

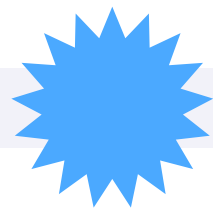
Программирование на C++



Минцифры
России

UCHi **DOMA**

20.35
УНИВЕРСИТЕТ

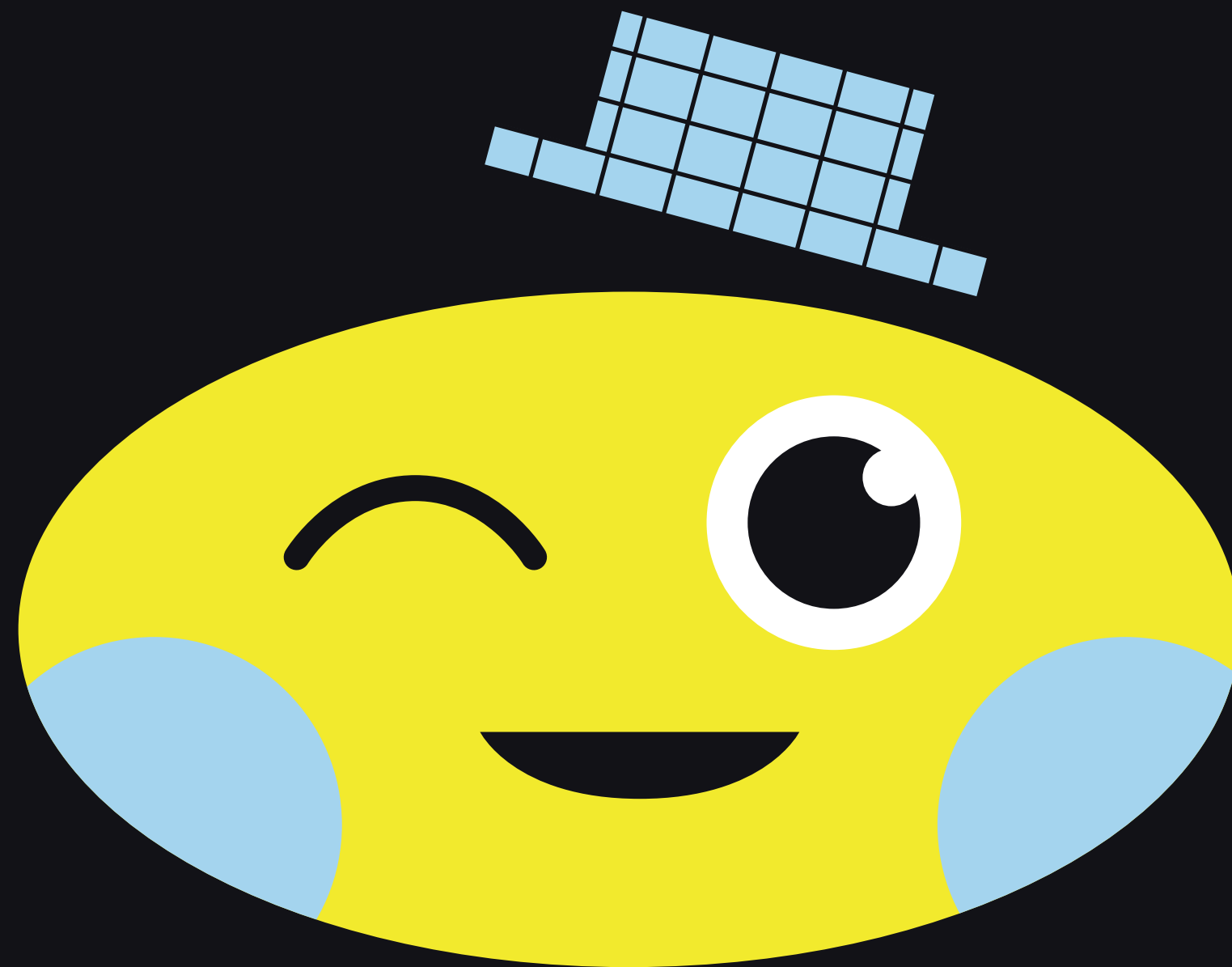


Модуль 3. Урок 3

Классы

```
unreadystatechan
String Function
n F(e){var t=_le
.stopOnFalse){r=
&&(s=t,c(r))}ret
rn u=[],this},di
{return p.fireWi
ate:function(){r
nise().done(n.re
{n=s},t[1^e][2].
rguments),r=n.le
);r>t;t++)n[t]&&
ole><a href='/a'
ut")[0],r.style.
tribute("style")
```

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?



Разомнѐмся



Во что объединяются объекты,
обладающие одинаковым набором
свойств и методов?

Разомнѐмся



Объекты принадлежат к определенному классу.



Все объекты одного класса обладают одинаковым набором свойств и методов.

Цели урока



изучить абстрактные типы данных



отработать на практике
написание алгоритмов
с использованием
абстрактных типов данных C++



Абстрактные типы данных



Язык C++ позволяет создавать типы данных, которые ведут себя аналогично базовым типам языка Си.

Такие типы обычно называют **абстрактными типами данных**.

Для реализации абстрактных типов данных языке Си используются структуры. Но использование данных структурного типа значительно ограничено по сравнению с использованием базовых типов данных.

Абстрактные типы данных

Элементы структуры никак не защищены от случайной модификации. Любая функция может обратиться к элементу структуры. Это противоречит одному из основных принципов объектно-ориентированного программирования — инкапсуляции данных: никакие другие функции, кроме специальных функций манипуляции этим типом данных, не должны иметь доступ к элементам данных.

Абстрактные типы данных

Создадим структуру Struct для хранения даты date и множества функций для работы с переменными этого типа:

```
1  #include <stdio.h>
2  struct date
3  {
4      int month; // месяц
5      int day; // день
6      int year; // год
7  };
8  void set_date(date* f, int d, int m, int y)
9  {
10     f->day = d;
11     f->month = m;
12     f->year = y;
13 }
14 void print_date(date* f)
15 {
16     printf("%d.%d.%d", f->day, f->month, f->year);
17 }
18 int main()
19 {
20     date today;
21     set_date(&today, 01, 01, 2023);
22     print_date(&today);
23     return 0;
24 }
25
```

Абстрактные типы данных

Функции `set_date` и `print_date` не имеют никакой явной связи с типом данных.

Для вызова любой из описанных функций требуется в качестве аргумента передать указатель на экземпляр структуры.

Абстрактные типы данных

Связь функций и данных можно установить, описав функции как члены структуры. Эти функции могут действовать на данные, содержащиеся в самой структуре.

По умолчанию при объявлении структуры ее данные и функции являются общими, то есть у объектов типа структура нет ни инкапсуляции, ни защиты данных:

```
1  #include <stdio.h>
2  struct date
3  {
4      int month; // месяц
5      int day; // день
6      int year; // год
7      void set_date(int d, int m, int y)
8      {
9          day = d; month = m; year = y;
10     }
11     void print_date(void);
12 };
13 void date::print_date(void)
14 {
15     printf("%d.%d.%d", day, month, year);
16 }
17 int main()
18 {
19     date today;
20     today.set_date(2, 4, 2014);
21     today.print_date();
22     getchar();
23     return 0;
24 }
```

Функции-члены и данные-члены

Функции, описанные в теле абстрактного типа данных, представляют собой **функции-члены** или **методы** и могут вызываться только для специальной переменной соответствующего типа с использованием стандартного синтаксиса для доступа к **данным-членам** или **полям** структуры.

Функции-члены и данные-члены

Определение функций-членов может осуществляться двумя способами:

- ★ описание функции непосредственно при описании структуры
- ★ описание функции вне структуры

Функции-члены, которые определены внутри структуры, являются неявно встроенными



В функции-члене имена членов могут использоваться без явной ссылки на объект. В этом случае имя относится к члену того объекта, для которого функция была вызвана.

```
тип АДД::имя(список аргументов)
{
    тело функции-члена;
}
```

- ★ тип — тип возвращаемого значения функции-члена
- ★ АДД — имя абстрактного типа данных (имя структуры или класса)
- ★ имя — имя функции-члена

```
void date::print_date(void)
{
    printf("%d.%d.%d", day, month, year);
}
```

Права доступа

Концепция структуры в языке C++ (в отличие от Си) позволяет членам структуры быть общими, частными или защищенными:



public — общие



private — частные



protected — защищенные

Права доступа

Использование ключевого слова **protected** связано с понятием **наследования**.

Использование ключевого слова **private** ограничивает доступ к членам, которые следуют за этой конструкцией. Члены **private** могут использоваться только несколькими категориями функций, в привилегии которых входит доступ к этим членам. В основном это функции-члены той же структуры.

Ключевое слово **public** образует интерфейс к объекту структуры.

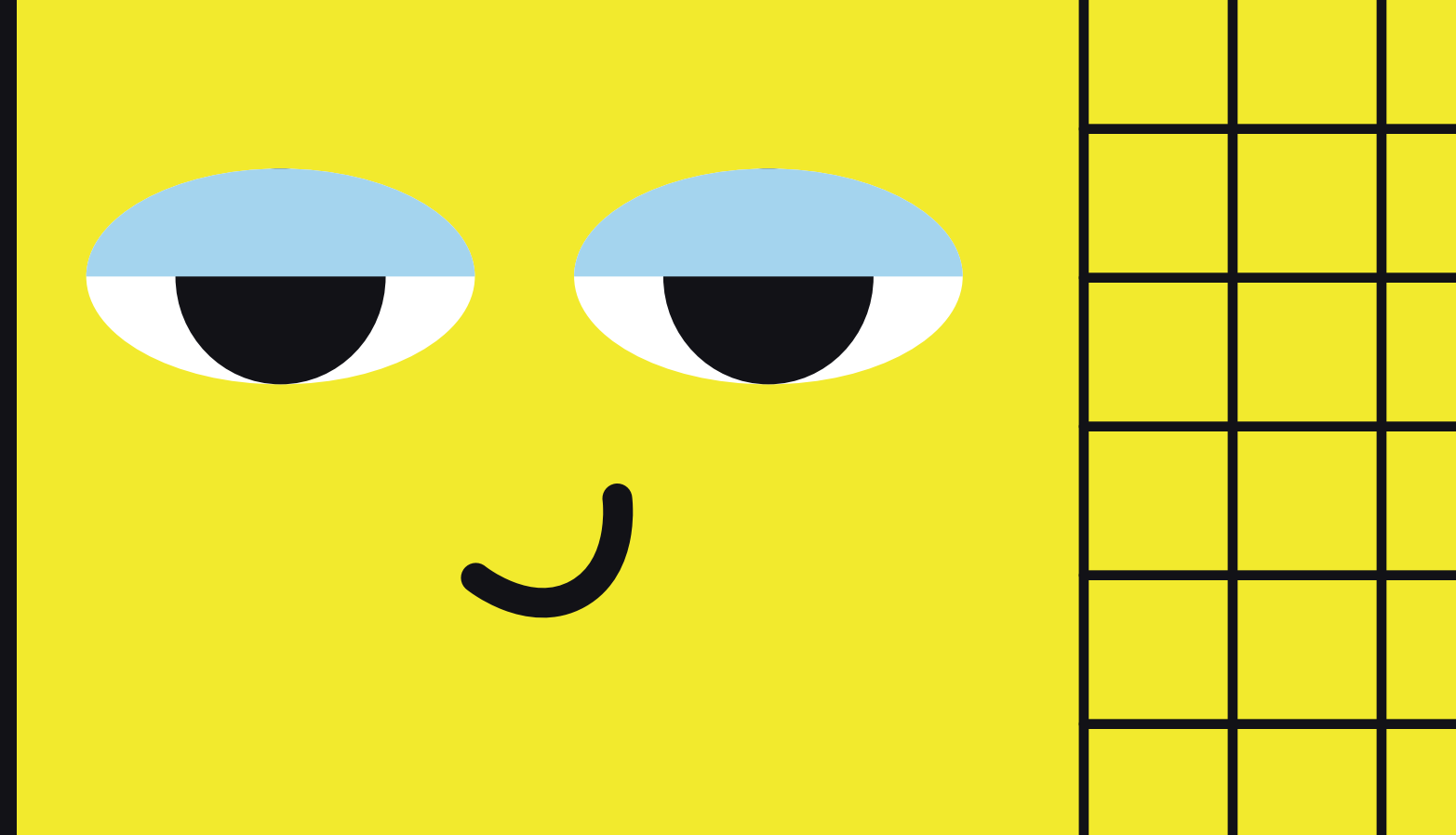
Права доступа

Стандартным является размещение член-данных в частной области (**private**), а части функций-членов — в общей части (**public**) абстрактного типа данных. В этом случае закрытая (**private**) часть определяет данные объекта и служебные функции, а функции-члены общей части реализуют методы работы с объектом.

Права доступа

Изменим структуру **date** так, чтобы скрыть представление данных (инкапсуляция данных):

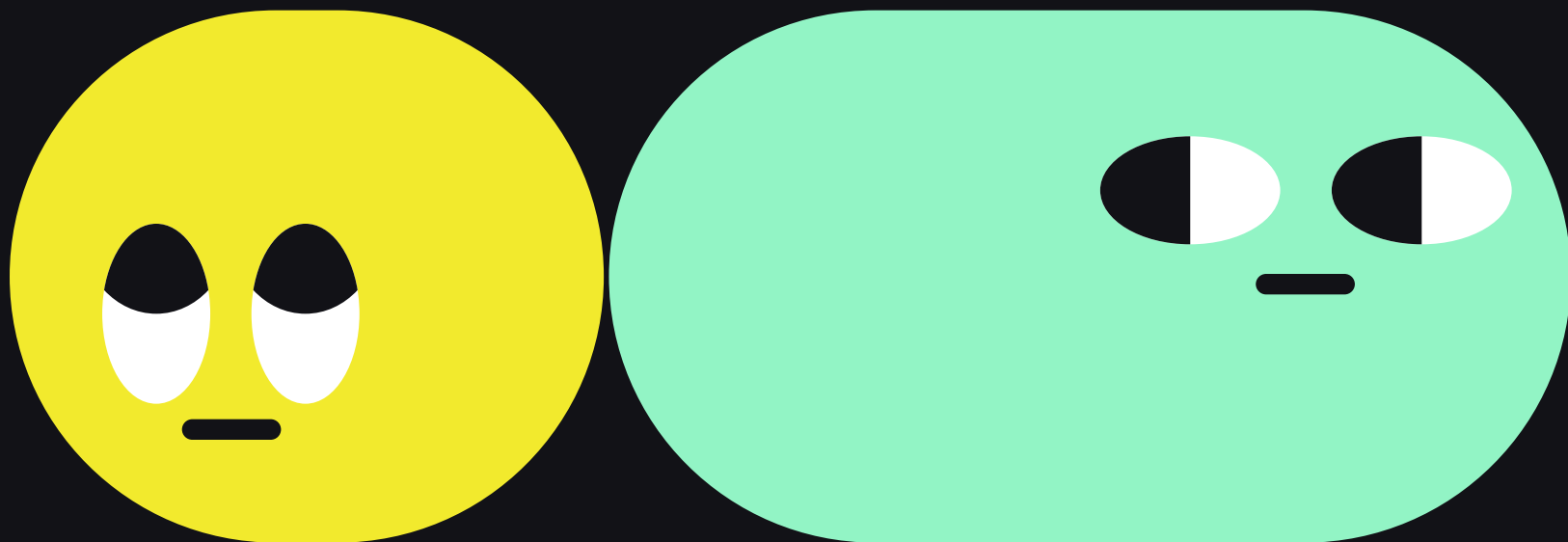
```
struct date
{
    private:
        int month, day, year;
    public:
        void set(int, int, int);
        void print();
};
```



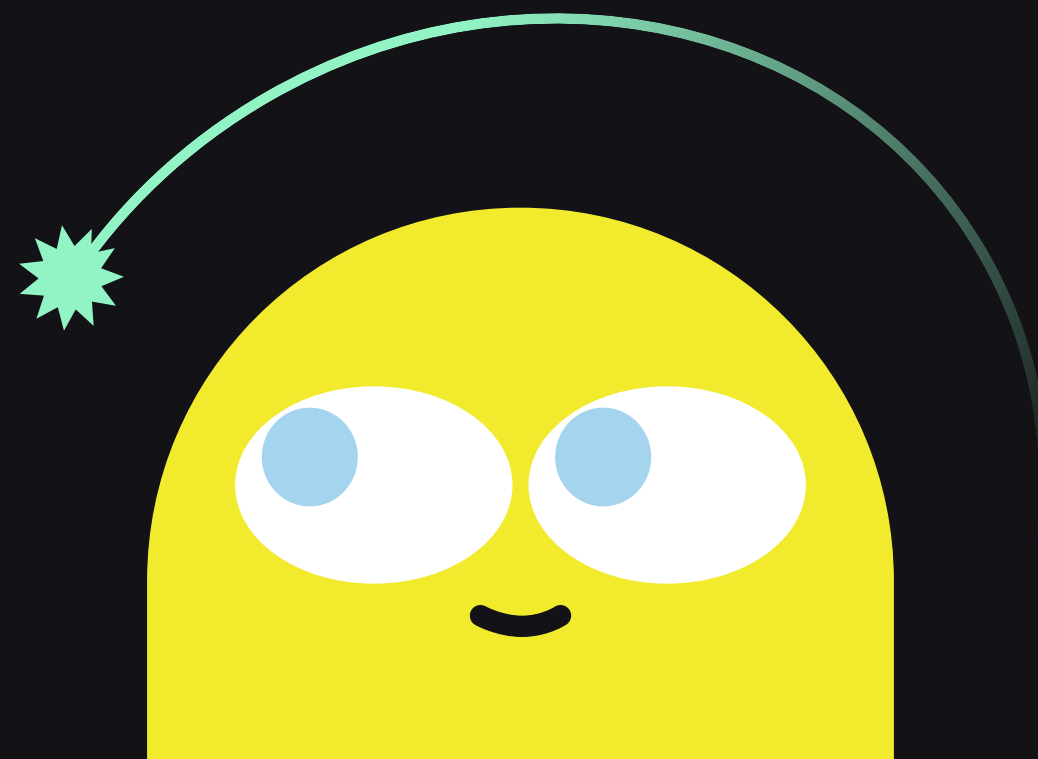
Практика

Перерыв

физкультминутка



Смотрим вверх–вниз, вправо–влево



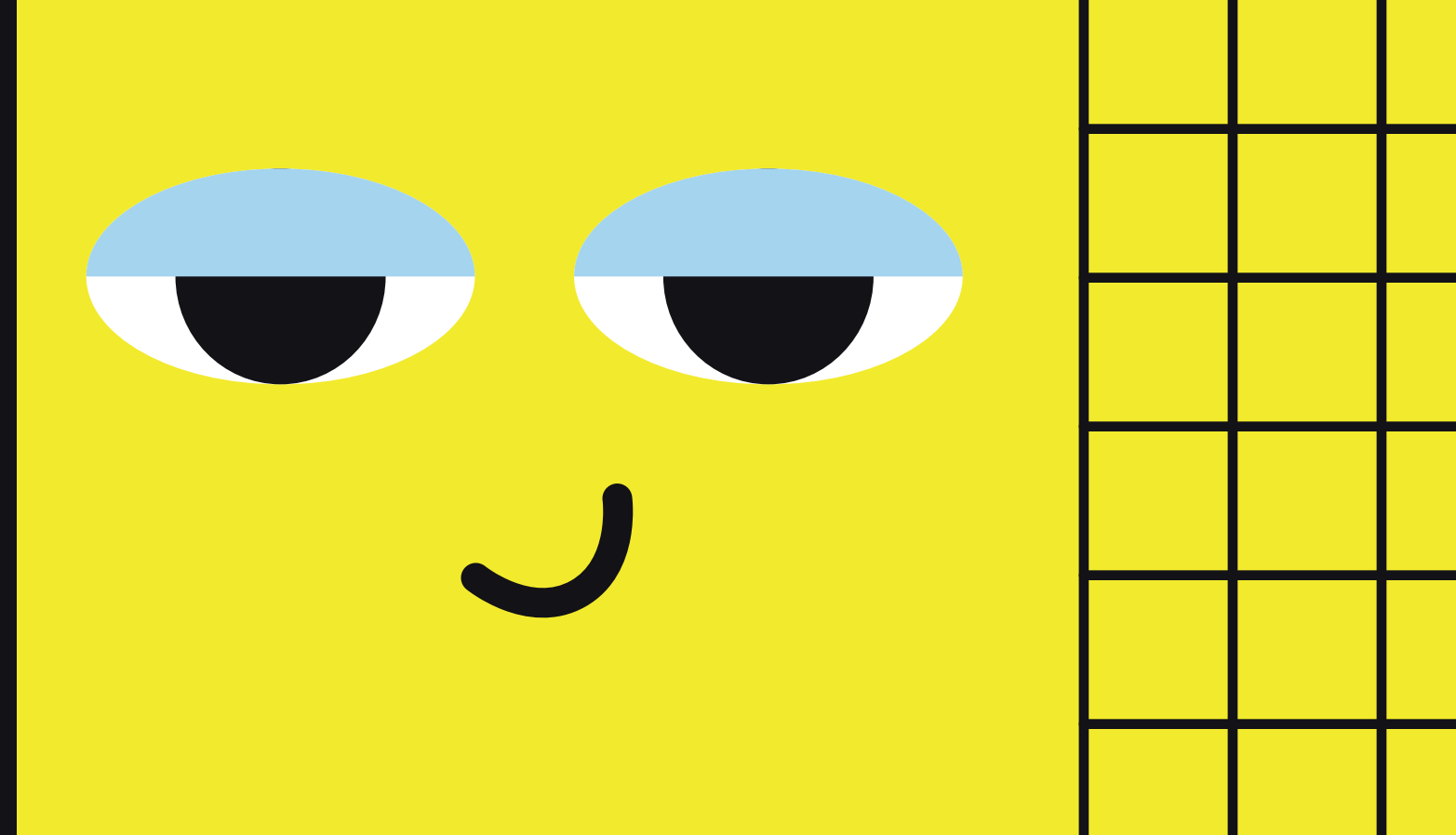
Вращаем по кругу туда–обратно



Крепко зажимаемся



Быстро моргаем



Практика

Закрепление

```
1  #include <iostream>
2  using namespace std;
3
4  struct Adder {
5      public:
6          void addNum(int number) {
7              total += number;
8          }
9          int getTotal() {
10             return total;
11         };
12     private:
13         int total;
14 };
15 int main() {
16     Adder a;
17
18     a.addNum(10);
19     a.addNum(20);
20     a.addNum(30);
21
22     cout << "Total " << a.getTotal() << endl;
23     return 0;
24 }
```

Что будет выведено на экран
в результате работы программы?

Закрепление

```
1  #include <iostream>
2  using namespace std;
3
4  struct Adder {
5      public:
6          void addNum(int number) {
7              total += number;
8          }
9          int getTotal() {
10             return total;
11         };
12     private:
13         int total;
14 };
15 int main() {
16     Adder a;
17
18     a.addNum(10);
19     a.addNum(20);
20     a.addNum(30);
21
22     cout << "Total " << a.getTotal() << endl;
23     return 0;
24 }
```

Что будет выведено на экран
в результате работы программы?

Результат работы программы:

Total 60



Подведём итоги

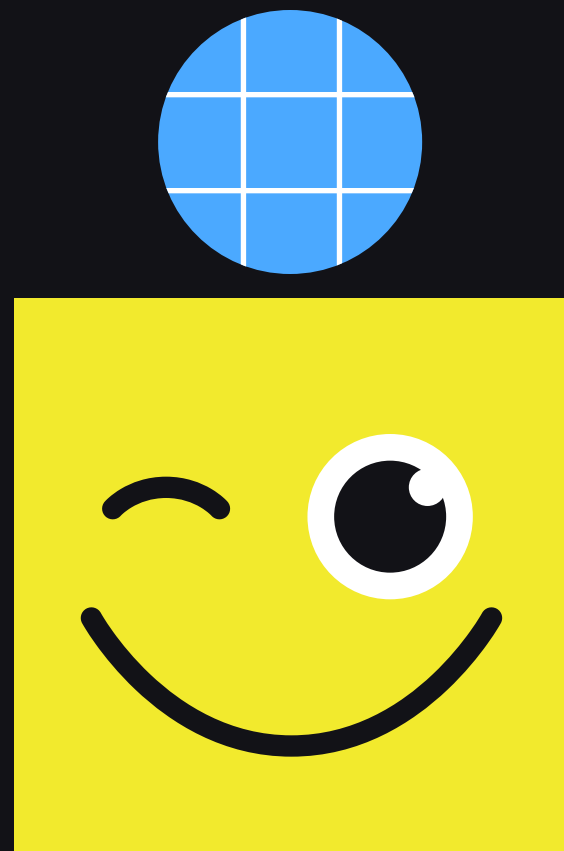


изучили абстрактные типы данных



отработали на практике
написание алгоритмов
с использованием абстрактных
типов данных C++

Итоги урока



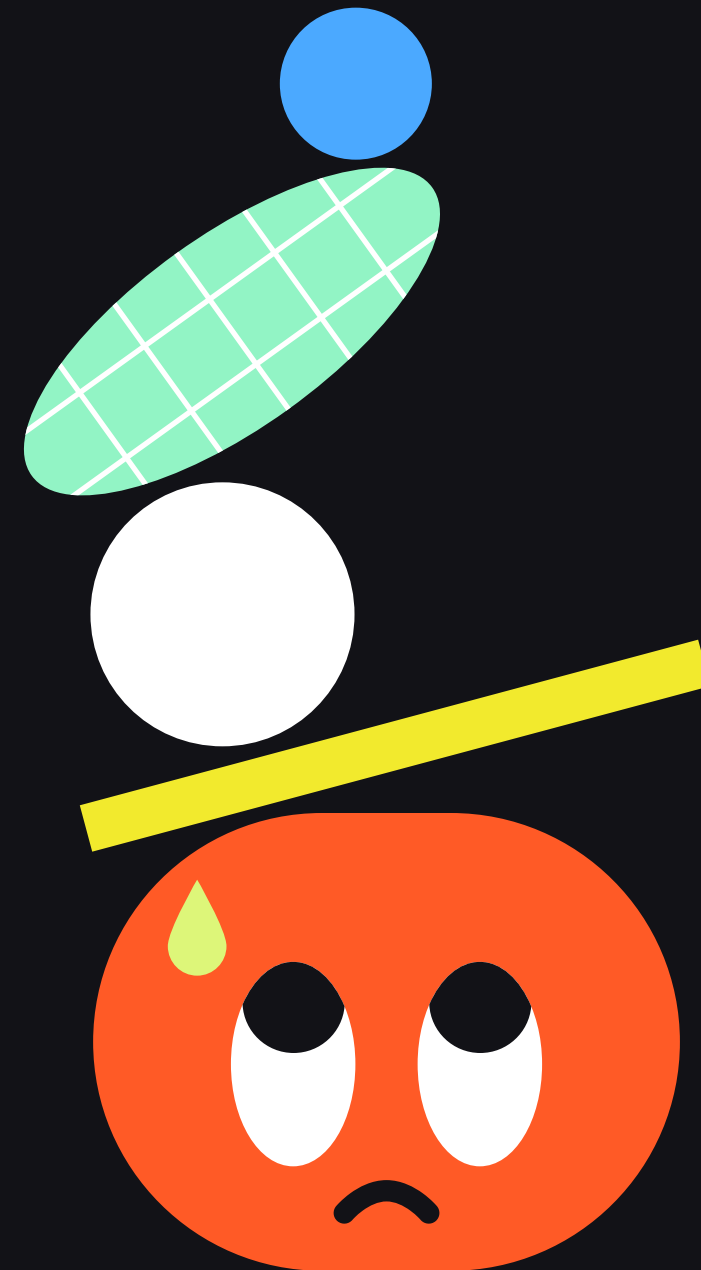
1

легко!



2

сложно, но можно



3

трудно

Домашнее задание



До встречи!