Программирование на C++

Минцифры России

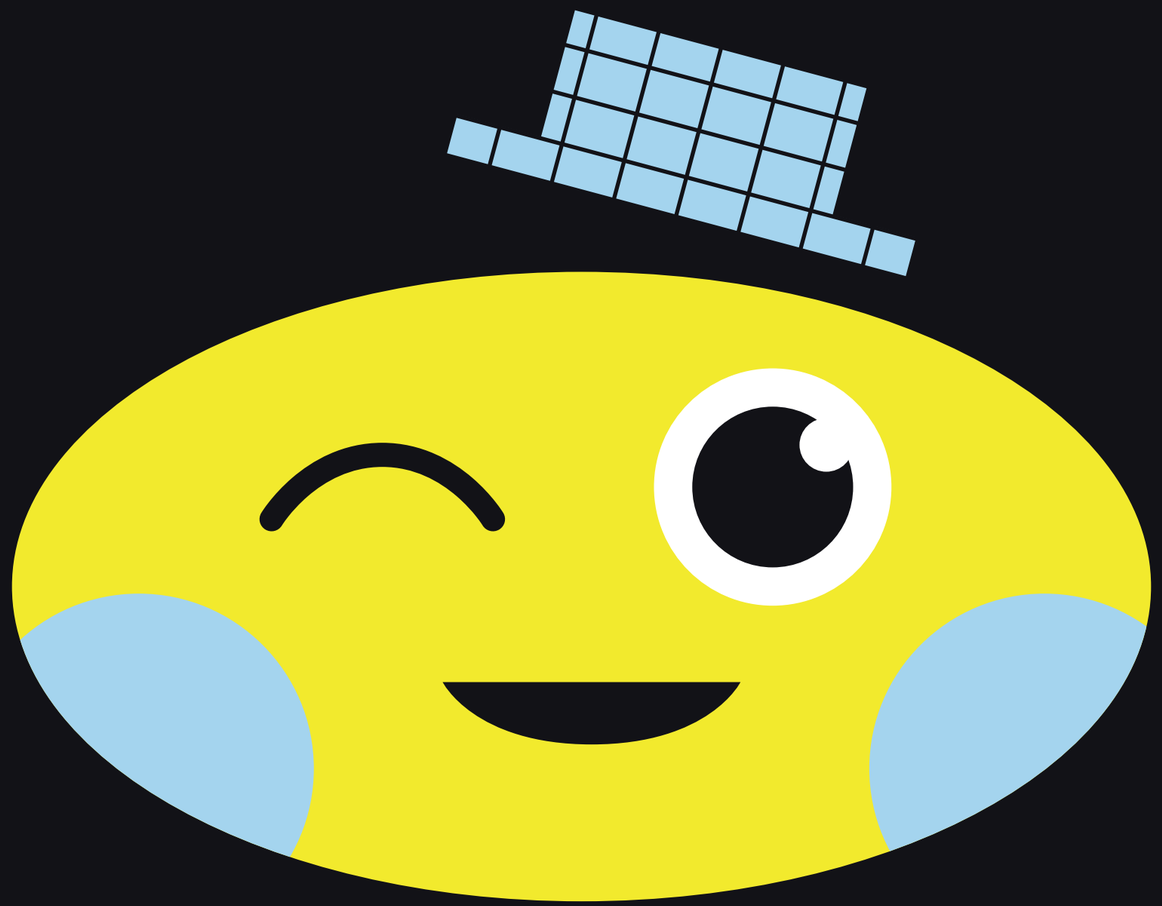
UCHi DOMA

20.35 УНИВЕРСИТЕТ

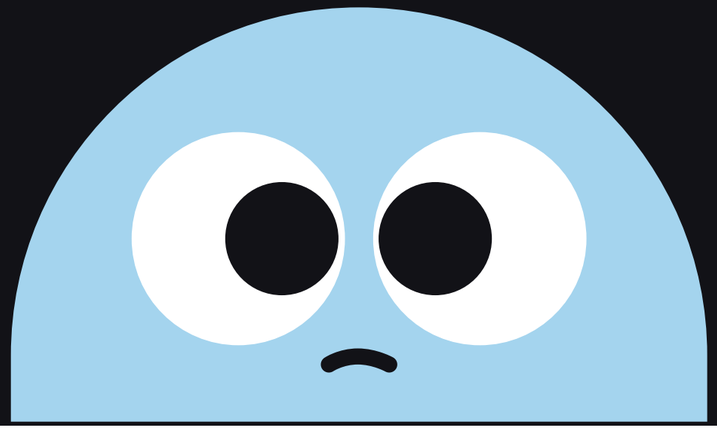
Модуль 3. Урок 5

Создание собственных классов

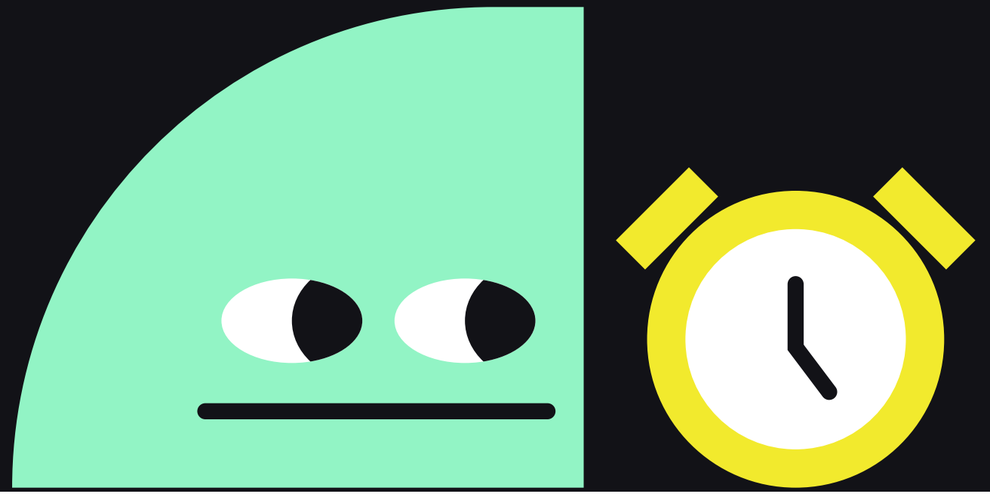
Привет!



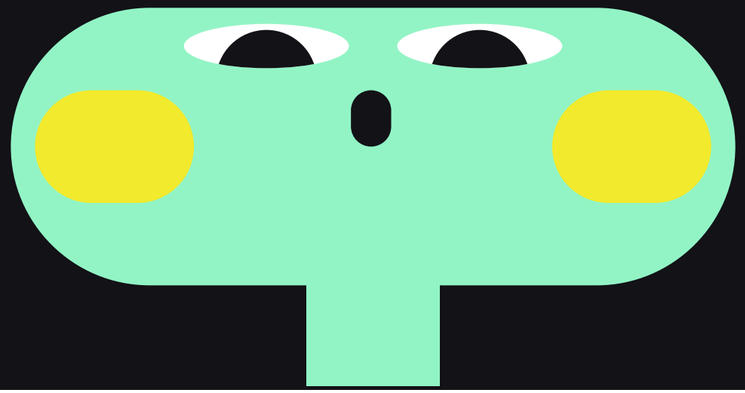
Проверка готовности



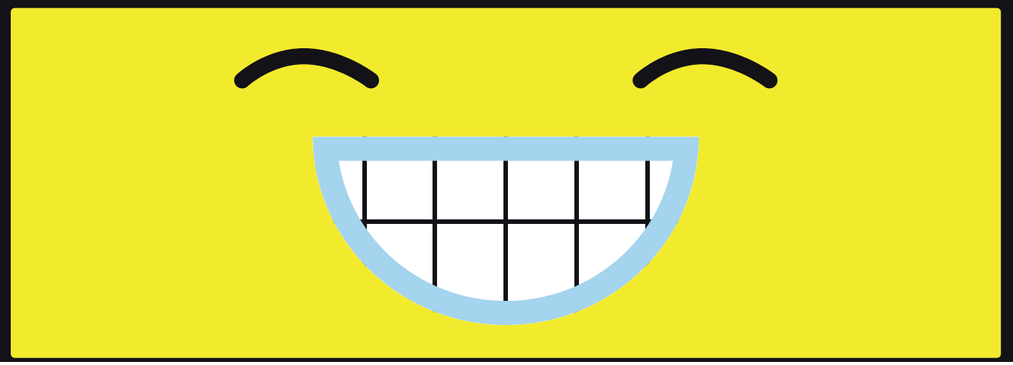
Видим и слышим друг друга без помех



Не опаздываем и не отвлекаемся



Сидим прямо



Улыбаемся, если всё ок

Как домашка?

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

Разомнёмся

Выберите программу с верным обращением к статическому полю класса

1

```cpp
#include <iostream>
using namespace std;
class A
{
  static int n;
public:
  static int getN() { return n; };
  static void setN(int value) { n = value; }
};
int A::n = 0;
int main()
{
  A::setN(5);
  cout << A::getN() << endl;
  return 0;
}
```

2

```cpp
#include <iostream>
using namespace std;
class A
{
  static int n;
public:
  static int getN() { return n; };
  static void setN(int value) { n = value; }
};
int A::n = 0;
int main()
{
  A::setN(5);
  cout << A::n << endl;
  return 0;
}
```

Разомнёмся

Выберите программу с верным обращением к статическому полю класса

1

```
#include <iostream>
using namespace std;
class A
{
  static int n;
public:
  static int getN() { return n; };
  static void setN(int value) { n = value; }
};
int A::n = 0;
int main()
{
  A::setN(5);
  cout << A::getN() << endl;
  return 0;
}
```

2

```
#include <iostream>
using namespace std;
class A
{
  static int n;
public:
  static int getN() { return n; };
  static void setN(int value) { n = value; }
};
int A::n = 0;
int main()
{
  A::setN(5);
  cout << A::n << endl;
  return 0;
}
```

Вопрос

Происходит ли инициализация
объекта, при создании объекта класса?

Вопрос

Нигде не утверждается, что объект должен быть инициализирован, и программист может забыть инициализировать его или сделать это дважды.

Цели урока

- изучить конструкторы и деструкторы
- отработать на практике написание алгоритмов с конструкторами и деструкторами на C++

Конструктор

Конструктор — функция, предназначенная для инициализации объектов класса

ООП дает возможность программисту описать функцию, явно предназначенную для инициализации объектов. Поскольку такая функция конструирует значения данного типа, она называется **конструктором**. Конструктор всегда имеет то же имя, что и сам класс и никогда не имеет возвращаемого значения.

Когда класс имеет конструктор, все объекты этого класса будут проинициализированы.

Конструктор

```
class date {
  int day, month, year;
public:
  date(int, int, int); // конструктор
};
```

Конструктор

Если конструктор требует аргументы, их следует указать:

```
date today = date(10,01,2023); // полная форма
date xmas(10,01,2023); // сокращенная форма
```

Неверное создание объекта.

```
date my_burthday; // недопустимо, опущена инициализация
```

Конструктор

Если необходимо обеспечить несколько способов инициализации объектов класса, задается несколько конструкторов:

```
class date {
  int month, day, year;
public:
  date(int, int, int); // день месяц год
  date(char*); // дата в строковом представлении
  date(); // дата по умолчанию: сегодня
};
```

Конструктор

Если конструкторы существенно различаются
по типам своих параметров, то компилятор при каждом использовании может выбрать правильный:

```
date july4("Февраль 27, 2014");

date guy(27, 2, 2014);

date now; // инициализируется по умолчанию
```

Конструктор по умолчанию

Конструктор, не требующий параметров, называется **конструктором по умолчанию**.

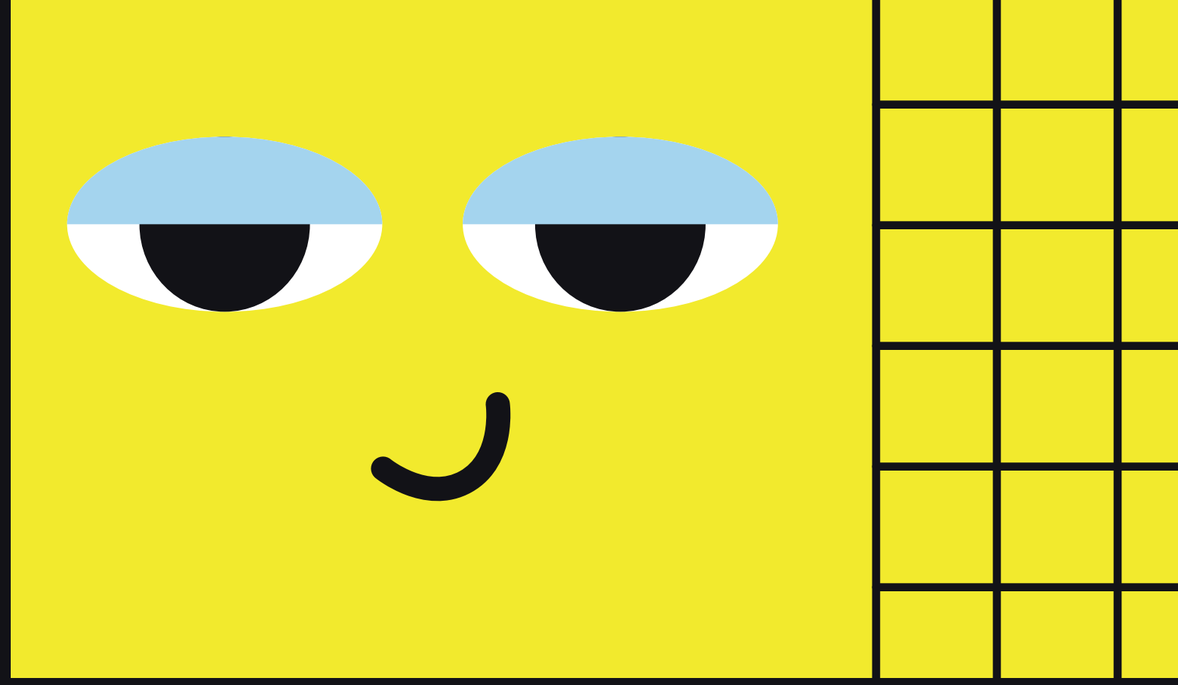
```
class date {
  int month, day, year;
public:
  date(int, int, int); // день месяц год
  date(char*); // дата в строковом представлении
  date(); // дата по умолчанию: сегодня
};
```

Деструкторы

- Определяемый пользователем класс имеет конструктор, который обеспечивает надлежащую инициализацию. Для многих типов также требуется обратное действие. Деструктор обеспечивает соответствующую очистку объектов указанного типа.
- Имя деструктора представляет собой имя класса с предшествующим ему знаком «тильда» ~.
- Так, для класса X деструктор будет иметь имя ~X().
- Многие классы используют динамическую память, которая выделяется конструктором, а освобождается деструктором.

Деструкторы

```
class date
{
  int day, year;
  char *month;
public:
  date(int d, char* m, int y)
  {
    day = d;
    month = new char[strlen(m)+1]; //выделяем память
    strcpy_s(month, strlen(m)+1,m); //копируем строку
    year = y;
  }
  ~date() { delete[] month; } // деструктор
};
```

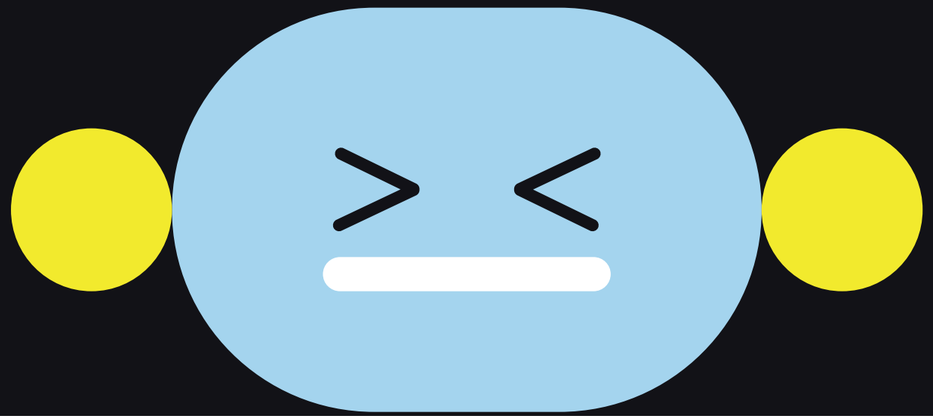


Практика

Перерыв физкультминутка

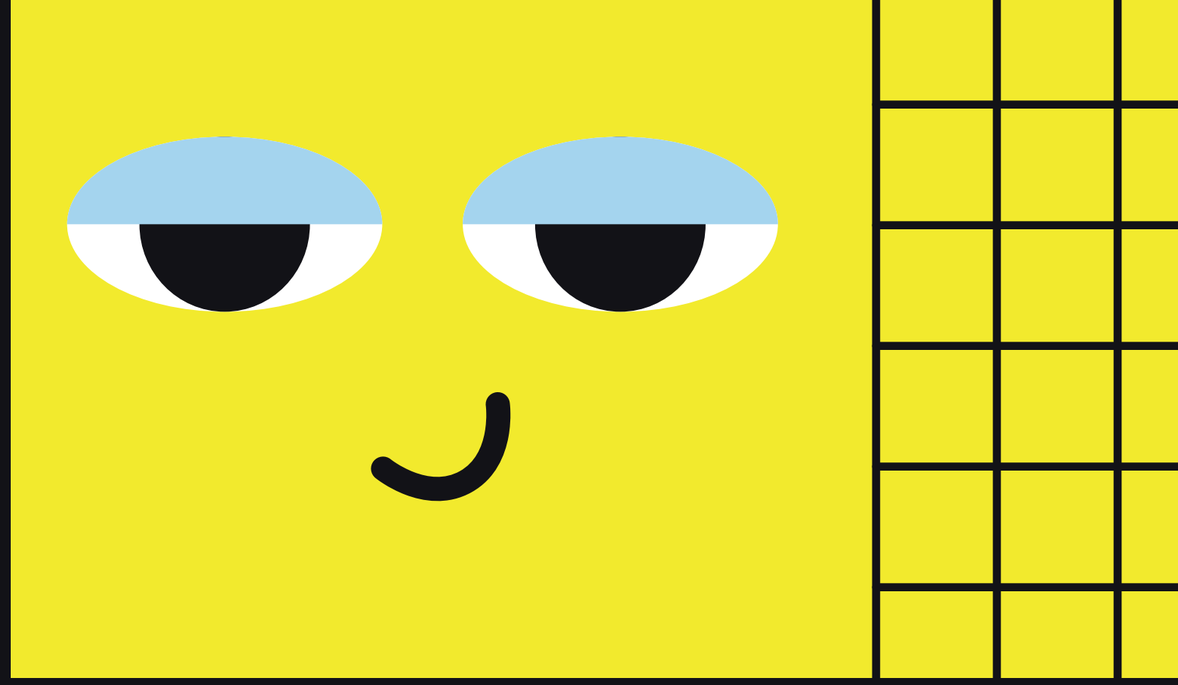
Смотрим вверх–вниз, вправо–влево

Вращаем по кругу туда–обратно



Крепко зажмуриваемся

Быстро моргаем



Практика

Закрепление

Как должен называться конструктор объектов класса?

```
class Person
{
  char *fname;
  char *lname;
  int age;
public:


};
```

Закрепление

Как должен называться конструктор объектов класса?

```
class Person
{
  char *fname;
  char *lname;
  int age;
public:
  Person (char* f, char* l, int a)
 {
    ...
 }
};
```

Закрепление

Как должен называться деструктор объектов класса?

```
class Person
{
  char *fname;
  char *lname;
  int age;
public:


};
```

Закрепление

Для как должен называться деструктор объектов класса?

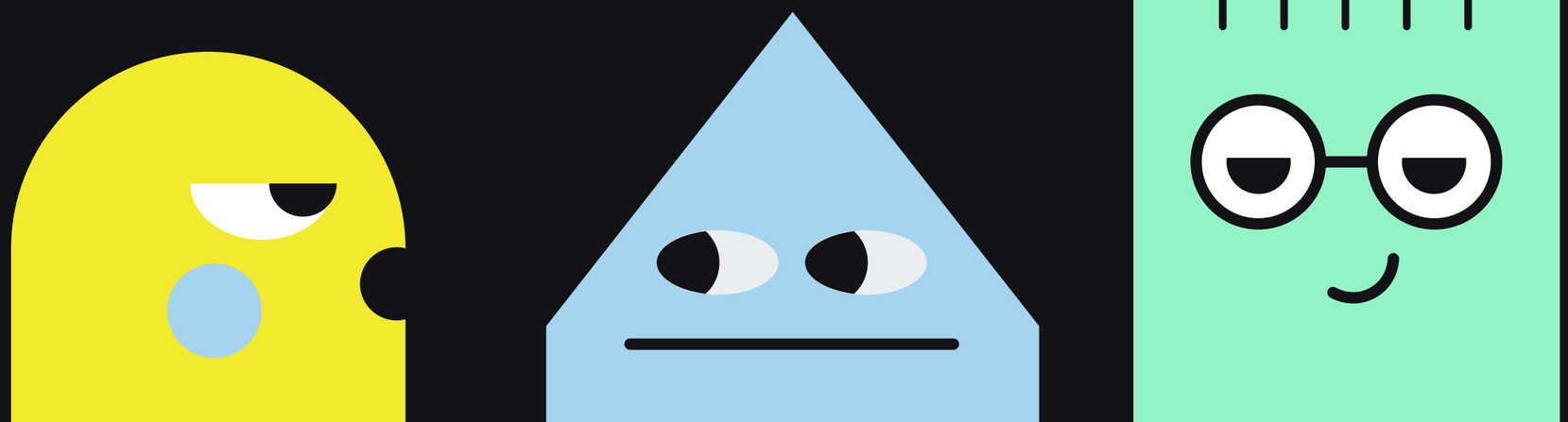
```
class Person
{
  char *fname;
  char *lname;
  int age;
public:
  Person (char* f, char* l, int a)
 {
    ...
 }
  ~Person() { delete[] fname; delete[] lname;}
};
```

Подведём итоги

- изучили конструкторы и деструкторы
- отработали на практике написание алгоритмов с конструкторами и деструкторами на C++

Оцени сложность урока

1	если тебе было совсем просто
2	было достаточно просто, но ты узнал(а) что-то новое
3	было не очень просто, но достаточно комфортно, ты узнал(а) много нового
4	было сложно, ты не знал(а) ничего из материала
5	было слишком сложно, многое осталось для тебя непонятным



Домашнее задание

До встречи!