Сибирь II

Классы

Проверка остаточных знаний

Что такое структуры?

Если помните, классы мало чем отличаются, ну а если не помните...

Классы

Это производные типы, включающие в себя множество элементов различных типов данных и методы их обработки

Задача

Написать игру “Сапёр”

Что мы должны “помнить” во время игры и что мы должны уметь делать?

Из данных:

- Поле (открытые клетки, флажки и мины)
- Статус игры
- Время, прошедшее с момента начала

Из методов:

- Рендеринг поля
- Постановка/удаление флажка
- Открытие клетки

Немножко духоты

Спецификация класса состоит из:

- Объявления класса (для удобства изучения возможностей класса в заголовочном файле)
- Определения методов класса

Инкапсуляция

Это принцип, по которому внутреннее устройство сущностей нужно объединять в специальной оболочке и скрывать от вмешательства извне.

Спецификаторы доступа

- public
- private
- protected

```cpp
class order {
 public:  // доступны вне класса
  std::string category;
  std::string product_name;
  unsigned int cost;

 private:  // доступны только внутри класса
  std::string customer;
};

int function() {
  order example = {...};
  std::cout << example.cost << std::endl;  // можно
  std::cout << example.customer << std::endl;  // нельзя
}
```

Конструкторы

1. По умолчанию
2. Копирования
3. Присваивания (пока не нужен)
4. Пользовательские (для собственного удобства)

Деструктор

Нужен, чтобы убирать за программистом

Перегрузка операторов

Синтаксический сахар

```
string& operator=(const string& oth);
```

`this`

Это указатель на объект класса

Правило трёх и правило пяти

Шаблоны

Помогают избавиться от дублирования кода для разных нужд

```
template <T>
T max(T a, T b);
```

Инстанцирование шаблона – генерация кода шаблона для конкретных параметров

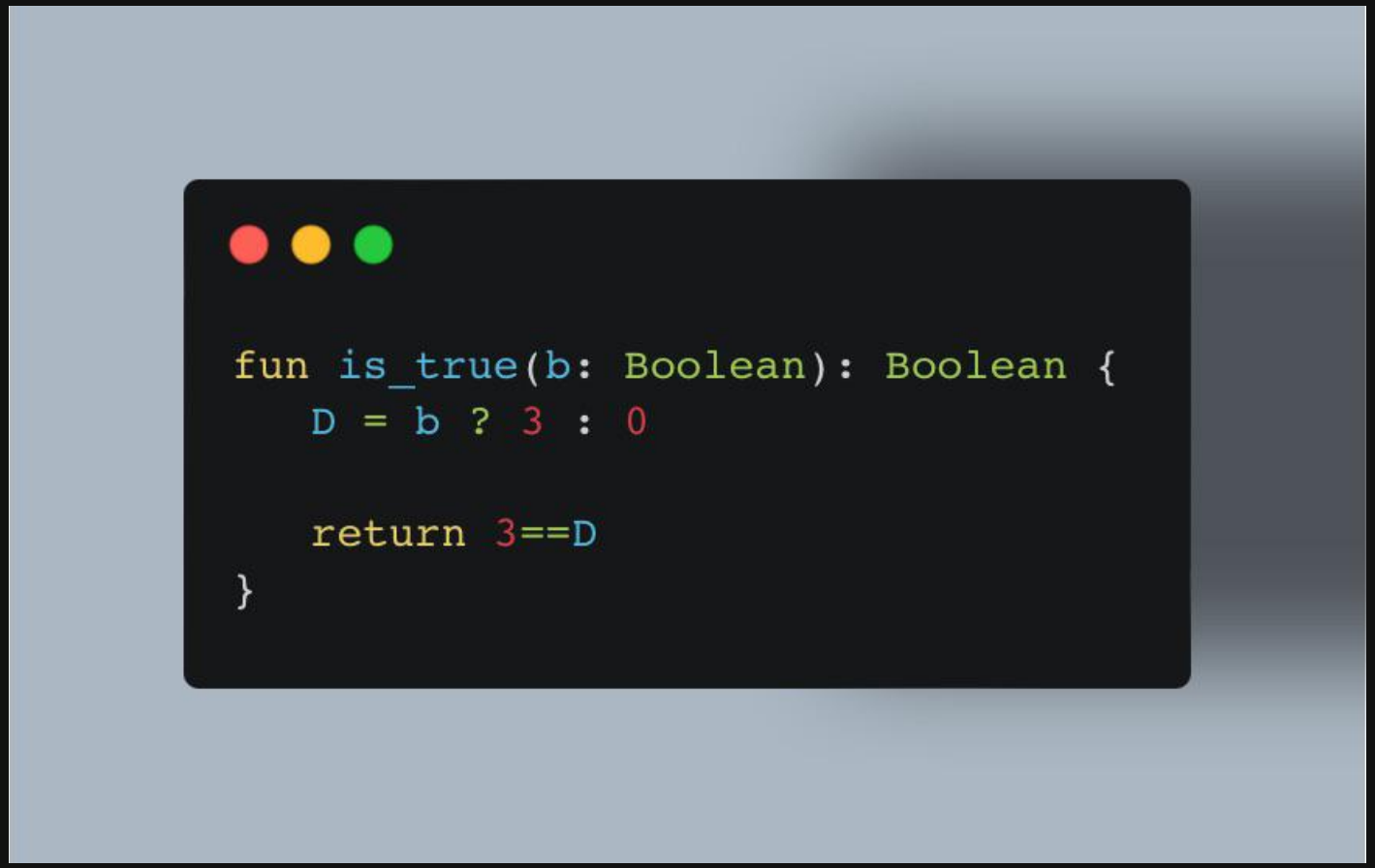
Иногда для шаблона стоит явно указывать тип

Специализация шаблонов:

```
template <>
const char* min(const char* a, const char* b) {
  return (strcmp(a, b) < 0) ? a : b;
}
```

Концепты

Концепты добавляют семантические требования к параметрам шаблона


fun is_true(b: Boolean): Boolean {
D = b ? 3 : 0
return 3==D
}

Пример

```
#include <concepts>
#include <iostream>

template<typename T>
requires std::integral<T>
auto gcd(T a, T b) {
  if( b == 0 ) return a;
  else return gcd(b, a % b);
}

template<typename T>
auto gcd1(T a, T b) requires std::integral<T> {
  if( b == 0 ) return a;
  else return gcd1(b, a % b);
}
```

```cpp
template<std::integral T>
auto gcd2(T a, T b) {
  if( b == 0 ) return a;
  else return gcd2(b, a % b);
}

auto gcd3(std::integral auto a, std::integral auto b) {
  if( b == 0 ) return a;
  else return gcd3(b, a % b);
}
```

Ключевое слово `requires` задает требования к шаблону на время компиляции