Программирование на C++

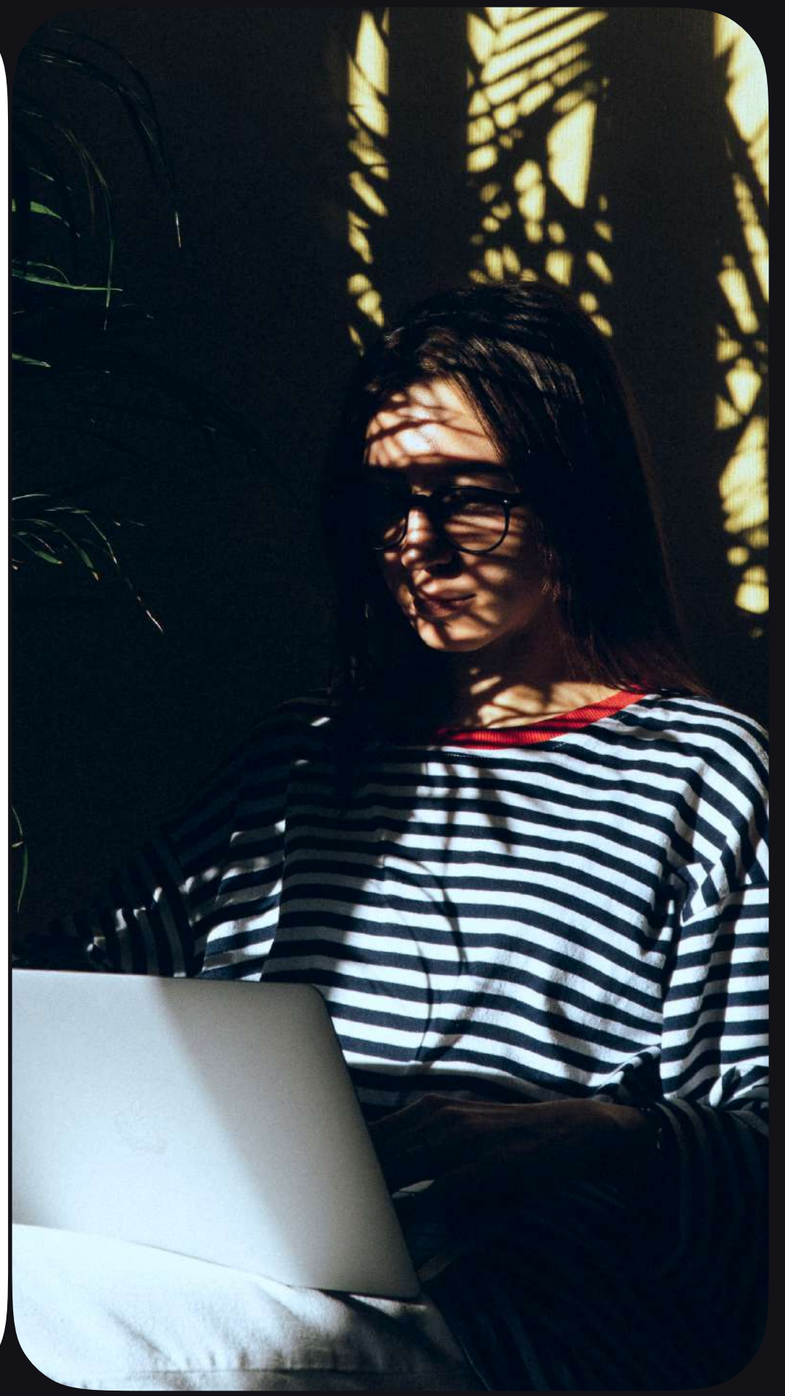
Минцифры России

UCHi DOMA

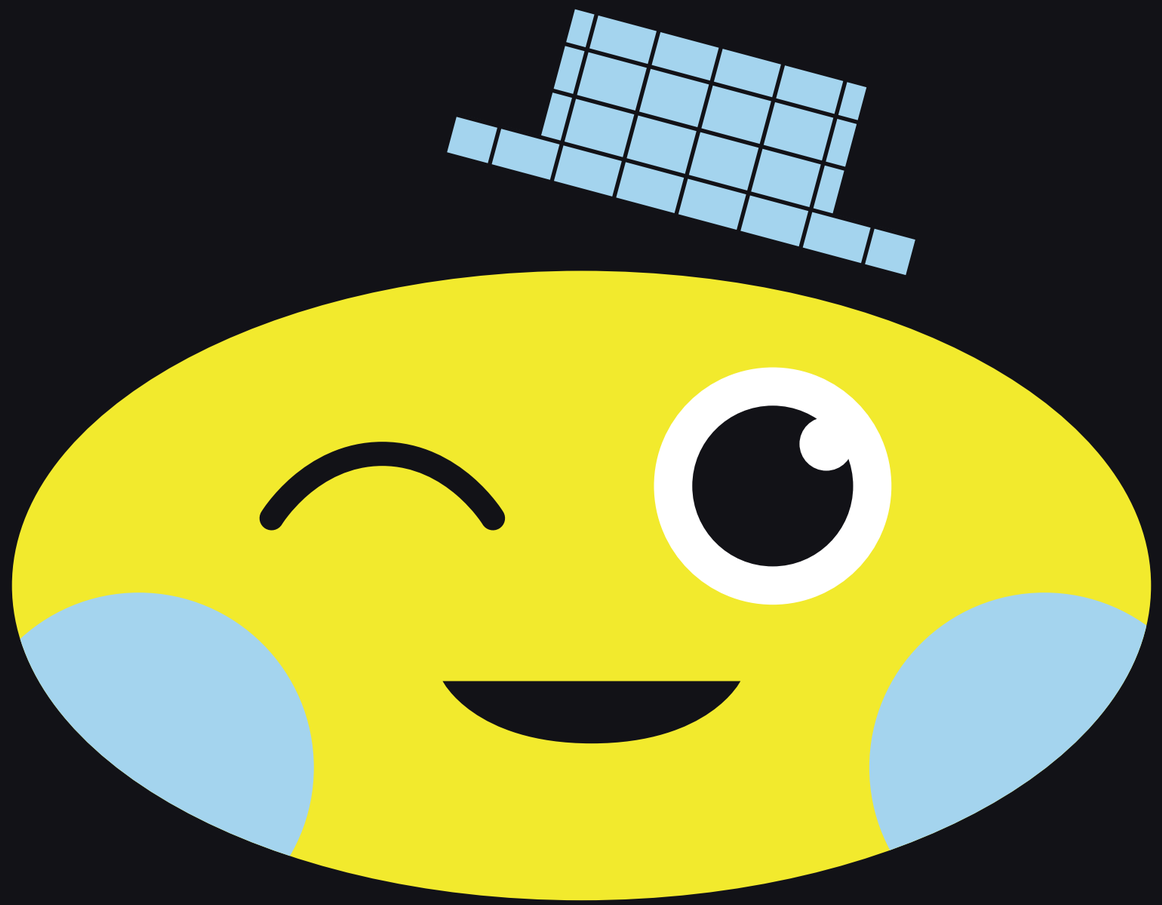
20.35 УНИВЕРСИТЕТ

Модуль 3. Урок 4

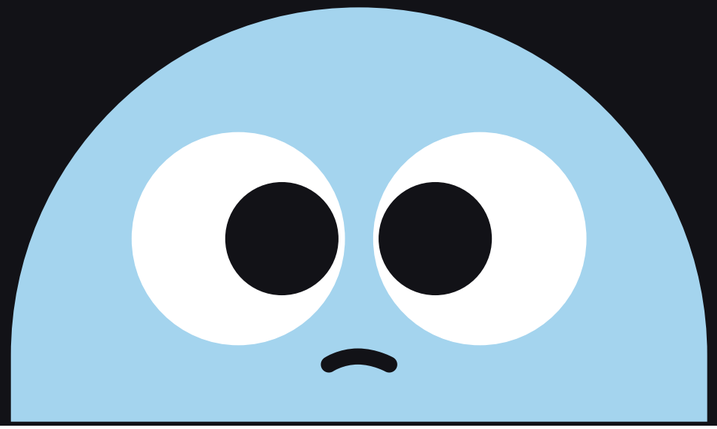
Классы



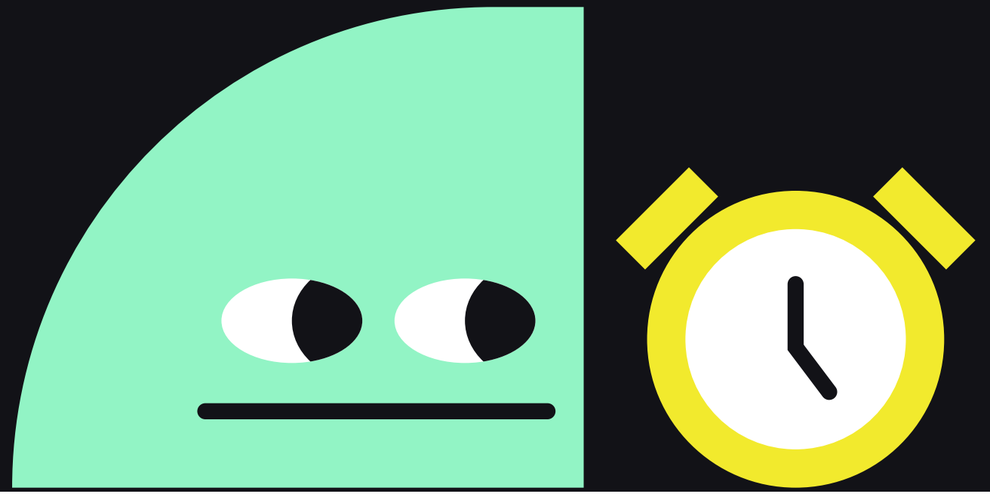
Привет!



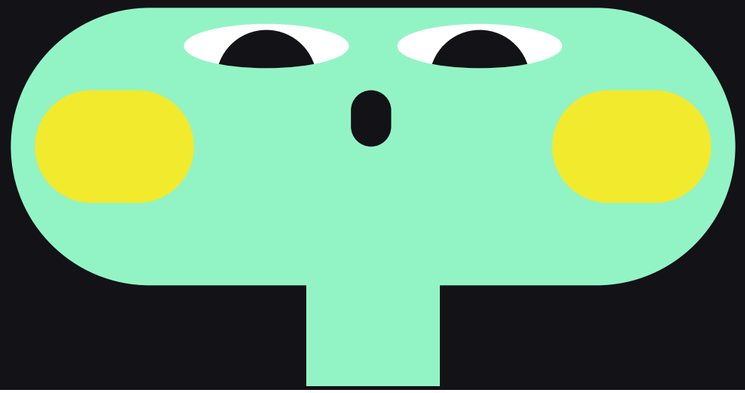
Проверка готовности



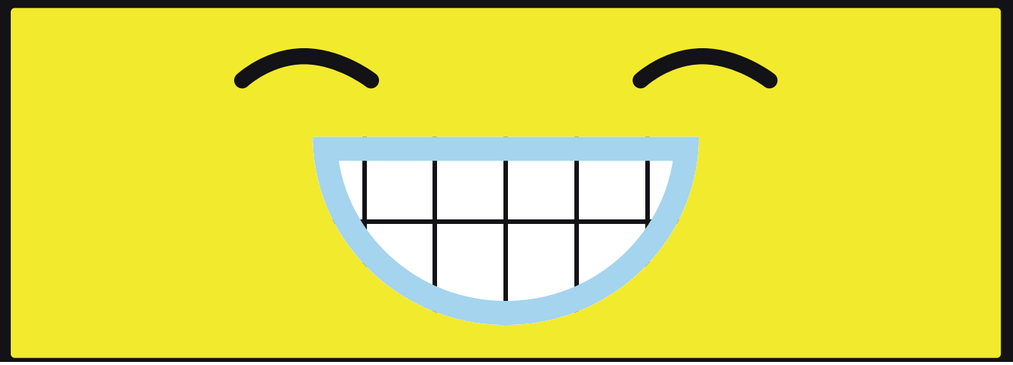
Видим и слышим друг друга без помех



Не опаздываем и не отвлекаемся



Сидим прямо



Улыбаемся, если всё ок

Как домашка?

- Какие были трудности?
- Какие остались вопросы?
- Сколько заданий выполнено?

Разомнёмся

Какой тип доступа устанавливается для данных внутри структуры для обеспечения инкапсуляции?

1. **public** — общие
2. **private** — частные
3. **protected** — защищенные

Разомнёмся

Стандартным является размещение член-данных в частной области (**private**).

Разомнёмся

Какой тип доступа устанавливается для функций (интерфейса) внутри структуры для обеспечения инкапсуляции?

1. **public** — общие
2. **private** — частные
3. **protected** — защищенные

Разомнёмся

Стандартным является размещение части функций-членов — в общей части (public) абстрактного типа данных.

Цели урока

- изучить классы
- отработать на практике написание алгоритмов с использованием классов на C++

Класс в ООП

Класс — это способ описания сущности, определяющий состояние и поведение, зависящее от этого состояния, а также правила для взаимодействия с данной сущностью (контракт).

С точки зрения программирования класс можно рассматривать как набор данных (полей, атрибутов, членов класса) и функций для работы с ними (методов).

С точки зрения структуры программы, класс является сложным типом данных.

Класс в C++

Класс в C++ — это определенная пользователем структура данных, объявленная с помощью ключевого слова class, которая имеет данные и функции (также называемые переменными-членами и функциями-членами) в качестве своих членов, доступ к которым регулируется тремя спецификаторами доступа private, protected или public.

Класс в C++

Классы в C++ определяются ключевым словом **class**.
Они представляют собой форму структуры, у которой спецификация доступа по умолчанию — **private**, то есть

`class s { ...};` это сокращенная запись `struct s {private: ...};`

Класс в C++

Понятие класса является Базовым в языке C++.

В C++ принято считать, что структура struct — это класс, все члены которого общие, то есть

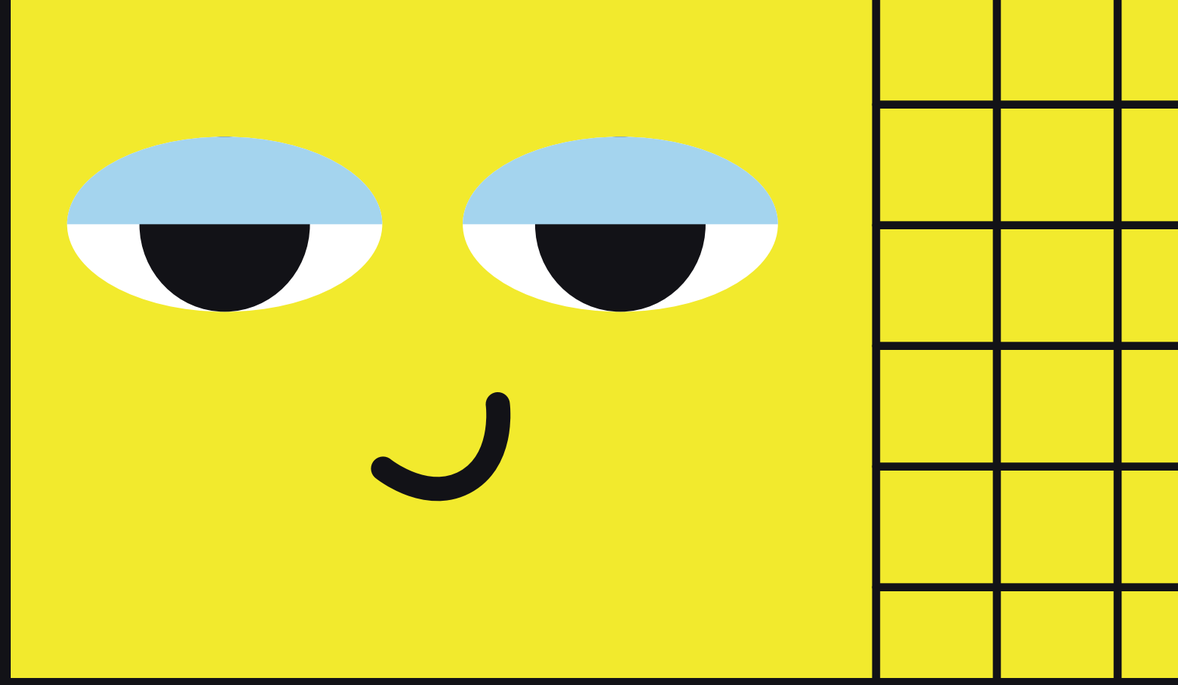
`struct s {...};` это сокращенная запись `class s {public: ...};`

Классы и структуры

Структуры необходимо использовать
в тех случаях, когда сокрытие данных неуместно.

Чаще всего структуры содержат только поля
и не содержат методов.

Если предполагается создание объектов, для которых
будут реализованы методы их использования,
то целесообразно использовать класс.

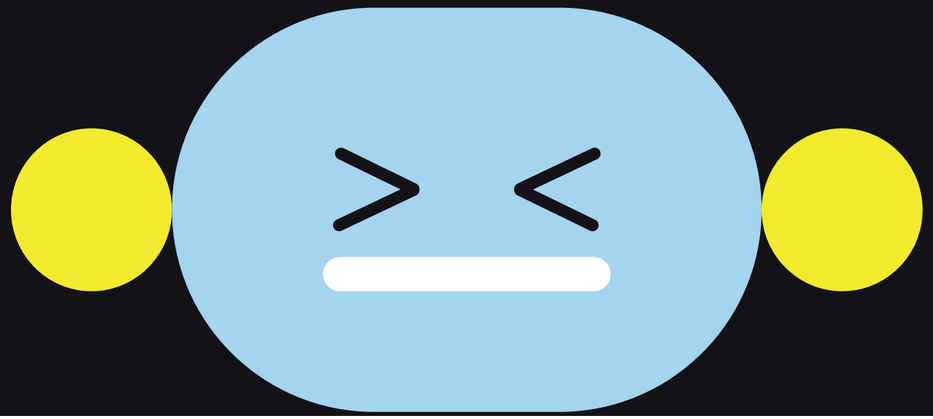


Практика

Перерыв физкультминутка

Смотрим вверх–вниз, вправо–влево

Вращаем по кругу туда–обратно



Крепко зажмуриваемся

Быстро моргаем

Статические члены класса

Класс — это тип, а не объект данных, и в каждом объекте класса имеется своя собственная копия данных — членов этого класса.

Некоторые типы требуется реализовать так, что все объекты этого типа могут совместно использовать некоторые данные. Такие совместно используемые данные должны быть описаны как часть класса.

Статические члены класса

Статические данные относятся ко всем объектам класса.
Такие данные используются, если:

1. требуется контроль общего количества объектов класса
2. требуется одновременный доступ ко всем объектам или части их
3. требуется разделение объектами общих ресурсов.

В этом случае в определение класса
могут быть введены статические члены.

Статические члены класса

Статические члены описываются с помощью ключевого слова static, которое может использоваться при объявлении член-данных и член-функций.

Такие члены классов называются статическими, и независимо от количества объектов данного класса, существует только одна копия статического элемента.

Статические члены класса

Обращение к статическому элементу осуществляется
с помощью оператора разрешения контекста и имени класса:

ИмяКласса :: ИмяЭлемента

Если **x** — статическое член-данное класса cl,
то к нему можно обращаться как

cl::**x**

При этом не имеет значения количество
объектов класса cl.

Пример статической член-функции

```
#include <iostream>
using namespace std;
class cl
{
public:
  static int f_st(int a) { return 2 * a; };
  int f(int a) { return 2 * a; };
};
int main()
{
  cl obj;
  cout << obj.f(10) << endl;  // для любой функции
  cout <<cl::f_st(10) << endl;  // только для статической функции
  return 0;
}
```

Статические члены класса

Статические член-данные (или поля) класса можно рассматривать как глобальную переменную класса. Но в отличие от обычных глобальных переменных на статические члены распространяются правила видимости **private** и **public**. Поместив статическую переменную в часть **private**, можно ограничить ее использование.

```
class X
{
  static int n;
};
```

Инициализация статических полей

Статические поля нельзя инициализировать в теле класса, а также в методах. Статические поля должны инициализироваться аналогично глобальным переменным в области видимости файла:

```
#include <iostream>
using namespace std;
class X
{
  static int n;
public:
  static int getN() { return n; };
  static int setN(int value) { n = value; }
};
int X::n = 0; // инициализация скрытого статического поля
int main()
{
  cout << X::getN() << endl;
  X::setN(10);
  cout << X::getN() << endl;
  //cout << X::n << endl; // ошибка - скрытое поле класса
  cin.get();
  return 0;
}
```

Закрепление

```
1   #include <iostream>
2   using namespace std;
3   class X
4   {
5     static int n;
6   public:
7     static int getN() { return n; };
8     static int setN(int value) { n = value; }
9   };
10  int X::n = 0;
11  int main()
12  {
13    cout << X::getN() << endl;
14    X::setN(10);
15    cout << X::getN() << endl;
16    cin.get();
17    return 0;
18  }
```

Какие поля класса объявлены как статические?

Закрепление

```
1   #include <iostream>
2   using namespace std;
3   class X
4   {
5     static int n;
6   public:
7     static int getN() { return n; };
8     static int setN(int value) { n = value; }
9   };
10  int X::n = 0;
11  int main()
12  {
13    cout << X::getN() << endl;
14    X::setN(10);
15    cout << X::getN() << endl;
16    cin.get();
17    return 0;
18  }
```

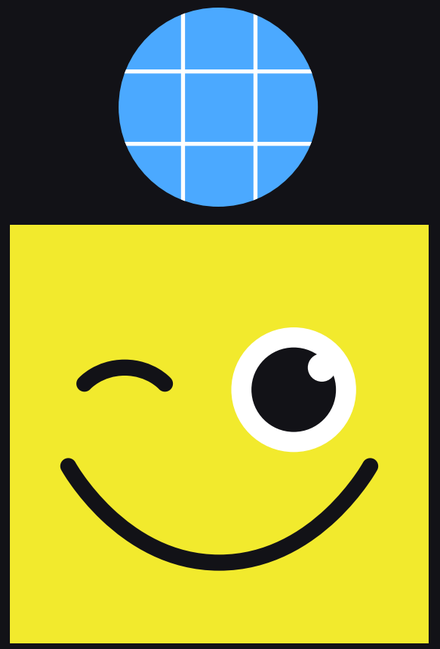
Статические поля класса:

n

Подведём итоги

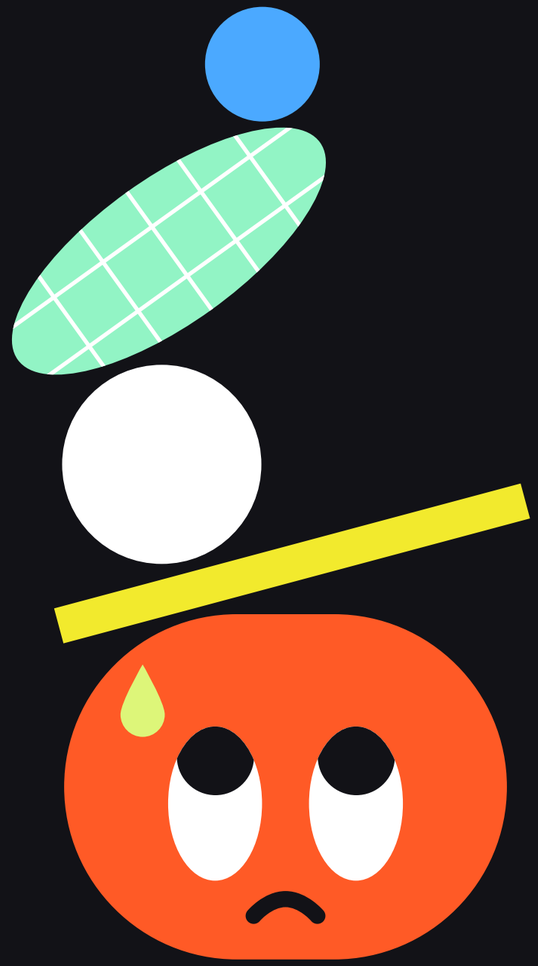
- изучили классы
- отработали на практике
написание алгоритмов
с использованием классов на C++

Итоги урока



1 легко!

2 сложно, но можно



3 трудно

Домашнее задание

До встречи!