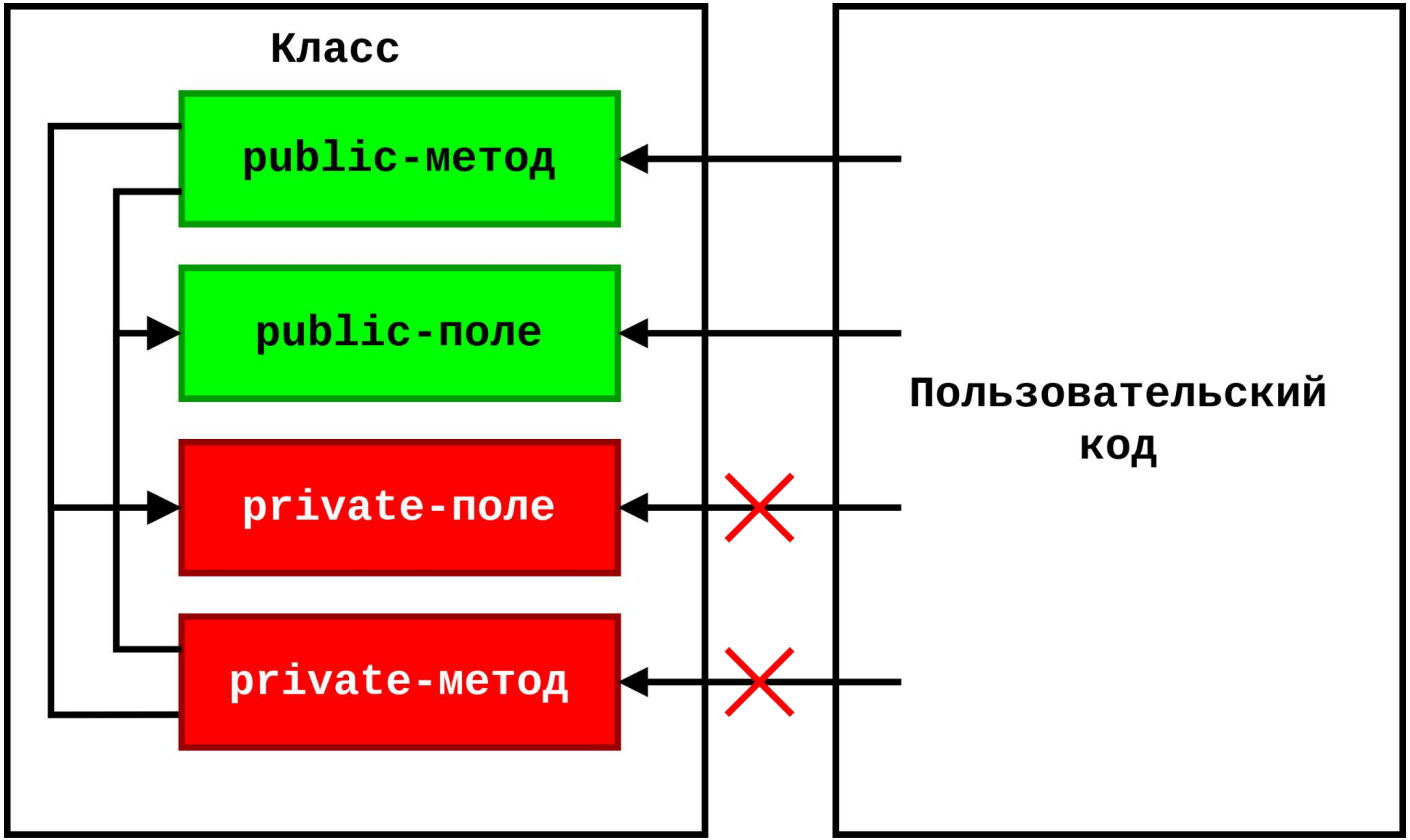
Отличие **`struct`** от **`class`**:

- По умолчанию для **`struct`** – всё **`public`**.
- По умолчанию для **`class`** – всё **`private`**.

Цели применения модификаторов доступа:

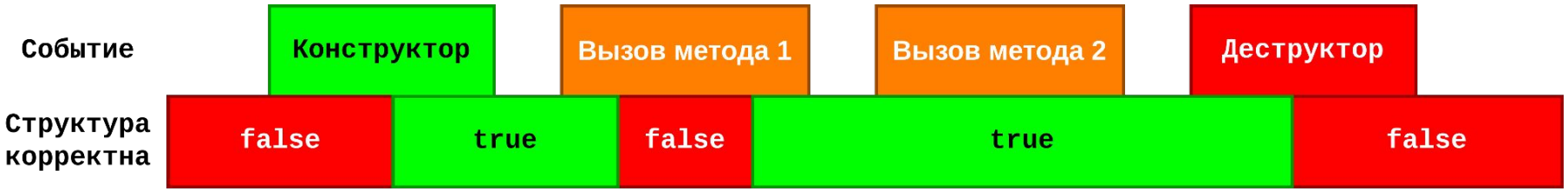
- **Инкапсуляция**, сокрытие сложности, простота использования.
- **Модульность**: можно изменить реализацию, не меняя интерфейс – код, использующий класс не изменится*.

Модификаторы доступа public и private



Автоматическое управление временем жизни: конструктор и деструктор

Конструкторы

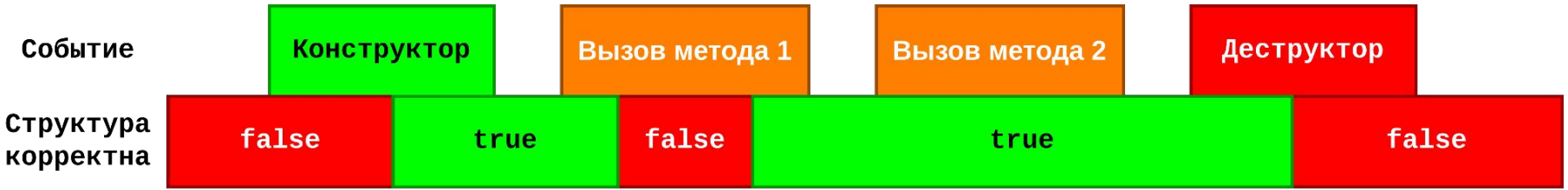


Конструкторы:

- Один из конструкторов вызывается при создании объекта.
- Задача – инициализация структуры корректным определённым значением.
- Конструктор по умолчанию – ничего не делает.

При аллокации дин.памяти операторы **`T::new[]`** и **`T::new`** вызывают конструктор по умолчанию.

Деструктор



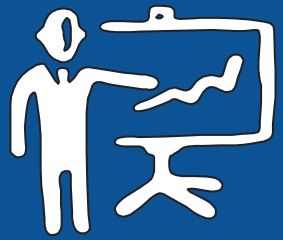
Деструктор:

- Деструктор вызывается по завершении времени жизни объекта.
- Задача – освобождение внешних выделенных ресурсов.
- Деструктор по умолчанию – ничего не делает.

При аллокации дин.памяти операторы **`T::delete[]`** и **`T::delete`** вызывают конструктор по умолчанию.

Регистрация в контесте «Активность»

Вопросы?



Красивые иконки взяты с сайта handdrawngoods.com