

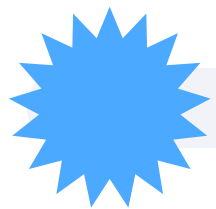
# Программирование на C++



Минцифры  
России

UCHi **DOMA**

**20.35**  
УНИВЕРСИТЕТ

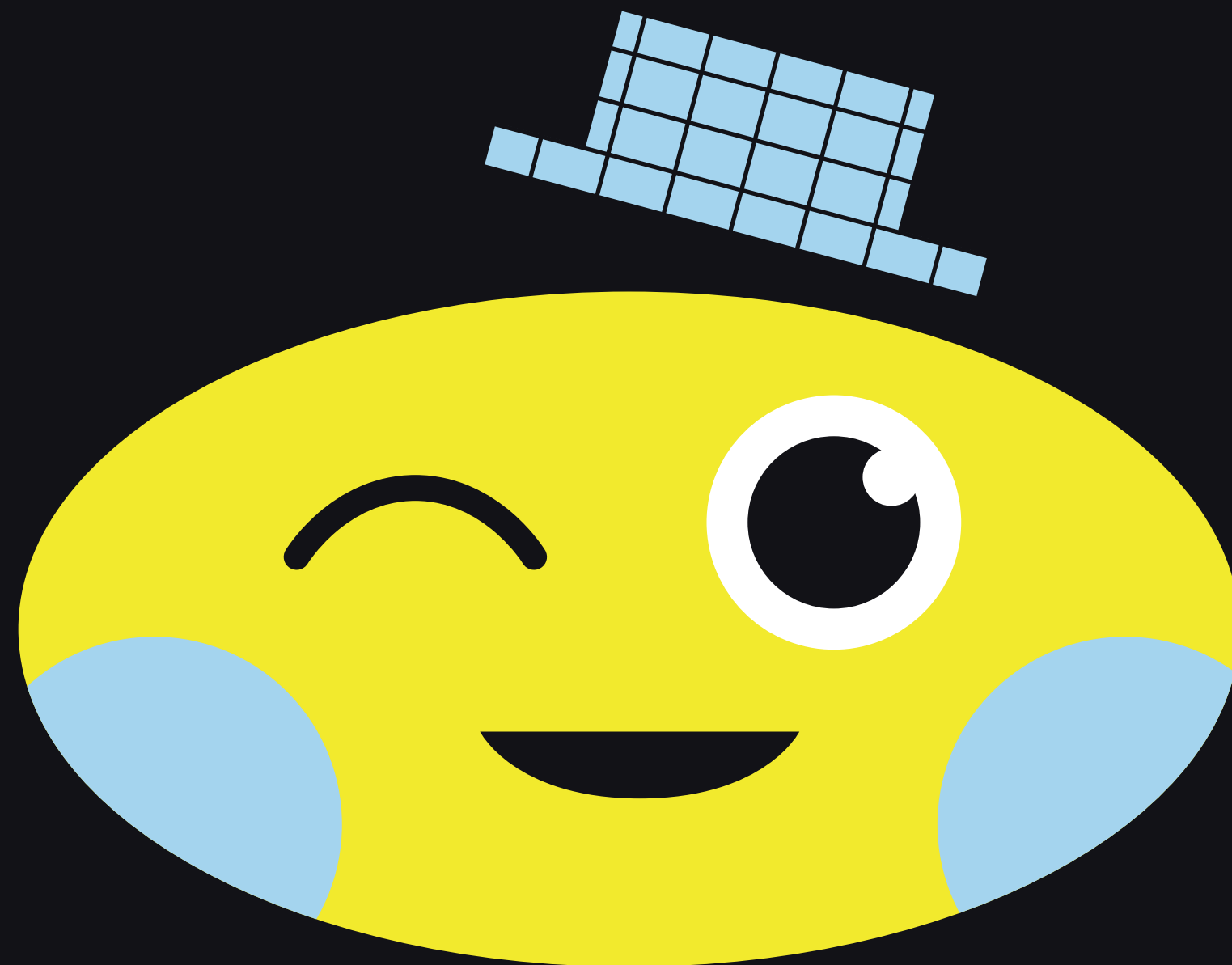


Модуль 3. Урок 1

# СИ C++



# 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



Кто Создал язык Си?

# Разомнёмся



Си разработан в 1969–1973 годах  
сотрудником Bell Labs **Деннисом Ритчи**  
как развитие языка Би.



Разомнѐмся



Кто Создал язык C++?



# Разомнёмся



Язык программирования C++ был создан в начале 1980-х годов, его создатель сотрудник фирмы Bell Laboratories — **Бьёрн Страуструп**.



Вопрос



Можно ли код, написанный на C,  
запустить на C++?



Ответ



Код, написанный на C, запустится в C++, так как C++ обладает обратной совместимостью с C

# Цели урока



изучить отличия С и С++



отработать на практике  
составление алгоритмов  
с вводом/выводом,  
выделением динамической  
памяти на С++



# Отличия С и С++

	С	С++
Появление	Разработал Деннис Ричи в 1969г. в Лабаратории Белла	Разработал Бьярне Страуструп в 1979г
Класс языка	Процедурный	Объектно-ориентированный, мультипарадигмальный, процедурный, функциональный, обобщённый
ООП	Поскольку С не поддерживает концепцию ООП, то он не поддерживает полиморфизм, инкапсуляцию и наследование	С++ поддерживает полиморфизм, инкапсуляцию и наследование, т. к. является объектно-ориентированным языком
Особенности	Не поддерживает перегрузку функций и операторов	С++ поддерживает как и перегрузку функций, так и перегрузку операторов, а также пространства имен и ссылки, обработка исключений, богатая библиотека

# Отличия С и С++

	С	С++
Ключевые слова	32 ключевых слова	52 ключевых слова
Функции	Не поддерживаются в структуре, не могут быть «друзьями» и виртуальными	Поддерживаются в структуре, могут быть «друзьями» и виртуальными
Память	Функции <b>malloc()</b> и <b>calloc()</b> для динамического выделения памяти, а также <b>free()</b> для освобождения	Для этих же операций используются операторы <b>new</b> и <b>delete</b>
Вход/Выход	Используются <b>scanf()</b> и <b>printf()</b>	Используются <b>cout</b> и <b>cin</b>

# Поточный ввод-вывод в C++



В C++, как и в C, нет встроенных в язык средств ввода-вывода



В C для этих целей используется библиотека **stdio.h**.



В C++ разработана новая библиотека ввода-вывода **iostream**, использующая концепцию объектно-ориентированного программирования:

```
#include <iostream>
```

# Поточный ввод-вывод в C++

Библиотека **`iostream`** определяет три стандартных потока:



**`cin`** стандартный входной поток (**`stdin`** в C)



**`cout`** стандартный выходной поток (**`stdout`** в C)



**`cerr`** стандартный поток вывода сообщений об ошибках (**`stderr`** в C)



# Поточный ввод-вывод в C++

Для выполнения операций ввода-вывода переопределены две операции поразрядного сдвига:



>> получить из входного потока



<< поместить в выходной поток

# Поточный вывод в C++

```
cout << значение;
```

Здесь значение преобразуется в последовательность символов и выводится в выходной поток:

```
cout << s;
```

Возможно многократное назначение потоков:

```
cout << 'значение1' << 'значение2' << ... << 'значение n';
```

# Пример поточного вывода

```
1  #include <iostream>
2  using namespace std;
3  int main()
4  {
5  int a=10,b=2,sum=0;
6  sum=a+b;
7  cout << a<<" + "<< b<<" = " << sum;
8  return 0;
9  }
```

Результат работы программы:

10 + 2 = 12

# Поточный ввод информации в C++

`cin >> идентификатор;`

При этом из входного потока читается последовательность символов до пробела, затем эта последовательность преобразуется к типу идентификатора, и получаемое значение помещается в идентификатор

Возможно многократное назначение потоков:

`cin >> переменная1 >> переменная2 >>...>> переменнаяn;`

# Пример поточного ввода информации

```
1  #include <iostream>
2  using namespace std;
3  int main()
4  {
5  int a,b,sum=0;
6  cin>>a>>b;
7  sum=a+b;
8  cout << a<<" + "<< b<< " = " << sum;
9  return 0;
10 }
```

Результат работы программы:

```
2
3
2 + 3 = 5
```

или

```
23
2 + 3 = 5
```

# Ввод символьных строк

По умолчанию потоковый ввод `cin` вводит строку до пробела, символа табуляции или перевода строки

```
1  #include <iostream>
2  using namespace std;
3  int main()
4  {
5      char s[80];
6      cin >> s;
7      cout << s << endl;
8      return 0;
9  }
```

Результат работы программы:

```
Привет, пользователь
Привет,
```

# Ввод символьных строк

Для ввода текста до символа перевода строки используется манипулятор потока **getline()**

```
1  #include <iostream>
2  using namespace std;
3  int main()
4  {
5      char s[80];
6      cin.getline(s, 80);
7      cout << s << endl;
8      return 0;
9  }
```

Результат работы программы:

```
Привет, пользователь
Привет, пользователь
```

# Манипуляторы потока

Функцию-манипулятор потока можно включать в операции помещения в поток и извлечения из потока (<<, >>)

Манипулятор	Описание
<code>endl</code>	Помещение в выходной поток символа конца строки '\n'
<code>dec</code>	Установка основания 10-ой системы счисления
<code>oct</code>	Установка основания 8-ой системы счисления
<code>hex</code>	Установка основания 16-ой системы счисления
<code>setbase</code>	Вывод базовой системы счисления
<code>width(ширина)</code>	Устанавливает ширину поля вывода
<code>fill('символ')</code>	Заполняет пустые знакоместа значением символа



# Манипуляторы потока

Манипулятор	Описание
<code>precision</code> (точность)	Устанавливает количество значащих цифр в числе (или после запятой) в зависимости от использования <b>fixed</b>
<code>fixed</code>	Показывает, что установленная точность относится к количеству знаков после запятой
<code>showpos</code>	Показывает знак + для положительных чисел
<code>scientific</code>	Выводит число в экспоненциальной форме
<code>get()</code>	Ожидает ввода символа
<code>getline</code> (указатель, количество)	Ожидает ввода строки символов. Максимальное количество символов ограничено полем <b>количество</b>

# Программы на Си в С++



Код, написанный на Си, будет работать в С++



С++ обратно совместим с С

# Пример программы на С и С++

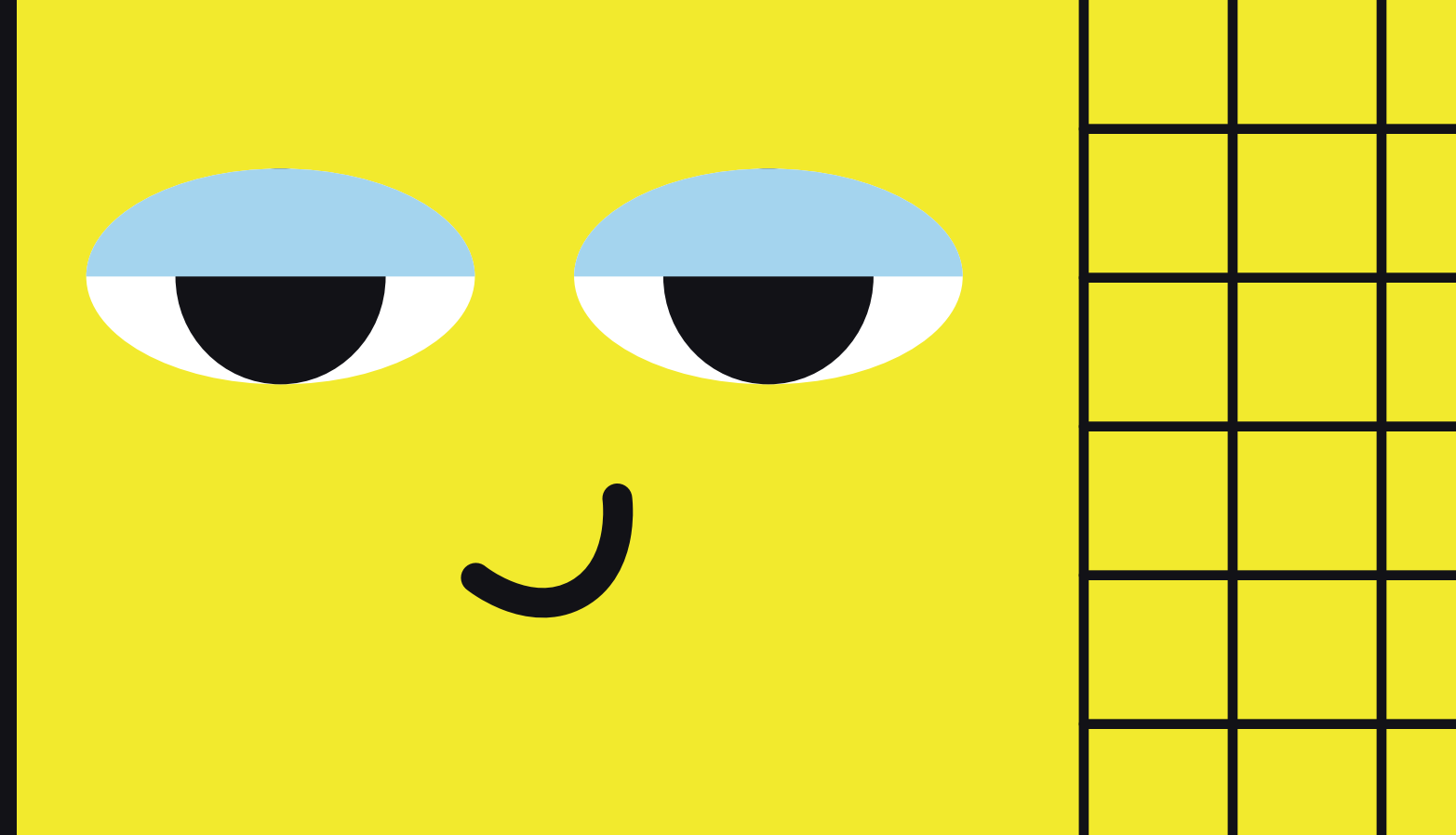
## Программа ввода-вывода значения переменной

### Программа на Си

```
1  #include <stdio.h>
2  int main()
3  {
4  ▼ int n;
5      printf("Введите n:");
6      scanf("%d", &n);
7      printf("Значение n равно: %d\n", n);
8      return 0;
9  }
```

### Программа на С++

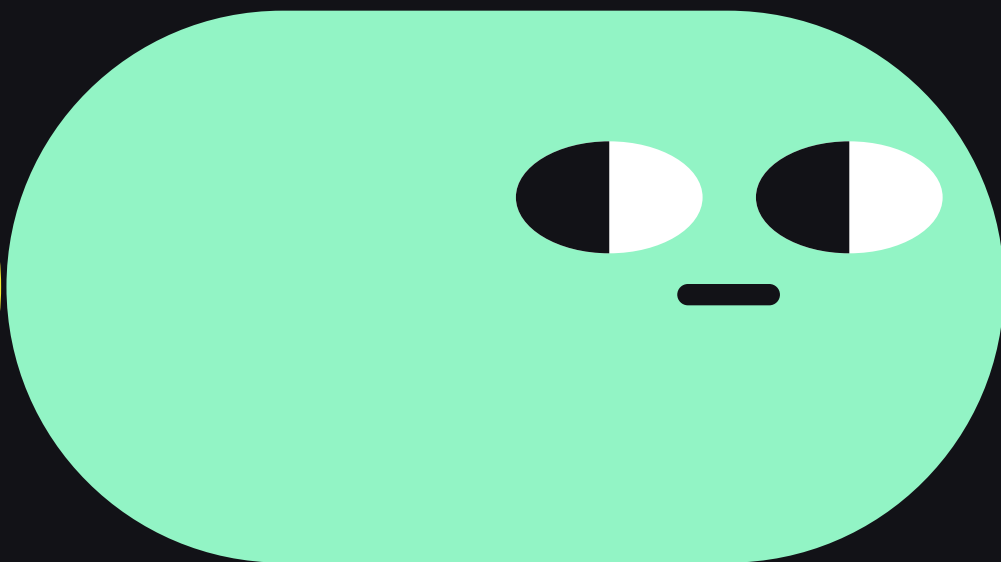
```
1  #include <iostream>
2  using namespace std;
3  int main()
4  ▼ {
5      int n;
6      cout << "Введите n:";
7      cin >> n;
8      cout << "Значение n равно: " << n << endl;
9      return 0;
10 }
```



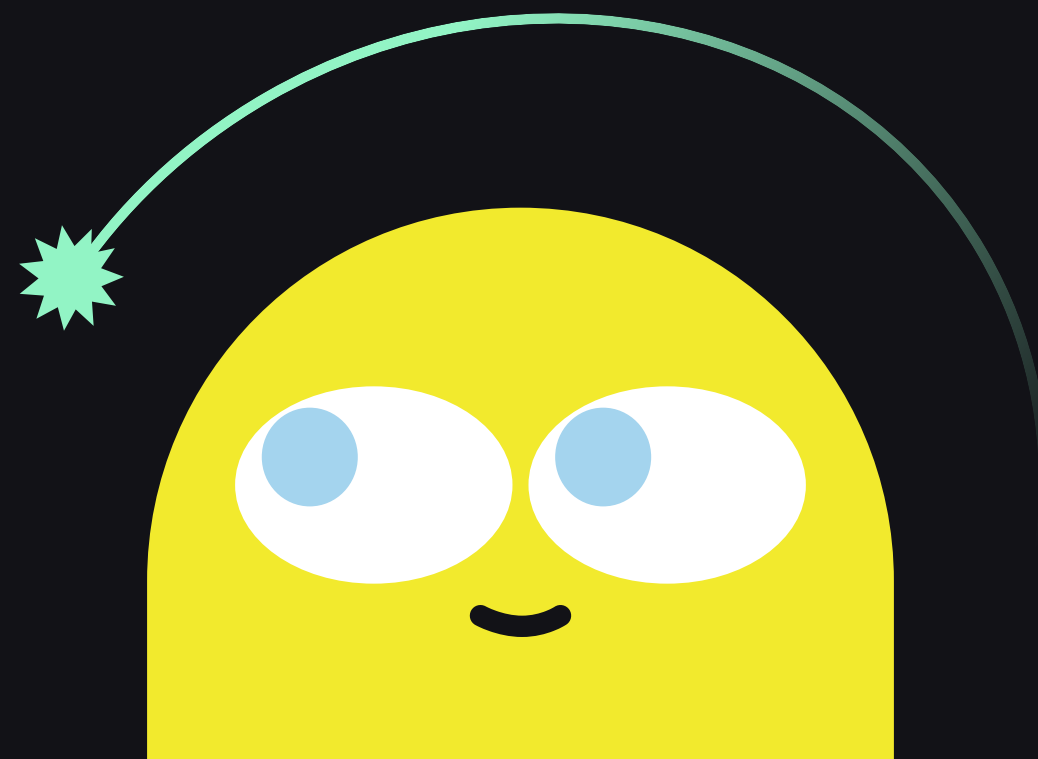
# Практика

перерыв

физкультминутка



Смотрим вверх–вниз, вправо–влево



Вращаем по кругу туда–обратно



Крепко зажимаемся



Быстро моргаем

# Динамическое выделение памяти в C++

В Си работать с динамической памятью можно при помощи соответствующих функций распределения памяти (**calloc**, **malloc**, **free**), для чего необходимо подключить библиотеку **malloc.h**

# Динамическое выделение памяти в C++

C++ использует новые методы работы с динамической памятью при помощи операторов **new** и **delete**:

Оператор **new** используется в следующих формах:



**new** — для выделения памяти



**new тип;** — для переменных



**delete** — для освобождения памяти



**new тип[размер];** — для массивов



Результатом выполнения операции **new** будет указатель на отведенную память, или исключение **std::bad\_alloc** в случае ошибки.

# Динамическое выделение памяти в C++

```
int *ptr_i;  
double *ptr_d;  
struct person *human;  
...  
ptr_i = new int;  
ptr_d = new double[10];  
human = new struct person;
```



# Динамическое выделение памяти в C++

Память, отведенная в результате выполнения **new**, будет считаться распределенной до тех пор, пока не будет выполнена операция **delete**.

Освобождение памяти связано с тем, как выделялась память — для одного элемента или для нескольких. В соответствии с этим существует и две формы применения **delete**:

✦ **delete** указатель; — для одного элемента

✦ **delete[]** указатель; — для массивов



Освободиться с помощью **delete** может только память, выделенная оператором **new**.

# Динамическое выделение памяти в C++

```
int *ptr_i;  
double *ptr_d;  
struct person *human;  
...  
ptr_i = new int;  
ptr_d = new double[10];  
human = new struct person;  
  
delete ptr_i;  
delete[] ptr_d;  
delete human;
```

# Пример создания динамического массива

```
1  #include <iostream>
2  using namespace std;
3  int main()
4  {
5      int size;
6      int *mas;
7      cout << "Ввести размерность массива : ";
8      cin >> size;
9      mas = new int[size];
10     for (int i = 0; i < size; i++) {
11         cout << "mas[" << i << "] = ";
12         cin >> mas[i];
13     }
14     for (int i = 0; i < size; i++)
15         cout << mas[i] << " ";
16     delete[] mas;
17     return 0;
18 }
```

Результат работы программы:

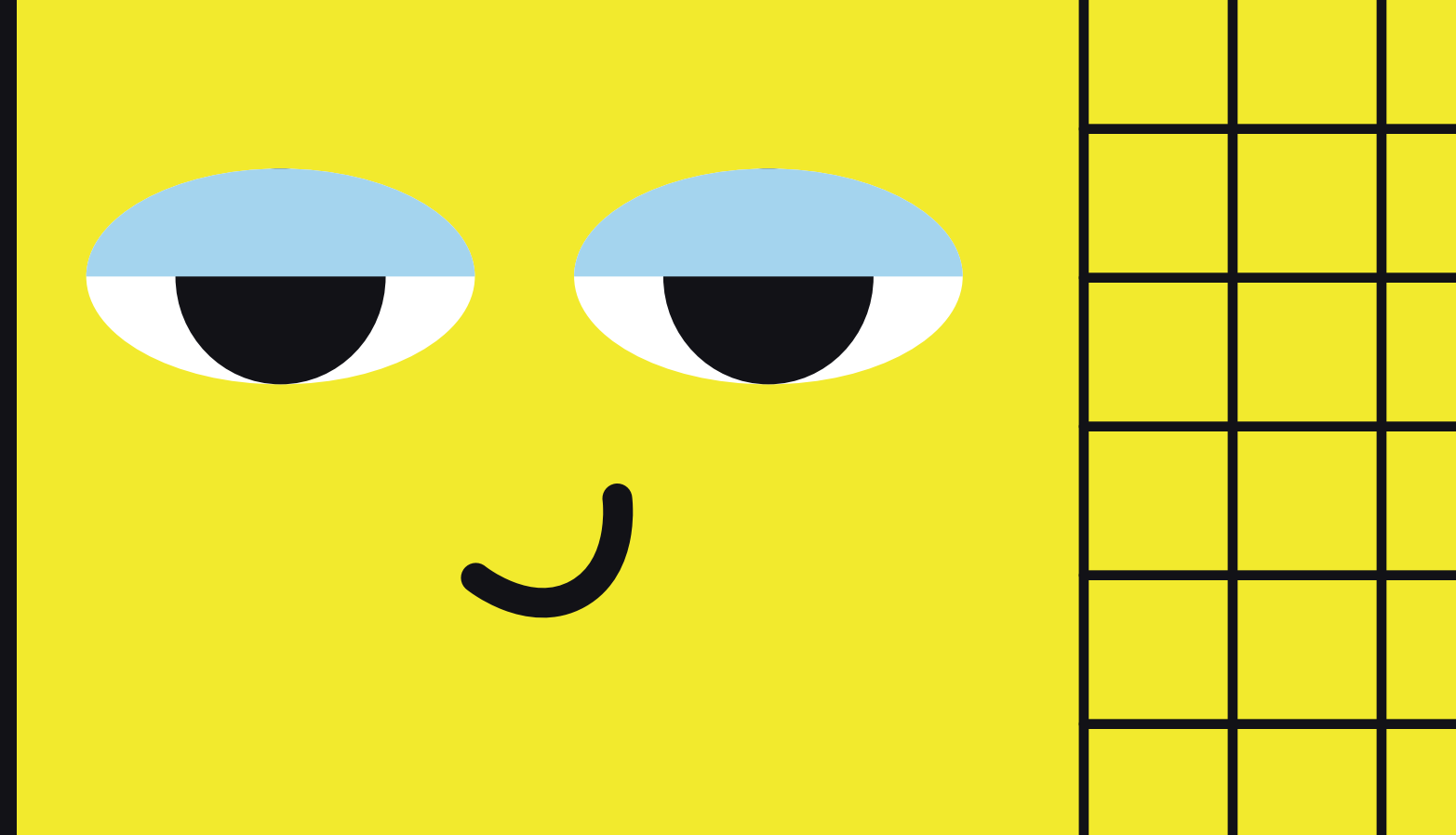
Ввести размерность массива : 3

mas [0] = 1

mas [1] = 2

mas [2] = 3

1 2 3



# Практика

# Закрепление

Программа должна запрашивать два целых числа и выводить их сумму, выберите программу, которая будет работать в C++

1

```
1  #include <iostream>
2  using namespace std;
3  int main()
4  {
5      int a,b;
6      cin >> a>>b;
7      cout << a+b;
8      return 0;
9  }
```

2

```
1  #include <stdio.h>
2  int main()
3  {
4  ▾  int a,b;
5      scanf("%d%d",&a,&b);
6      printf("%d", a+b);
7      return 0;
8  }
```

3

```
1  #include <iostream>
2  int main()
3  {
4  ▾  int a,b;
5      scanf("%d%d",&a,&b);
6      printf("%d", a+b);
7      return 0;
8  }
```

# Закрепление

Программа должна запрашивать два целых числа и выводить их сумму, выберите программу, которая будет работать в C++

1



```
1  #include <iostream>
2  using namespace std;
3  int main()
4  {
5      int a,b;
6      cin >> a>>b;
7      cout << a+b;
8      return 0;
9  }
```

2



```
1  #include <stdio.h>
2  int main()
3  {
4  ▾  int a,b;
5      scanf("%d%d",&a,&b);
6      printf("%d", a+b);
7      return 0;
8  }
```

3



```
1  #include <iostream>
2  int main()
3  {
4  ▾  int a,b;
5      scanf("%d%d",&a,&b);
6      printf("%d", a+b);
7      return 0;
8  }
```

# Закрепление

Программа должна запрашивать два целых числа и выводить их сумму, выберите программу, которая будет работать в C

1

```
1  #include <iostream>
2  using namespace std;
3  int main()
4  {
5      int a,b;
6      cin >> a>>b;
7      cout << a+b;
8      return 0;
9  }
```

2

```
1  #include <stdio.h>
2  int main()
3  {
4  ▾  int a,b;
5      scanf("%d%d",&a,&b);
6      printf("%d", a+b);
7      return 0;
8  }
```

3

```
1  #include <iostream>
2  int main()
3  {
4  ▾  int a,b;
5      scanf("%d%d",&a,&b);
6      printf("%d", a+b);
7      return 0;
8  }
```

# Закрепление

Программа должна запрашивать два целых числа и выводить их сумму, выберите программу, которая будет работать в C

1



```
1 #include <iostream>
2 using namespace std;
3 int main()
4 {
5     int a,b;
6     cin >> a>>b;
7     cout << a+b;
8     return 0;
9 }
```

2



```
1 #include <stdio.h>
2 int main()
3 {
4     int a,b;
5     scanf("%d%d",&a,&b);
6     printf("%d", a+b);
7     return 0;
8 }
```

3



```
1 #include <iostream>
2 int main()
3 {
4     int a,b;
5     scanf("%d%d",&a,&b);
6     printf("%d", a+b);
7     return 0;
8 }
```





## Подведём итоги



изучили отличия С и С++



отработали на практике  
составление алгоритмов  
с вводом/выводом, выделением  
динамической памяти на С++

# Оцени сложность урока

1

было совсем  
просто

2

было достаточно  
просто, но ты  
узнал(а) что-то  
новое

3

было не очень  
просто, но  
интересно

4

было сложно,  
не знал(а) ничего  
из материала

5

было слишком  
сложно, многое  
осталось  
непонятным

# Домашнее задание



**До встречи!**