ВЫСШАЯ ШКОЛА ЭКОНОМИКИ

НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ

Департамент программной инженерии

Алгоритмы и структуры данных

Семинар №1. 2021-2022 учебный год

Нестеров Роман Александрович, *ДПИ ФКН и НУЛ ПОИС*

Бессмертный Александр Игоревич, *ДПИ ФКН*

Higher School of Economics

Faculty of Computer science

Цели курса

- Получить практические навыки разработки программ на языке C++
- Получить практические навыки проектирования и применения различных структур данных для решения задачи

Тематический план

Пролог (Неделя 1-2)	Программирование на C++ – *вводные занятия*
Часть 1 (Неделя 3-5)	Элементарные структуры данных – *массивы, стеки, очереди, строки*
Часть 2 (Неделя 6-7)	Рекурсивные структуры данных – *списки и деревья*
Часть 3 (Неделя 9-12)	Специальные и продвинутые структуры данных – *сбалансированные и ветвистые деревья, хеширование, хеш-таблицы, вероятностные структуры*
Часть 5 (Неделя 13-14)	Абстрактные типы данных и стандартная библиотека шаблонов (STL)
Эпилог (Неделя 15)	Свободная тема

Оценивание

- **HW** Домашние задания в системе **Яндекс.Контест**
- **SM** Мини-тесты по основному материалу (раз в 2 семинара)
- **TS** Итоговый тест
- **EX** Экзамен (с возможностью автоматической оценки)

Накопленная оценка **CM** = 0,5 · (0,9 · **HW** + 0,1 · **SM**) + 0,5 · **TS**

Итоговая оценка = 0,5 · **CM** + 0,5 · **EX**

Нам помогают

- Никита Игумнов
- Евгений Герасименко
- Вацлав Соколовский
- Джушкинбек Хамроев
- Иван Симонович
- Екатерина Штанько
- Игорь Егоров
- Даниил Горбачев

Telegram-канал дисциплины

https://t.me/joinchat/SfAuVWFPPwYwNjk6

Литература

- Сэджвик Р. *Алгоритмы на C++: анализ, структуры данных, сортировка, поиск, алгоритмы на графах.*
- Кормен Т., Лейзерсон Ч., Ривест Р., Клиффорд Ш. *Алгоритмы: построение и анализ.*
- Александреску, А. *Современное проектирование на C++: Обобщённое программирование и прикладные шаблоны проектирования.*
- Weiss M. A. *Data Structures and Algorithm Analysis in C++.*
- Джосаттис Н. М. *Стандартная библиотека C++. Справочное руководство.*

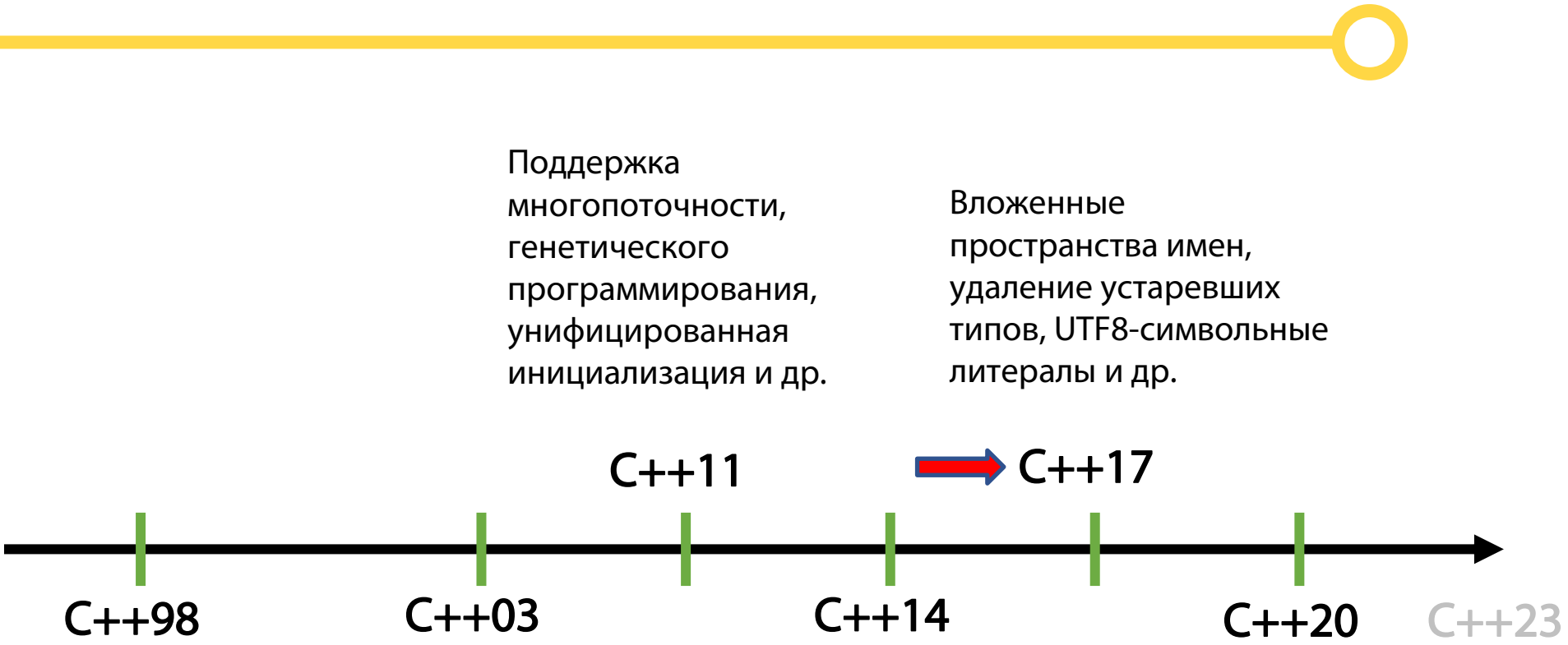
План семинара

1. C++ через призму сравнения с C#
2. Основные типы данных
3. Структура программы на C++.
   Компиляция и организация программы в памяти
4. Ввод-вывод
5. Структуры и классы

C++ versus C#

	C++	C#
Парадигмы	С поддержкой ООП	Компонентно-ориентированный
Управление памятью	Ручное выделение и освобождение памяти	Автоматическая «сборка мусора»
Кросс-платформенность	Широкая совместимость “write once, compile everywhere”	Ограниченная совместимость
Компиляция исходного кода	Напрямую в инструкции машинного кода	В инструкции промежуточного языка
Указатели	Использование без ограничений	Использования в «небезопасной» области кода
Наследование	Не ограничивается – возможно множественное	Множественное наследование не поддерживается

Стандарты C++


Поддержка многопоточности, генетического программирования, унифицированная инициализация и др.
Вложенные пространства имен, удаление устаревших типов, UTF8-символьные литералы и др.
C++11
C++17
C++98
C++03
C++14
C++20
C++23
Хранение элементов вектора (STL) в смежных ячейках памяти
Двоичные литералы, вывод типов функций, цифровые разделители, шаблоны переменных и др.

Тип `void`

Имеет **пустое** множество допустимых значений.

- Функция, которая не возвращает значение

```
void f(int a) { cout << a << endl; }
```

- Указатель, который может указывать на объекты любого типа (**небезопасно**)

```
void *ptr;
```

Целочисленный тип данных `int`

Модификаторы знака	
`signed`	Представление целого числа со знаком (**по умолчанию**)
`unsigned`	Представление целого числа без знака

Модификаторы размера	
`short`	**16-битное** представление целого числа
`long`	**32-битное** представление целого числа
`long long`	**64-битное** представление целого числа

Модификаторы знака и размера могут комбинироваться

Целочисленный тип данных `int`

Модификаторы знака	
`signed`	Представление целого числа со знаком (**по умолчанию**)
`unsigned`	Представление целого числа без знака

Модификаторы размера	
`short`	**16-битное** представление целого числа
`long`	**32-битное** представление целого числа
`long long`	**64-битное** представление целого числа

Модификаторы знака и размера могут комбинироваться

Q: Каков диапазон допустимых значений типа `long int`?

Целочисленный тип данных `int`

```
int x;
cout << "The size of x is " << sizeof(x) * 8 << " bits" << endl;

long int y;
cout << "The size of y is " << sizeof(y) * 8 << " bits" << endl;

short unsigned z;
cout << "The size of z is " << sizeof(z) * 8 << " bits" << endl;

int16_t t;
cout << "The size of t is " << sizeof(t) * 8 << " bits" << endl;
```

```
The size of x is 32 bits
The size of y is 32 bits
The size of z is 16 bits
The size of t is 16 bits
```

Символьные типы данных

`char`	**8-битное** представление символов
`char16_t`	Для поддержки представления 16 и 32-битных символов Unicode
`char32_t`	

```
char c1 = 'p’;

char c2 = 98;

char c3 = -85;

cout << c2 << '\n';

cout << c3;
```

```
b
л
```

Логический тип данных `bool`

Диапазон включает два значения: `true` (1) и `false` (0).

```
int k = 3;
if (k >= 2)
        cout << "yes";
else
        cout << "no";
```

yes

```
bool isEqual(int x, int y)
{
    return (x == y);
}
x = 2; y = 4;
cout << isEqual(x, y);
```

0

```
if (6)
        cout << “yes”
else
        cout << “no”
```

???

Типы данных с плавающей точкой C++

Тип	Минимальный размер
`float`	**32-битное** представление вещественных чисел
`double`	**64-битное** представление вещественных чисел
`long double`	

```
sizeof(long double) >= sizeof(double) >= sizeof(float)
```

Типы данных с плавающей точкой C++

```
double d = 1 / 10;
double r = 0.1;

if (d == r)
        cout << “equal”;
else
        cout << “not equal”;
```

???

```
double d = 1.0;
double r = 0.1 + 0.1 + 0.1 +
0.1 + 0.1 + 0.1 + 0.1 + 0.1 +
0.1 + 0.1;

if (d == r)
        cout << “equal”;
else
        cout << “not equal”;
```

???

Типы данных с плавающей точкой C++

```
double d = 1 / 10;
double r = 0.1;

if (d == r)
        cout << “equal”;
else
        cout << “not equal”;
```

```
not equal
```

```
double d = 1.0;
double r = 0.1 + 0.1 + 0.1 +
0.1 + 0.1 + 0.1 + 0.1 + 0.1 +
0.1 + 0.1;

if (d == r)
        cout << “equal”;
else
        cout << “not equal”;
cout << setprecision(17);
cout << d << endl;
cout << r << endl;
```

```
not equal
1
0.99999999999999989
```

Типы данных с плавающей точкой C++

Сравнение чисел с ограниченной точностью

```
double d = 1.0;
double r = 0.1 + 0.1 + 0.1 +
0.1 + 0.1 + 0.1 + 0.1 + 0.1 +
0.1 + 0.1;
double epsilon = 0.0001;

if (fabs(d - r) < epsilon)
        cout << “equal”;
else
        cout << “not equal”;
```

```
equal
```

Типы данных с плавающей точкой C++

Бесконечности и неопределенности

```
double zero = 0.0;
double pinf = 1.0 / zero;
double ninf = -5.0 / zero;
double nan = zero / zero;
double nan1 = sqrt(-6);

cout << pinf << endl;
cout << ninf << endl;
cout << nan << endl;
cout << nan1 << endl;
```

```
inf
-inf
-nan(ind)
-nan(ind)
```

Структура программы на C++

Документация

- Общие пояснения к программе в комментариях

Ссылки

- Подключение заголовочных файлов и пространств имен

Определения

- Определение пользовательских типов, констант, `#define`

Глобальные объявления

- Объявление переменных, классов, структур, которые доступны до окончания работы программы

Пользовательские функции

Точка входа компилятора

- Функция `main() { … }`

Структура программы на C++

Вычисление факториала

```
/*  Программа итерационного вычисления факториала */

#include <iostream>
using namespace std;

#define msg "FACTORIAL\n"
typedef int k;

k num = 0, fact = 1, value = 0;

k factorial(k& num) {
    for (k i = 1; i <= num; i++) {
        fact *= i;
    }
    return fact;
}

int main() {
    k Num = 5;
    value = factorial(Num);
    cout << msg;
    cout << Num << "! = “ << value << endl;
    return 0;
}
```

Структура программы на C++

Вычисление факториала

```
/*  Программа итерационного вычисления факториала */
#include <iostream>
using namespace std;
#define msg "FACTORIAL\n"
typedef int k;
k num = 0, fact = 1, value = 0;
k factorial(k& num) {
    for (k i = 1; i <= num; i++) {
        fact *= i;
    }
    return fact;
}
int main() {
    k Num = 5;
    value = factorial(Num);
    cout << msg;
    cout << Num << "! = “ << value << endl;
    return 0;
}
```

Цель программы

Структура программы на C++

Вычисление факториала

```
/*  Программа итерационного вычисления факториала */

#include <iostream>
using namespace std;

#define msg "FACTORIAL\n"
typedef int k;

k num = 0, fact = 1, value = 0;

k factorial(k& num) {
    for (k i = 1; i <= num; i++) {
        fact *= i;
    }
    return fact;
}

int main() {
    k Num = 5;
    value = factorial(Num);
    cout << msg;
    cout << Num << "! = “ << value << endl;
    return 0;
}
```

Подключение стандартных средств поддержки ввода-вывода и соответствующего пространства имен

Структура программы на C++

Вычисление факториала

```
/*  Программа итерационного вычисления факториала */

#include <iostream>
using namespace std;

#define msg "FACTORIAL\n"
typedef int k;

k num = 0, fact = 1, value = 0;

k factorial(k& num) {
    for (k i = 1; i <= num; i++) {
        fact *= i;
    }
    return fact;
}

int main() {
    k Num = 5;
    value = factorial(Num);
    cout << msg;
    cout << Num << "! = “ << value << endl;
    return 0;
}
```

Создание макроса `msg` и псевдонима для стандартного типа `int`

Структура программы на C++

Вычисление факториала

```
/*  Программа итерационного вычисления факториала */

#include <iostream>
using namespace std;

#define msg "FACTORIAL\n"
typedef int k;

k num = 0, fact = 1, value = 0;

k factorial(k& num) {
    for (k i = 1; i <= num; i++) {
        fact *= i;
    }
    return fact;
}

int main() {
    k Num = 5;
    value = factorial(Num);
    cout << msg;
    cout << Num << "! = “ << value << endl;
    return 0;
}
```

Объявление глобальных переменных `num`, `fact`, `value` типа `k`

Структура программы на C++

Вычисление факториала

```
/*  Программа итерационного вычисления факториала */

#include <iostream>
using namespace std;

#define msg "FACTORIAL\n"
typedef int k;

k num = 0, fact = 1, value = 0;

k factorial(k& num) {
    for (k i = 1; i <= num; i++) {
        fact *= i;
    }
    return fact;
}

int main() {
    k Num = 5;
    value = factorial(Num);
    cout << msg;
    cout << Num << "! = " << value << endl;
    return 0;
}
```

Определение функции, которая вычисляет факториал и возвращает его значение в глобальную переменную `fact`

Структура программы на C++

Вычисление факториала

```
/*  Программа итерационного вычисления факториала */

#include <iostream>
using namespace std;

#define msg "FACTORIAL\n"
typedef int k;

k num = 0, fact = 1, value = 0;

k factorial(k& num) {
    for (k i = 1; i <= num; i++) {
        fact *= i;
    }
    return fact;
}

int main() {
    k Num = 5;
    value = factorial(Num);
    cout << msg;
    cout << Num << "! = “ << value << endl;
    return 0;
}
```

Функция `main()`, которая возвращает 0 в результате успешного вычисления факториала `Num`

Структура программы на C++

Вычисление факториала

```
/*  Программа итерационного вычисления факториала */

#include <iostream>
using namespace std;

#define msg "FACTORIAL\n"
typedef int k;

k num = 0, fact = 1, value = 0;

k factorial(k& num) {
    for (k i = 1; i <= num; i++) {
        fact *= i;
    }
    return fact;
}

int main() {
    k Num = 5;
    value = factorial(Num);
    cout << msg;
    cout << Num << "! = “ << value << endl;
    return 0;
}
```

Функция `main()`, которая возвращает 0 в результате успешного вычисления факториала `Num`

Заголовочные файлы (*.h)

Предварительное объявление функции add (прототип)

prog1.cpp

```
#include <iostream>

int add(int x, int y);

int main()
{
    std::cout << add(5,6);
    return 0;
}
```

sum.cpp

```
int add(int x, int y)
{
    return x + y;
}
```

Заголовочные файлы (*.h)

sum.cpp

```
int add(int x, int y)
{
    return x + y;
}
```

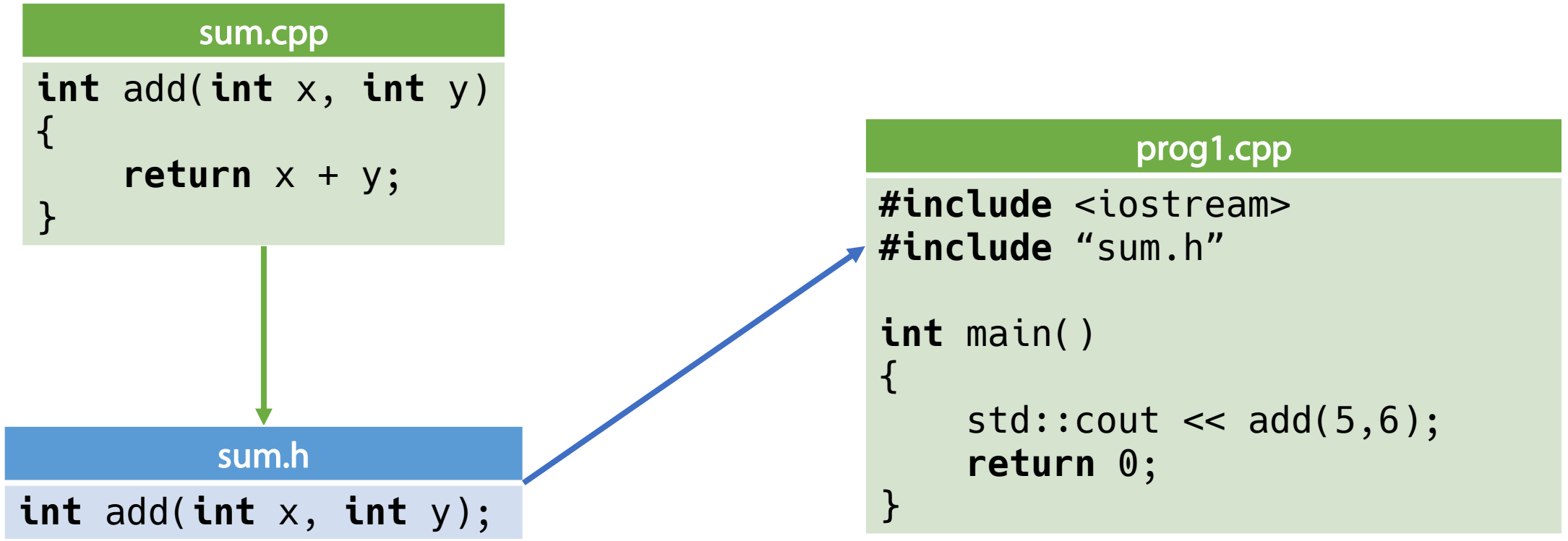
division.cpp

```
int add(int x, int y);

double frac(double x, double y)
{
    return add(x, y) / (x - y);
}
```

prog1.cpp

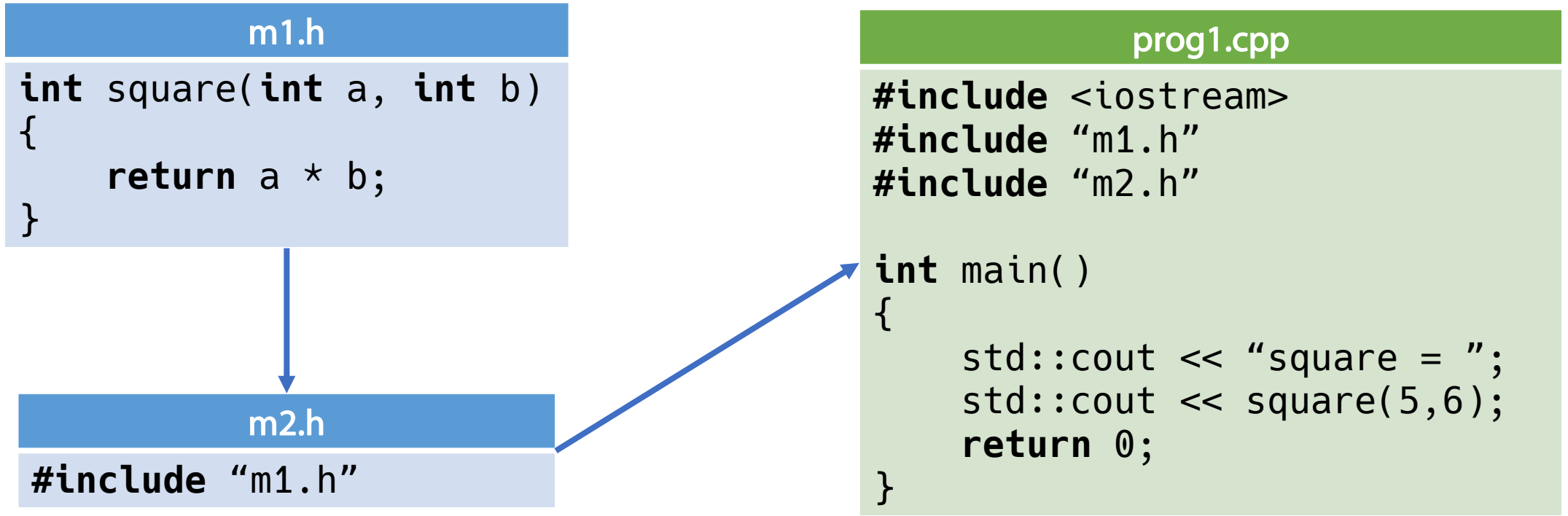
```
#include <iostream>

int add(int x, int y);
double frac(double x, double y);

int main()
{
    std::cout << add(5,6);
    std::cout << frac(4,3);
    return 0;
}
```

Заголовочные файлы (*.h)



Заголовочные файлы (*.h)

Проблема дублирования объявлений



Заголовочные файлы (*.h)

Проблема дублирования объявлений – header guards

m1.h

```
#ifndef M1_H
#define M1_H
int square(int a, int b)
{
    return a * b;
}
#endif
```

m2.h

```
#include “m1.h”
```

prog1.cpp

```
#include <iostream>
#include “m1.h”
#include “m2.h”

int main()
{
    std::cout << “square = ”;
    std::cout << square(5,6);
    return 0;
}
```

Заголовочные файлы (*.h)

Проблема дублирования объявлений – header guards

m1.h

```
#pragma once
int square(int a, int b)
{
    return a * b;
}
```

m2.h

```
#include “m1.h”
```

prog1.cpp

```
#include <iostream>
#include “m1.h”
#include “m2.h”

int main()
{
    std::cout << “square = ”;
    std::cout << square(5,6);
    return 0;
}
```

Заголовочные файлы (*.h)

- Использовать директивы препроцессора
- Избегать определения переменных и функций
- Придерживаться соответствия имен заголовочных файлов и файлов исходного кода (`sum.h` ↔ `sum.cpp`)
- Избегать подключения одних заголовочных файлов из других
- Не подключать файлы исходного кода с помощью `#include`

Пространства имен (namespaces)

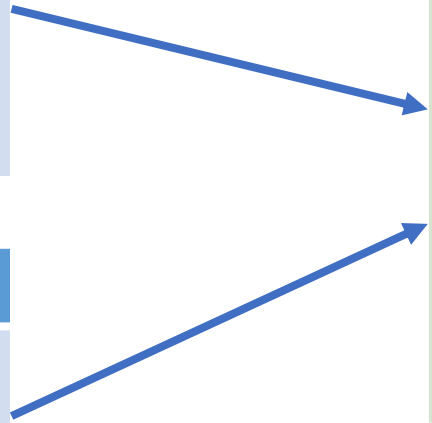
Конфликт имен

sumr.h

```
int doIt(int a, int b)
{
    return a + b;
}
```

subtractr.h

```
int doIt(int a, int b)
{
    return a - b;
}
```



prog1.cpp

```
#include “sumr.h”
#include “subtract.h”

int main()
{
    int sum = doIt(5,-8);
    return 0;
}
```

Какую версию вызвать?

Пространства имен (namespaces)

sumr.h

```
namespace sumr {
    int doIt(int a, int b)
    {
        return a + b;
    }
}
```

subtractr.h

```
namespace subtractr {
    int doIt(int a, int b)
    {
        return a - b;
    }
}
```

prog1.cpp

```
#include “sum.h”
#include “subtract.h”

int main()
{
    int s = doIt(5,-8);
    return 0;
}
```

Без указания пространства имен компилятор не находит определение `doIt`

Пространства имен (namespaces)

sumr.h

```
namespace sumr {
    int doIt(int a, int b)
    {
        return a + b;
    }
}
```

subtract.h

```
namespace subtractr {
    int doIt(int a, int b)
    {
        return a - b;
    }
}
```

prog1.cpp

```
#include “sum.h”
#include “subtract.h”

int main()
{
    int s;
    s = sumr::doIt(5,-8);
    return 0;
}
```

Доступ к функции `doIt` из простанства имен `sumr` с помощью оператора разрешения области видимости `::`

Пространства имен (namespaces)

sumr.h

```
namespace sumr {
    int doIt(int a, int b)
    {
        return a + b;
    }
}
```

subtract.h

```
namespace subtractr {
    int doIt(int a, int b)
    {
        return a - b;
    }
}
```

prog1.cpp

```
#include “sum.h”
#include “subtract.h”

using sumr::doIt;

int main()
{
    int s;
    s = doIt(5,-8);
    return 0;
}
```

using-объявление говорит, что используется doIt из пространства sumr

Пространства имен (namespaces)

sumr.h

```
namespace sumr {
    int doIt(int a, int b)
    {
        return a + b;
    }
}
```

subtract.h

```
namespace subtractr {
    int doIt(int a, int b)
    {
        return a - b;
    }
}
```

prog1.cpp

```
#include “sum.h”
#include “subtract.h”

using namespace sumr;

int main()
{
    int s;
    s = doIt(5,-8);
    return 0;
}
```

using-директива говорит, что должны быть подключены все имена из пространства sumr

Пространства имен (namespaces)

- Пространство имен – это область кода, внутри которой гарантируется уникальность используемых идентификаторов
- Пространство имен может быть описано в нескольких файлах или в разных местах одного файла
- Пространства имен могут быть вложены друг в друга, но этого лучше избегать

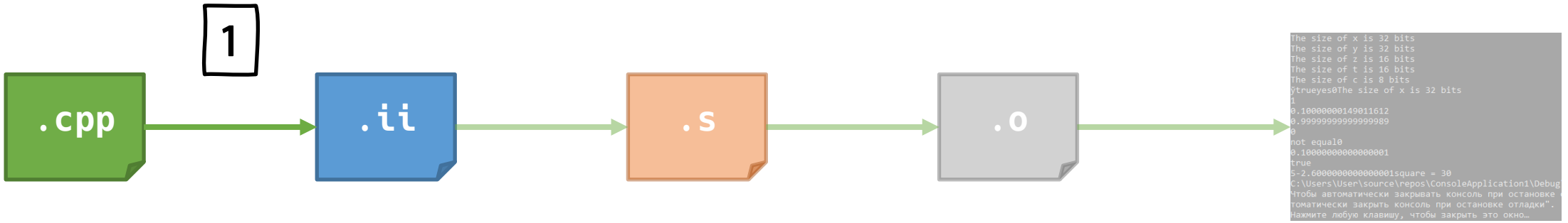
Стандартные библиотеки функций

Находятся в пространстве имен `std`

Заголовочные файлы	Назначение
`iostream, iomanip, fstream, …`	Ввод-вывод, форматирование
`string`	Работа со строками
`math`	Математические операции и функции
`complex`	Функции для работы с комплексными числами
`cstdlib`	Функции общего назначения
…	…
`algorithm, bitset, map, queue, …`	Стандартная библиотека шаблонов (STL)

Компиляция программы на C++

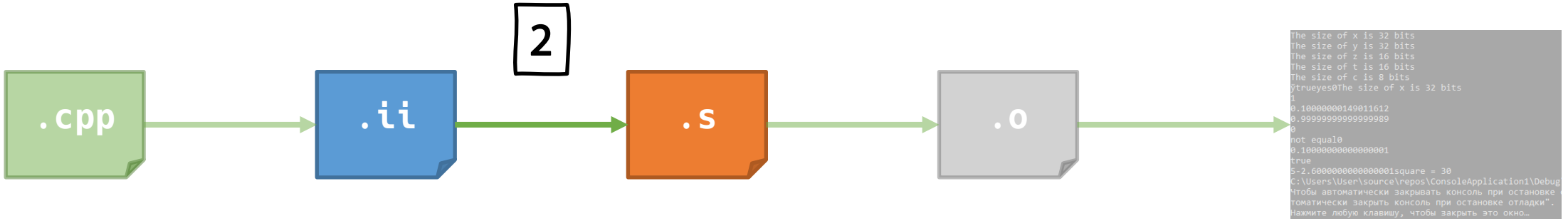
Компиляция программы на C++



Препроцессинг `g++ -E prog.cpp -o prog.ii`

- Включение заголовочных файлов в код (**`#include`**)
- Макроподстановки (**`#define`**)
- Выбор фрагментов кода (**`#if`**, **`#ifndef`**, **`#ifdef`**)

Компиляция программы на C++

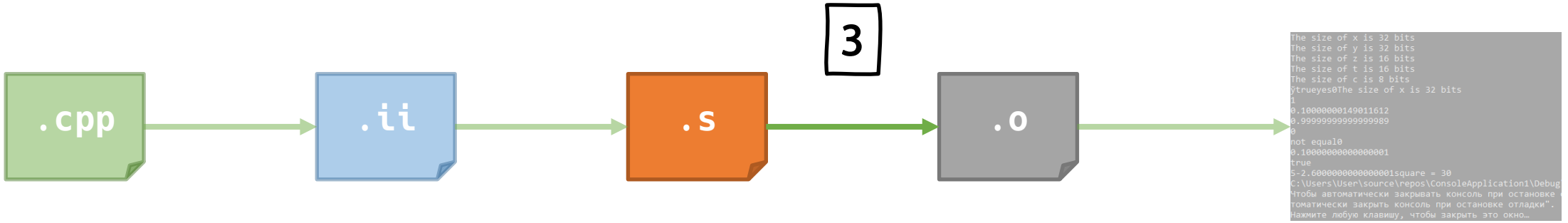


Компиляция `g++ -S prog.ii -o prog.s`

Преобразование кода без директив в *ассемблерный код.*

Промежуточное представление между ЯП и машинными кодами.

Компиляция программы на C++

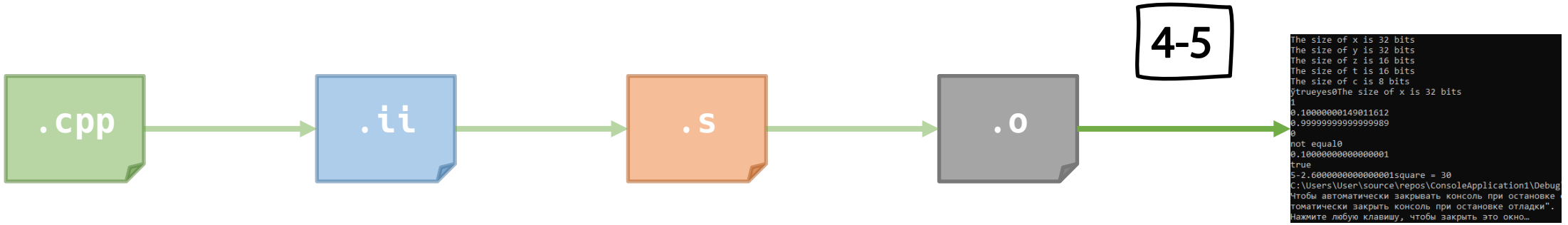


Ассемблирование

Преобразование ассемблерного кода в *машинный код.*
Машинные коды сохраняются в объектном файле.

```
as prog.s -o prog.o
```

Компиляция программы на C++



Компоновка и загрузка

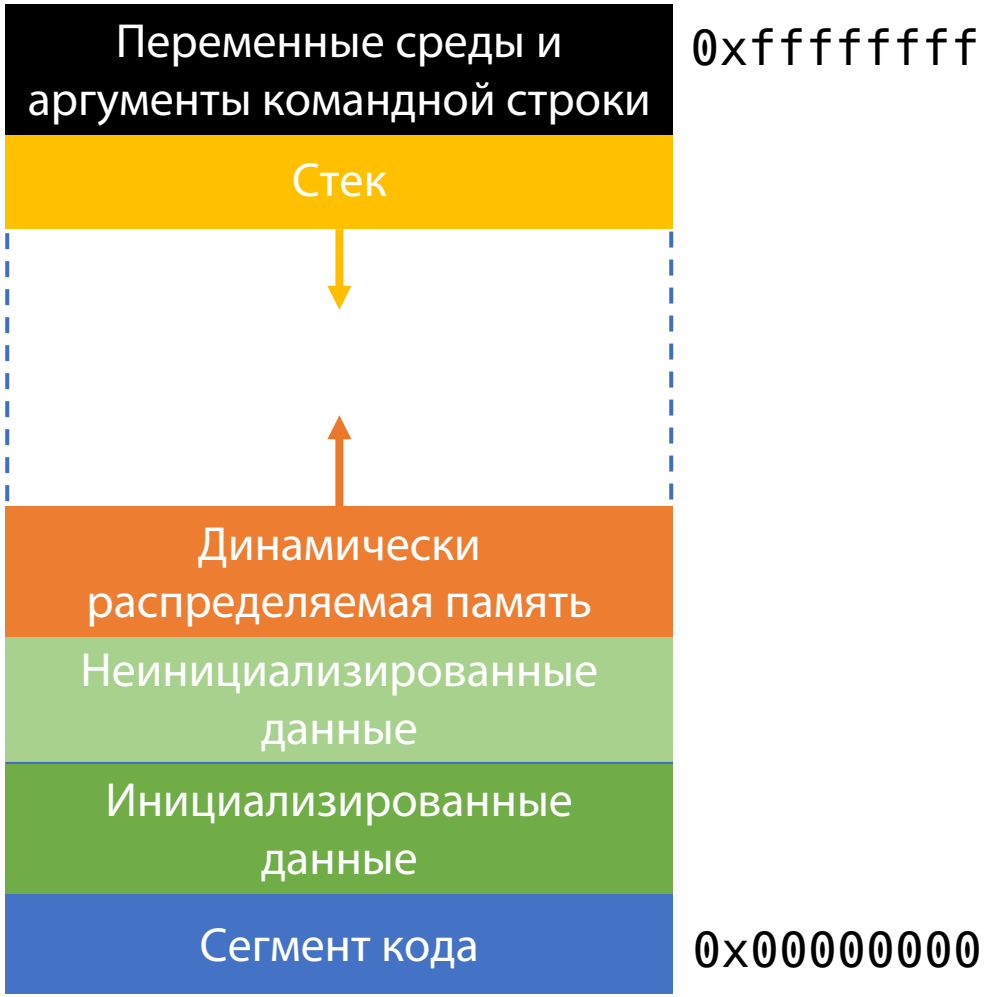
```
g++ prog.o -o prog
```

Связывание всех объектных файлов в единый исполняемый файл, который может быть загружен в память.

Связывание управляется *таблицей символов*.

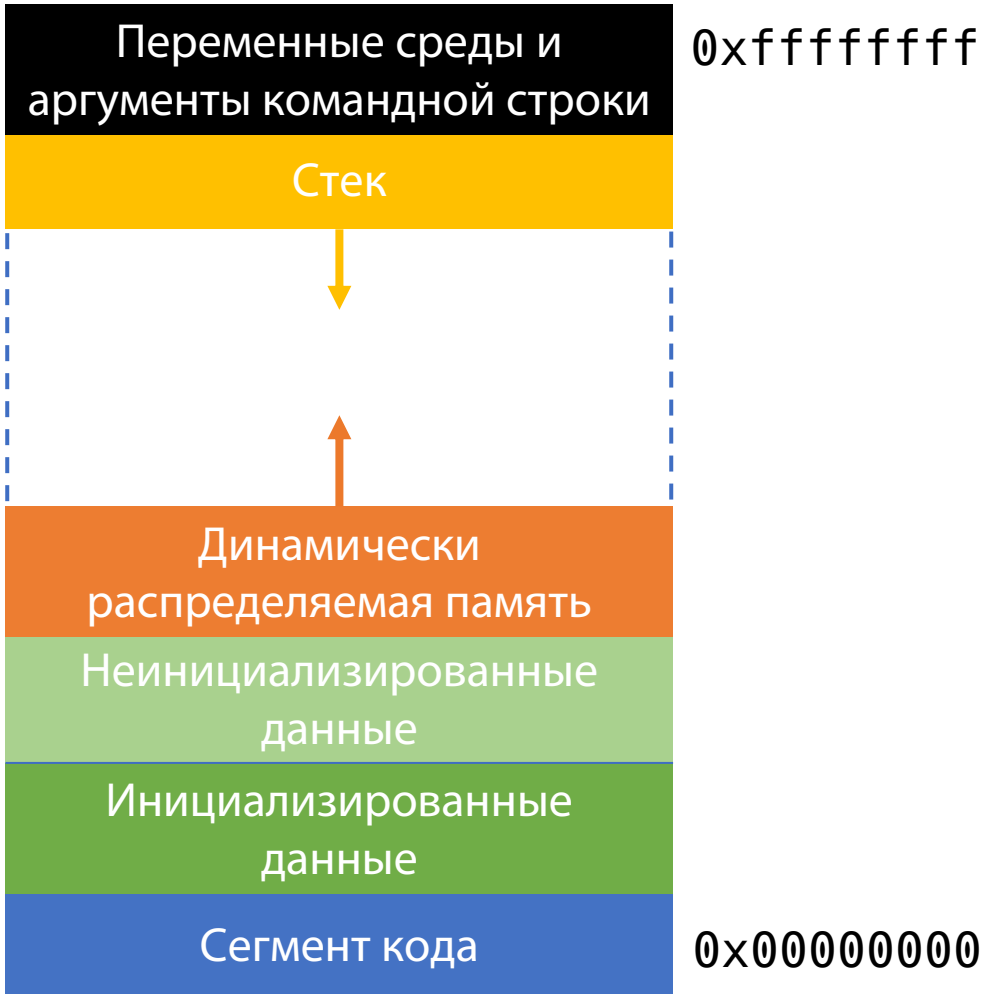
Организация программы в памяти

```
#include <iostream>

int var1;
int var2 = 10;

void function1()
{
    cout << “I am function!”;
}

int main()
{
   function1();
   return 0;
}
```



Организация программы в памяти

```
g++ test.cpp -o memory-layout

size memory-layout
text    data     bss     dec     hex filename
3345    2188     480    6013    177d memory-layout
```



Потоки ввода и вывода `<iostream>`

- **`cin`** – класс, связанный со стандартным **вводом** (клавиатура)
- **`cout`** – класс, связанный со стандартным **выводом** (экран)
- **`cerr`** - класс, связанный со стандартным выводом сообщений об **ошибках** (экран)

Потоки ввода и вывода <iostream>

```
#include <iostream>
#include <cstdlib> // для экстренного завершения программы

int main() {
    std::cout << “Enter positive number: ” << std::endl;

    int x;
    std::cin >> x;

    if (x <= 0) {
        std::cerr << “You entered a negative number!\n”;
        exit(1);
    }

    std::cout << “Terminated correctly” << std:endl;
    return 0;
}
```

```
Enter positive number:
-9
You entered a negative number!


Enter positive number:
8
Terminated correctly
```

Потоки ввода и вывода <iostream>

```
#include <iostream>
#include <cstdlib> // для экстренного завершения программы

int main() {
    std::cout << “Enter positive number: ” << std::endl;

    int x;
    std::cin >> x;

    if (x <= 0) {
        std::cerr << “You entered a negative number!\n”;
        exit(1);
    }

    std::cout << “Terminated correctly” << std::endl;
    return 0;
}
```

```
Enter positive number:
-9
You entered a negative number!


Enter positive number:
8
Terminated correctly
```

Форматирование вывода

- **Флаги** – логические переменные, которые определяют формат вывода (включение `setf`, отключение `unsetf`)
- **Манипуляторы** – объекты, помещаемые в поток вывода и изменяющие формат вывода (автоматически включают и отключают флаги)

Форматирование вывода

Вывод в разных системах счисления

```
#include <iostream>

int main() {
    std::cout << std::hex << 128 << std::endl;
    std::cout << 679 << std::endl;
    std::cout << std::oct << 65536 << std::endl;
    return 0;
}
```

```
80
2a7
200000
```

Вывод логических значений

```
#include <iostream>

int main() {
    int x = 4; int y = 6;
    std::cout << (x < y) << std::endl;
    std::cout << std::boolalpha << (y < x);
    return 0;
}
```

```
1
false
```

Манипулятор действует до тех пор, пока в поток вывода не помещен другой манипулятор

Форматирование вывода

Задание точности

```
#include <iostream>

int main() {
    double k = 0.1;

    std::cout << std::fixed;
    std::cout << std::setprecision(9);
    std::cout << k << std::endl;
    std::cout << std::setprecision(17);
    std::cout << k << std::endl;

    return 0;
}
```

```
0.100000000
0.10000000000000001
```

Экспоненциальная запись

```
#include <iostream>

int main() {
    double k = 0.1 / 0.3;

    std::cout << std::scientific;
    std::cout << std::setprecision(9);
    std::cout << k << std::endl;
    std::cout << std::setprecision(17);
    std::cout << k << std::endl;

    return 0;
}
```

```
3.333333333e-01
3.3333333333333370e-01
```

Файловый ввод ifstream

```
#include <fstream>
#include <iostream>
#include <string>

using namespace std;

int main()
{
    ifstream inpFile("test.txt");

    while (inpFile)
    {
        string str;
        getline(inpFile, str);
        cout << str << '\n';
    }

    retrun 0;
}
```

test.txt

The first line of a file
The second line of a file

Файловый вывод ofstream

Запись в файл

```
#include <fstream>
#include <iostream>
#include <string>

using namespace std;

int main()
{
    ofstream outFile("test.txt");

    outFile << “Hello\n”;
    outFile << “new line\n”;
    outFile.close();
    return 0;
}
```

Файл полностью перезаписывается

Добавление в файл без перезаписи

```
#include <fstream>
#include <iostream>

using namespace std;

int main()
{
    ofstream outFile("test.txt“, ios::app);

    outFile << “some\n”;
    outFile << “new lines\n”;
    outFile.close();
    return 0;
}
```

Строки добавляются в файл

Структуры `struct`

Объявление, определение и инициализация структур

```
struct Employee
{
    string name;
    int age;
    double salary;
};
```

```
Employee Nick;
Nick.age = 25;
Nick.name = “Nick Wellington”;
Nick.salary = 1250.69;

Employee John{“John Philips”, 46};

Employee Pete;
Pete = {“Pete Jenkins”, 36, 1256.45};
```

От класса структуру отличает только открытый доступ к ее членам (по умолчанию). Структура может содержать конструкторы и методы.

Структуры struct

Вложенные структуры и передача структуры в функцию

```
struct Date
{
    int day;
    int month;
    int year;
};

struct Employee
{
    string name;
    Date birthDay;
    double salary;
};
```

```
int calcAge(Employee x)
{
    return 2021 - x.birthDay.year;
}

int main()
{
    Employee Fred;
    Fred = {“Fred Williams”, {12, 3, 1989}, 3450};

    cout << “Fred’s Age is ” << calcAge(Fred);
    return 0;
}
```

Структуры `struct`

Назначенная инициализация структуры (C++20)

```
struct Date
{
    int day;
    int month;
    int year;
};

struct Employee
{
    string name;
    Date birthDay;
    double salary;
};
```

```
int main()
{
    Employee Fred;
    Fred = {“Fred Williams”, {12, 3, 1989}, 3450};

    Employee Rick;
    Rick = {.name=“Rick Hopkins”, .salary=“4591”};

    Employee Simon;
    Simon = {.birthDay{.day = 1, month = 12, year=1976}};

    return 0;
}
```

ООП и C++

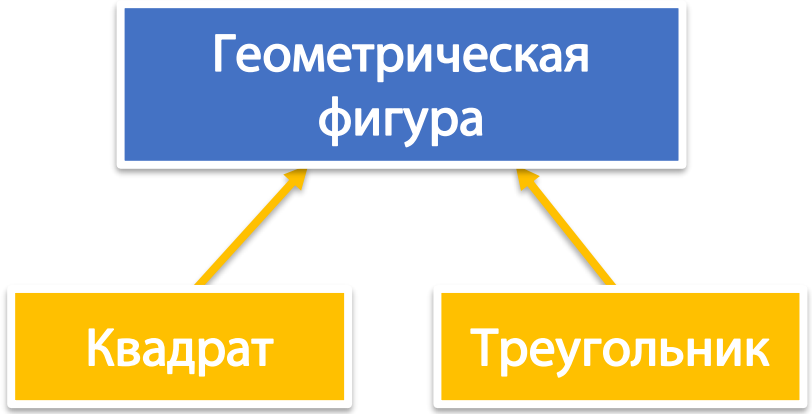
- Инкапсуляция
- Наследование
- Полиморфизм

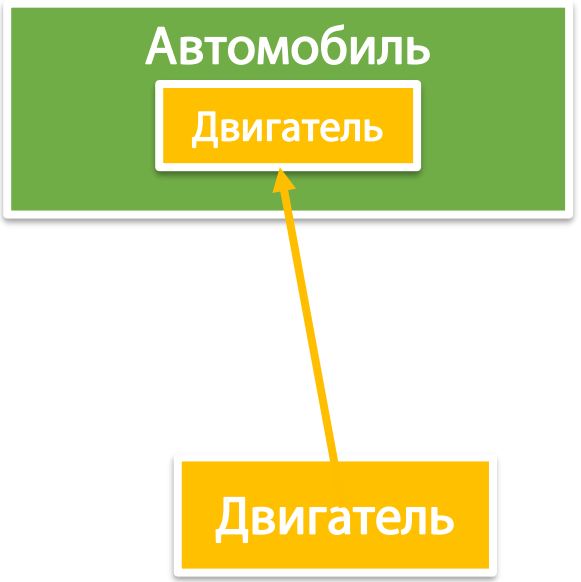
ООП и C++

- **Инкапсуляция**
- Наследование
- Полиморфизм

ООП и C++

- Инкапсуляция
- **Наследование**
- **Ассоциация**
- Полиморфизм

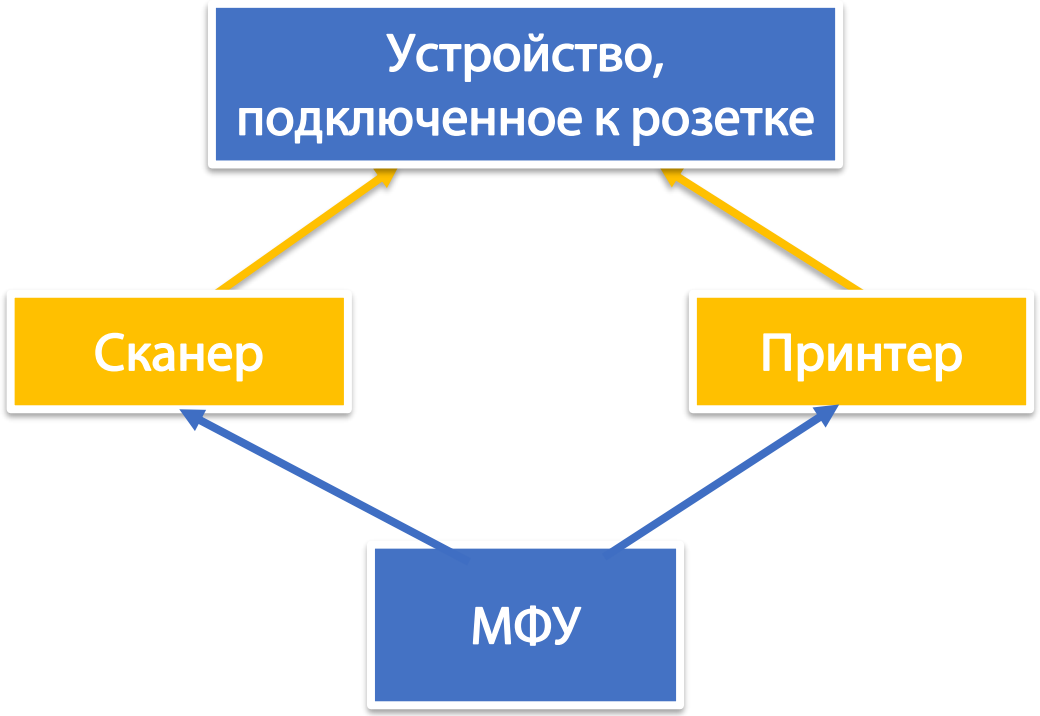




ООП и C++

- Инкапсуляция
- **Наследование**
- **Ассоциация**
- Полиморфизм

Множественное наследование

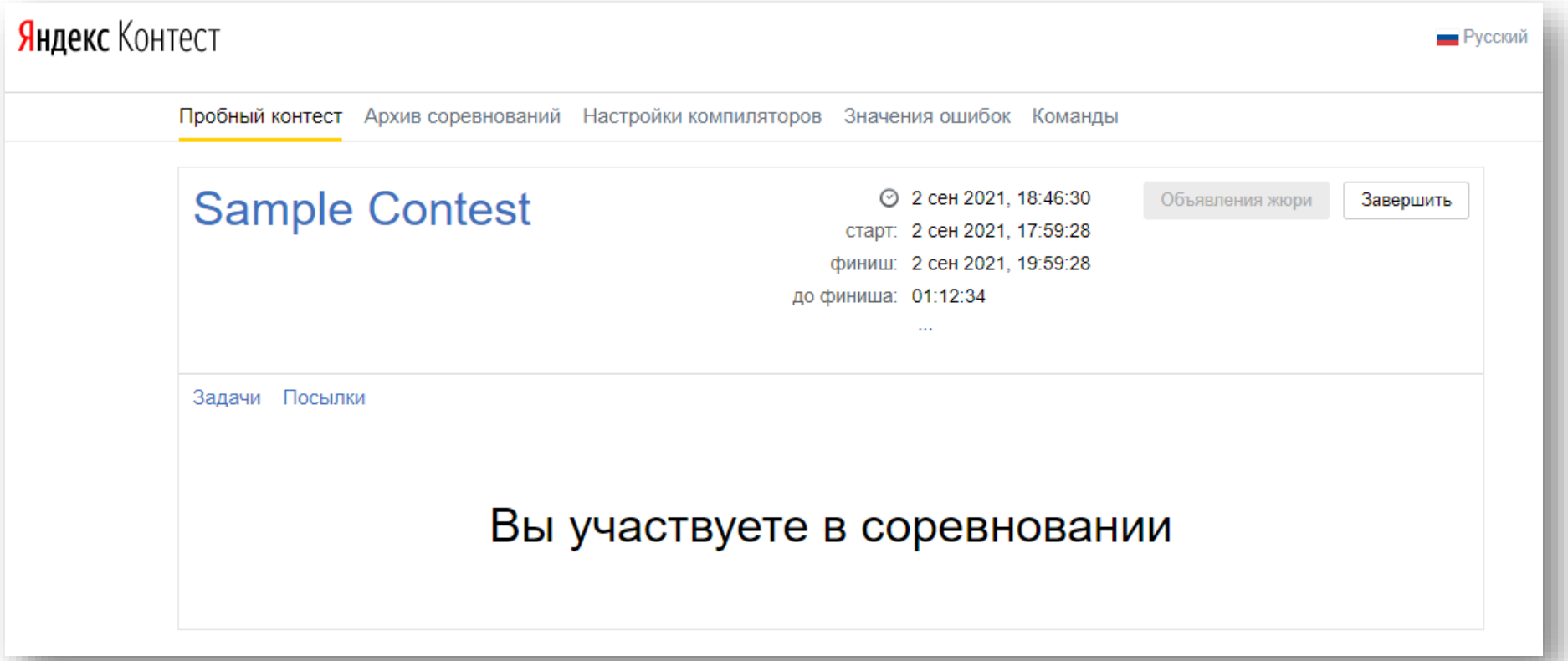


ООП и C++

- Инкапсуляция
- Наследование
- **Полиморфизм**

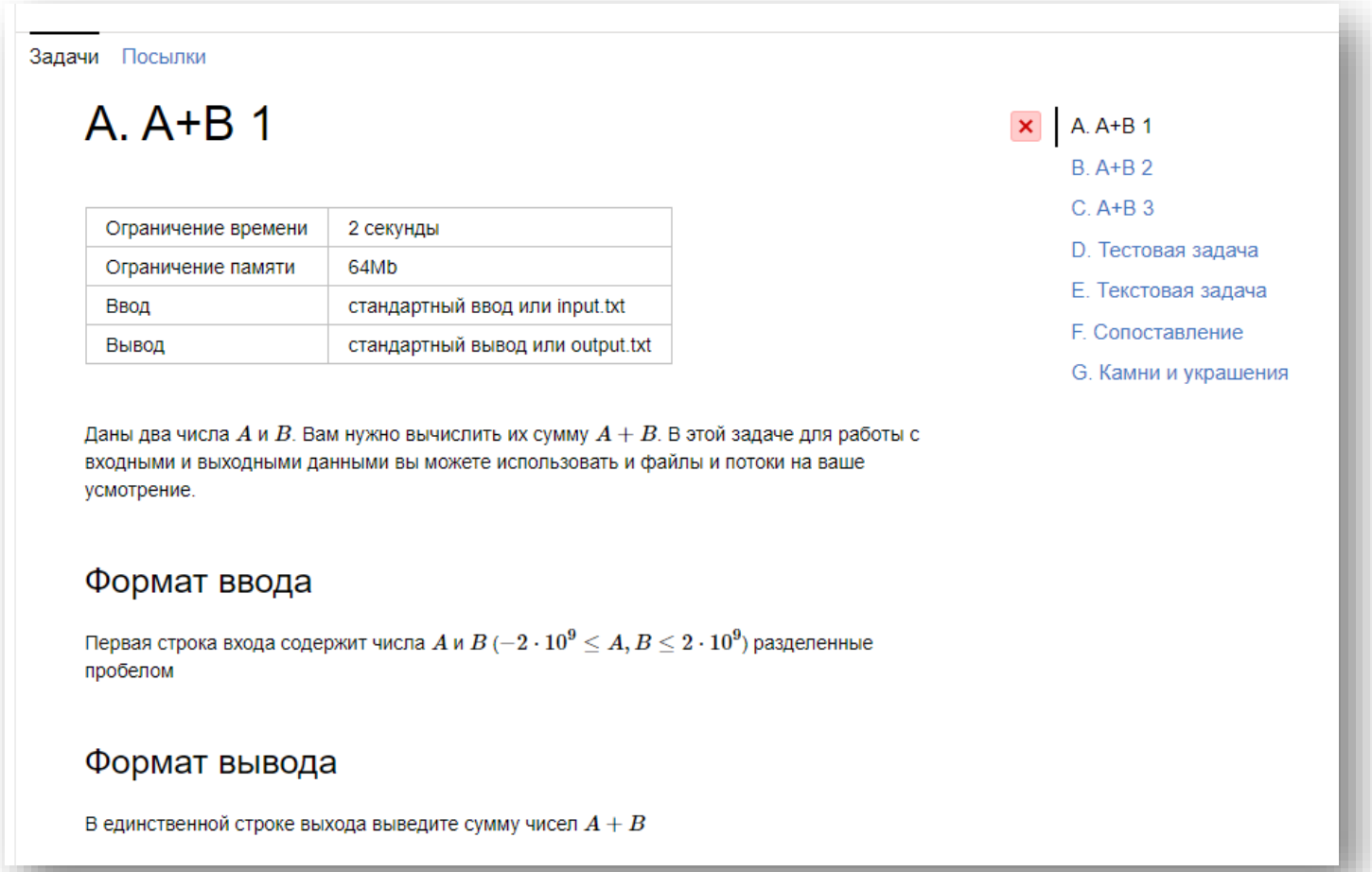
Яндекс.Контест

Система для автоматического тестирования программ



Яндекс.Контест

Система для автоматического тестирования программ



Задачи Посылки

A. A+B 1

Ограничение времени	2 секунды
Ограничение памяти	64Mb
Ввод	стандартный ввод или input.txt
Вывод	стандартный вывод или output.txt

Даны два числа A и B. Вам нужно вычислить их сумму $A + B$. В этой задаче для работы с входными и выходными данными вы можете использовать и файлы и потоки на ваше усмотрение.

Формат ввода

Первая строка входа содержит числа A и B ($-2 \cdot 10^9 \le A, B \le 2 \cdot 10^9$) разделенные пробелом

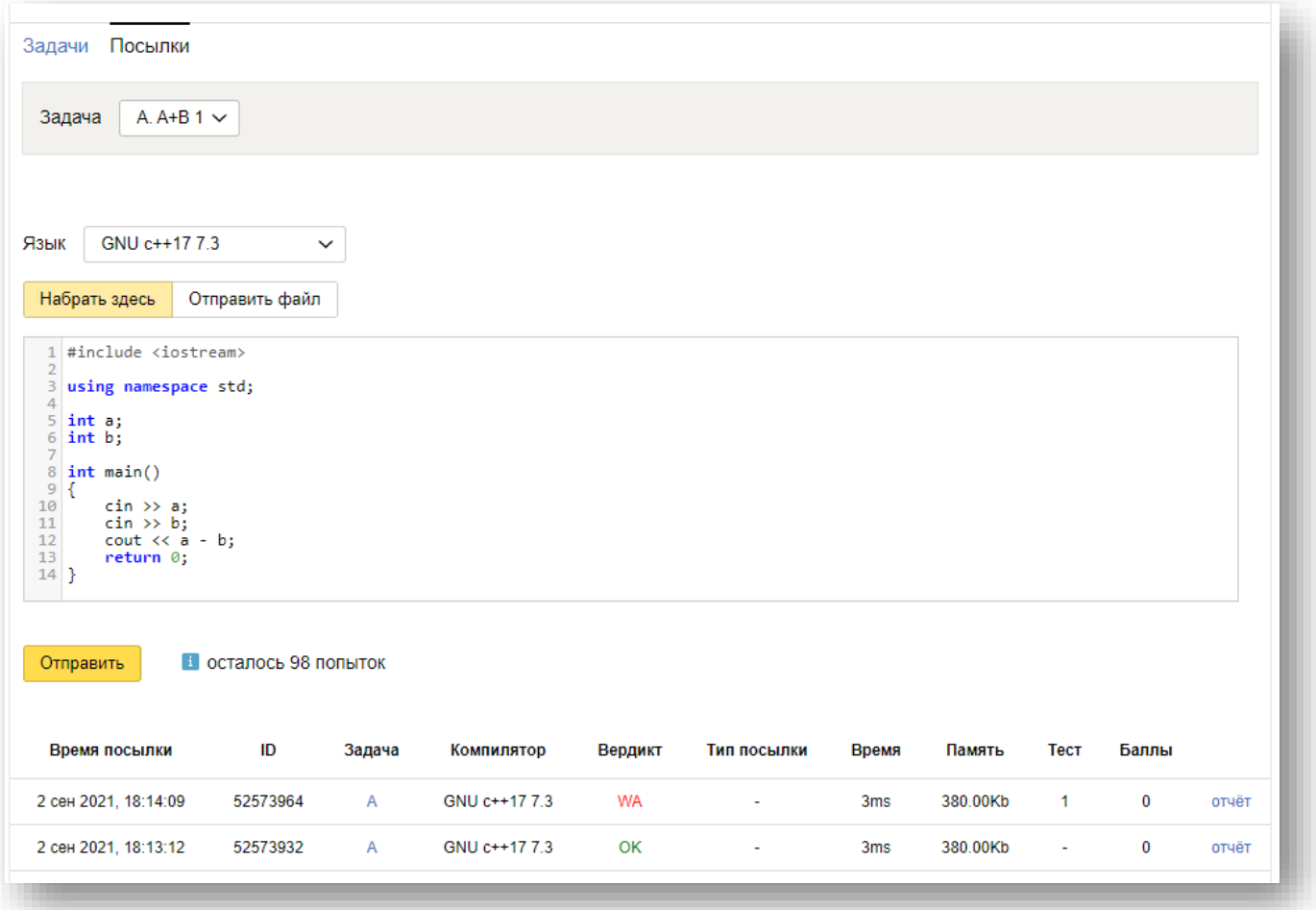
Формат вывода

В единственной строке выхода выведите сумму чисел $A + B$

A. A+B 1
B. A+B 2
C. A+B 3
D. Тестовая задача
E. Текстовая задача
F. Сопоставление
G. Камни и украшения

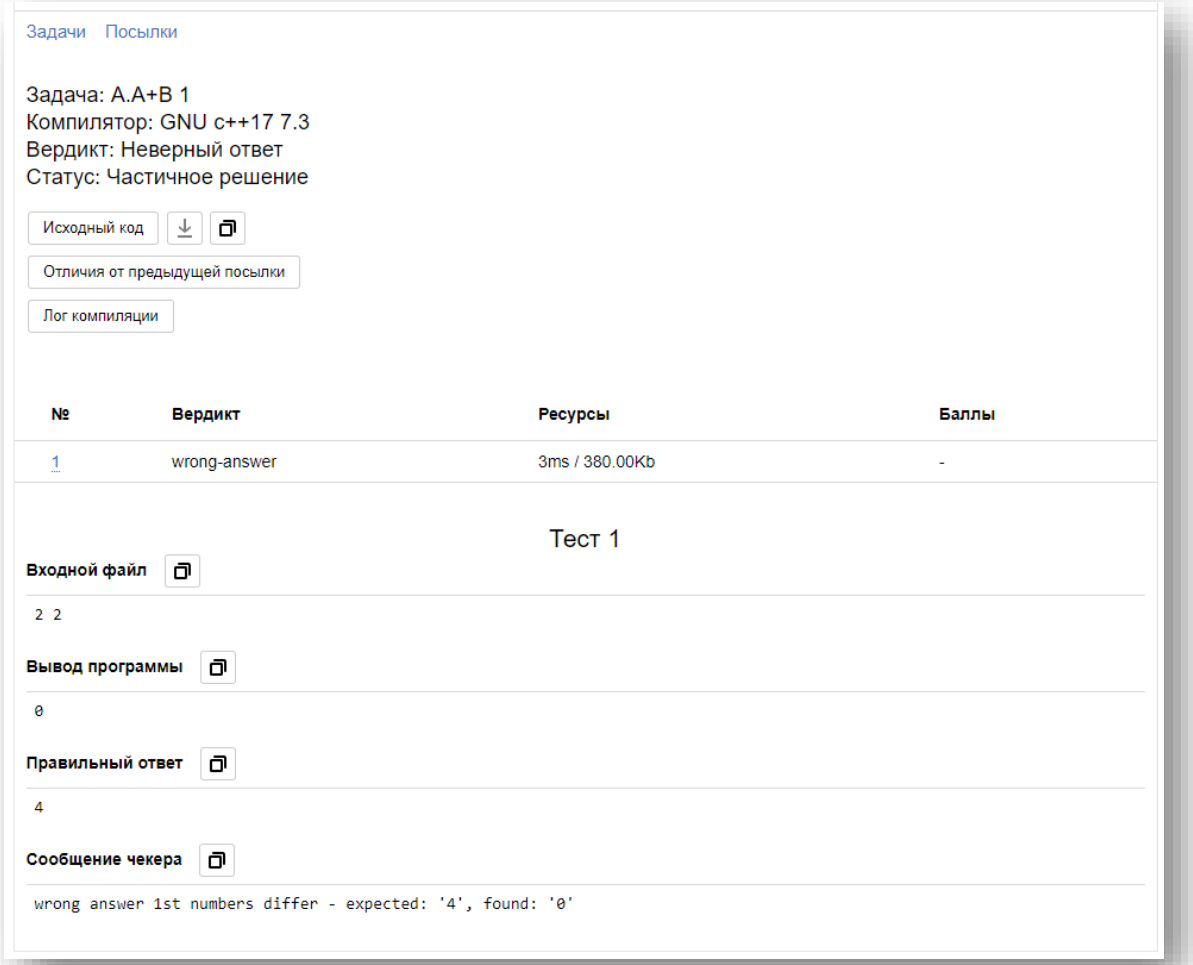
Яндекс.Контест

Система для автоматического тестирования программ



Яндекс.Контест

Система для автоматического тестирования программ



Яндекс.Контест

- Задачи текущей недели доступны для решения:
  с **11:30 пятницы** до **23:59 четверга (след. неделя)**
- Объявления о контестах публикуются в Telegram-канале

На следующем семинаре…

- Указатели и работа с динамической памятью
- Классы, объекты, методы
- Функции, механизмы передачи параметров
- Виртуальные функции и перегрузка операторов