Введение: язык C

Никита Гаевой (102)
Иван Казменко (101, 103)
Владислав Макаров (104)
Семён Петров (106)
Лиана Хазалия (105)

Санкт-Петербургский Государственный Университет

Четверг, 9 сентября 2021 года

Ссылка

Ссылка на материалы занятий:
`http://acm.math.spbu.ru/~gassa/bachelor-2021`

Проверка решений

Проверка решений

- Автоматическая проверяющая система: `http://acm.math.spbu.ru/tsweb`
- Логины и пароли разосланы на почту (?!)
- Submit: послать решение на проверку
- Monitor: кто что решил
- Clarifications: задать преподавателям вопрос по формальному условию задачи
- Submissions: предыдущие попытки и их результаты (`NO` → `OK`, `WA`, `TL`, ...
- Statements: условия задач
- List of all contests: предыдущие занятия
- О вердиктах: `http://acm.math.spbu.ru/testsys-errors.html`

Среда программирования

Логины и пароли

- Логин и пароль в классе: `stXXXXXX`
- Логин и пароль в системе проверки: `21mXXX`

Среда программирования

- Visual Studio (`visualstudio.microsoft.com`)
- CLion (`jetbrains.com/community/education`)
- CodeBlocks (`codeblocks.org`)
- online (не забудьте выбрать Secret или Private!): IdeOne (`ideone.com`)
- ...

Для тех, кто всё знает

Для тех, кто всё знает

- Можно сразу сдавать задачи
- Дополнительные задачи: соревнование «напишите короткий код» (на тех же задачах, языки — C и C++ без странных ограничений)

Что делать в среде

Что делать в среде

Например, в Visual Studio.

- Create empty C++ project
  - No precompiled headers!
- Create C++ file
- Add file to project
- Написать решение
- Поправить ошибки компиляции
  - `#define _CRT_SECURE_NO_WARNINGS`
- Run without debugging (`Ctrl+F5`), чтобы окно не закрывалось сразу при завершении работы
  - Лучше, чем каждый раз дописывать и стирать что-то
- Когда заработает, послать на проверку

Пример 1: задача

Задача

- В первой строке задано число n ($1 \leqslant n \leqslant 10\,000$).
- Во второй строке заданы целые числа $a_1, a_2, \ldots, a_n$, разделённые пробелами ($1 \leqslant a_i \leqslant 1000$).
- Выведите одно целое число: сумму $a_1 + a_2 + \ldots + a_n$.

Пример 1: решение

```c
#include <stdio.h>
int a [10000];
int main ()
{
    int n;
    scanf ("%d", &n);
    for (int i = 0; i < n; i++)
        scanf ("%d", &a[i]);
    int s = 0;
    for (int i = 0; i < n; i++)
        s += a[i];
    printf ("%d\n", s);
    return 0;
}
```

Ввод:

```
5
1 5 3 2 4
```

Вывод:

```
15
```

Пример 1: решение

```
 1  #include <stdio.h>
 2  int a [10000];
 3  int main ()
 4  {
 5      int n;
 6      scanf ("%d", &n);
 7      for (int i = 0; i < n; i++)
 8          scanf ("%d", &a[i]);
 9      int s = 0;
10      for (int i = 0; i < n; i++)
11          s += a[i];
12      printf ("%d\n", s);
13      return 0;
14  }
```

Ввод:

```
5
1 5 3 2 4
```

Вывод:

```
15
```

- 1: включаем стандартную библиотеку ввода-вывода

Пример 1: решение

```
 1  #include <stdio.h>
 2  int a [10000];
 3  int main ()
 4  {
 5      int n;
 6      scanf ("%d", &n);
 7      for (int i = 0; i < n; i++)
 8          scanf ("%d", &a[i]);
 9      int s = 0;
10      for (int i = 0; i < n; i++)
11          s += a[i];
12      printf ("%d\n", s);
13      return 0;
14  }
```

Ввод:

```
5
1 5 3 2 4
```

Вывод:

```
15
```

- 2: заводим массив — место в памяти для хранения 10 000 целых чисел, они называются `a[0]`, `a[1]`, ..., `a[9999]`

Пример 1: решение

```
 1  #include <stdio.h>
 2  int a [10000];
 3  int main ()
 4  {
 5      int n;
 6      scanf ("%d", &n);
 7      for (int i = 0; i < n; i++)
 8          scanf ("%d", &a[i]);
 9      int s = 0;
10      for (int i = 0; i < n; i++)
11          s += a[i];
12      printf ("%d\n", s);
13      return 0;
14  }
```

Ввод:

```
5
1 5 3 2 4
```

Вывод:

```
15
```

- 3: заголовок основной функции (`main`) в программе

Пример 1: решение

```
 1  #include <stdio.h>
 2  int a [10000];
 3  int main ()
 4  {
 5      int n;
 6      scanf ("%d", &n);
 7      for (int i = 0; i < n; i++)
 8          scanf ("%d", &a[i]);
 9      int s = 0;
10      for (int i = 0; i < n; i++)
11          s += a[i];
12      printf ("%d\n", s);
13      return 0;
14  }
```

Ввод:

```
5
1 5 3 2 4
```

Вывод:

```
15
```

- 4: всё, что внутри этих фигурных скобок — это тело функции `main`

Пример 1: решение

```
 1  #include <stdio.h>
 2  int a [10000];
 3  int main ()
 4  {
 5      int n;
 6      scanf ("%d", &n);
 7      for (int i = 0; i < n; i++)
 8          scanf ("%d", &a[i]);
 9      int s = 0;
10      for (int i = 0; i < n; i++)
11          s += a[i];
12      printf ("%d\n", s);
13      return 0;
14  }
```

Ввод:

```
5
1 5 3 2 4
```

Вывод:

```
15
```

- 5: заводим место в памяти для хранения целого числа n

Пример 1: решение

```
 1  #include <stdio.h>
 2  int a [10000];
 3  int main ()
 4  {
 5      int n;
 6      scanf ("%d", &n);
 7      for (int i = 0; i < n; i++)
 8          scanf ("%d", &a[i]);
 9      int s = 0;
10      for (int i = 0; i < n; i++)
11          s += a[i];
12      printf ("%d\n", s);
13      return 0;
14  }
```

Ввод:

```
5
1 5 3 2 4
```

Вывод:

```
15
```

- 6: вызов функции: читаем целое число в десятичной записи из стандартного потока ввода и записываем в n

Пример 1: решение

```
 1  #include <stdio.h>
 2  int a [10000];
 3  int main ()
 4  {
 5      int n;
 6      scanf ("%d", &n);
 7      for (int i = 0; i < n; i++)
 8          scanf ("%d", &a[i]);
 9      int s = 0;
10      for (int i = 0; i < n; i++)
11          s += a[i];
12      printf ("%d\n", s);
13      return 0;
14  }
```

Ввод:

```
5
1 5 3 2 4
```

Вывод:

```
15
```

- 7: цикл `for`: сначала заводим `int i = 0`, и пока `i < n`, выполняем тело цикла, а за ним `i++`

Пример 1: решение

```
 1  #include <stdio.h>
 2  int a [10000];
 3  int main ()
 4  {
 5      int n;
 6      scanf ("%d", &n);
 7      for (int i = 0; i < n; i++)
 8          scanf ("%d", &a[i]);
 9      int s = 0;
10      for (int i = 0; i < n; i++)
11          s += a[i];
12      printf ("%d\n", s);
13      return 0;
14  }
```

Ввод:

```
5
1 5 3 2 4
```

Вывод:

```
15
```

- 8: тело цикла `for`: для каждого $\texttt{i} = 0, 1, \ldots, \texttt{n} - 1$ читаем число и записываем в `a[i]`

Пример 1: решение

```
 1  #include <stdio.h>
 2  int a [10000];
 3  int main ()
 4  {
 5      int n;
 6      scanf ("%d", &n);
 7      for (int i = 0; i < n; i++)
 8          scanf ("%d", &a[i]);
 9      int s = 0;
10      for (int i = 0; i < n; i++)
11          s += a[i];
12      printf ("%d\n", s);
13      return 0;
14  }
```

Ввод:

```
5
1 5 3 2 4
```

Вывод:

```
15
```

- 9: после цикла заводим место в памяти для хранения целого числа `s` и кладём туда 0

Пример 1: решение

```
 1  #include <stdio.h>
 2  int a [10000];
 3  int main ()
 4  {
 5      int n;
 6      scanf ("%d", &n);
 7      for (int i = 0; i < n; i++)
 8          scanf ("%d", &a[i]);
 9      int s = 0;
10      for (int i = 0; i < n; i++)
11          s += a[i];
12      printf ("%d\n", s);
13      return 0;
14  }
```

Ввод:

```
5
1 5 3 2 4
```

Вывод:

```
15
```

- 10: такой же цикл `for`; если бы в цикле было больше действий, мы поставили бы вокруг них фигурные скобки

Пример 1: решение

```
 1  #include <stdio.h>
 2  int a [10000];
 3  int main ()
 4  {
 5      int n;
 6      scanf ("%d", &n);
 7      for (int i = 0; i < n; i++)
 8          scanf ("%d", &a[i]);
 9      int s = 0;
10      for (int i = 0; i < n; i++)
11          s += a[i];
12      printf ("%d\n", s);
13      return 0;
14  }
```

Ввод:

```
5
1 5 3 2 4
```

Вывод:

```
15
```

- 11: тело цикла: для каждого $\mathtt{i} = 0, 1, \ldots, \mathtt{n} - 1$ выполнить присваивание $\mathtt{s}_{\text{new}} \leftarrow \mathtt{s}_{\text{old}} + \mathtt{a[i]}$

Пример 1: решение

```
 1  #include <stdio.h>
 2  int a [10000];
 3  int main ()
 4  {
 5      int n;
 6      scanf ("%d", &n);
 7      for (int i = 0; i < n; i++)
 8          scanf ("%d", &a[i]);
 9      int s = 0;
10      for (int i = 0; i < n; i++)
11          s += a[i];
12      printf ("%d\n", s);
13      return 0;
14  }
```

Ввод:

```
5
1 5 3 2 4
```

Вывод:

```
15
```

- 12: вызов функции: пишем целое число `s` в десятичной записи и перевод строки в стандартный поток вывода

Пример 1: решение

```
 1  #include <stdio.h>
 2  int a [10000];
 3  int main ()
 4  {
 5      int n;
 6      scanf ("%d", &n);
 7      for (int i = 0; i < n; i++)
 8          scanf ("%d", &a[i]);
 9      int s = 0;
10      for (int i = 0; i < n; i++)
11          s += a[i];
12      printf ("%d\n", s);
13      return 0;
14  }
```

Ввод:

```
5
1 5 3 2 4
```

Вывод:

```
15
```

- 13: выход из `main`: число 0 говорит операционной системе, что наша программа завершила работу как задумано

Пример 1: решение

```
 1  #include <stdio.h>
 2  int a [10000];
 3  int main ()
 4  {
 5      int n;
 6      scanf ("%d", &n);
 7      for (int i = 0; i < n; i++)
 8          scanf ("%d", &a[i]);
 9      int s = 0;
10      for (int i = 0; i < n; i++)
11          s += a[i];
12      printf ("%d\n", s);
13      return 0;
14  }
```

Ввод:

```
5
1 5 3 2 4
```

Вывод:

```
15
```

- 14: конец функции `main`

Типы данных

- `int x = 45;` — 32-битные целые числа со знаком
  - значения от $-2\,147\,473\,648$ до $+2\,147\,473\,647$
  - `scanf ("%d", &x); printf ("%d", x);`
- `double v = 1.2345;` — вещественные числа
  - на самом деле двоично-рациональные
  - хранятся первые несколько цифр после старшей
  - `scanf ("%lf", &v); printf ("%.10lf", v);`
- `int arr [100] = {1, 2, 3};` — массивы
  - в памяти подряд лежат `arr[0]`, `arr[1]`, ..., `arr[99]`
  - `scanf ("%d", &arr[99]); printf ("%d", arr[99]);`
- `char s [101] = "abc";` — строки
  - после всех символов строки записан `'\0'` (символ с кодом 0)
  - `scanf ("%s", s); printf ("%s", s);`
- `int a [100] [50] = {{0}};` — многомерные массивы
  - в памяти подряд лежат `a[0][0]`, `a[0][1]`, ..., `a[0][49]`, затем `a[1][0]`, `a[1][1]`, и так далее до `a[99][49]`
  - `scanf ("%d", &a[1][2]); printf ("%d", a[3][4]);`

Конструкции языка

- `int x, y = 45, z [10];` — объявление переменных
- `x = y - z[i] / t;` — выражения
- `if (x == 5 && y < z[i]) {...} else {...}` — условный оператор
  - `if (x != 5 || y >= z[i]) {...}` — можно и без `else`
- `while (cond) {...}` — цикл `while`
- `for (start; cond; step) {...}` — цикл `for`
  - эквивалент: `start; while (cond) {... step;}`
- `do {...} while (cond);` — цикл с пост-условием
- `x = fun (arg1, arg2);` — вызов функции
- `int fun (int param1, int param2) {...}` — определение функции

Пример 2: алгоритм Евклида

- Задача: найдите наибольший общий делитель двух заданных чисел.

Пример 2: алгоритм Евклида

- Задача: найдите наибольший общий делитель двух заданных чисел.

```
1  int gcd (int a, int b)
2  {
3      if (a == 0)
4          return b;
5      return gcd (b % a, a);
6  }
```

Пример 2: алгоритм Евклида

- Задача: найдите наибольший общий делитель двух заданных чисел.

```
1  int gcd (int a, int b)
2  {
3      if (a == 0)
4          return b;
5      return gcd (b % a, a);
6  }
```

Как работает вызов `gcd (7, 5)`?

Пример 2: алгоритм Евклида

- Задача: найдите наибольший общий делитель двух заданных чисел.

```
1  int gcd (int a, int b)
2  {
3      if (a == 0)
4          return b;
5      return gcd (b % a, a);
6  }
```

Как работает вызов `gcd (7, 5)`?

`gcd (a:7, b:5) line 3`

Пример 2: алгоритм Евклида

- Задача: найдите наибольший общий делитель двух заданных чисел.

```
1  int gcd (int a, int b)
2  {
3      if (a == 0)
4          return b;
5      return gcd (b % a, a);
6  }
```

Как работает вызов `gcd (7, 5)`?

`gcd (a:7, b:5) line 5`

Пример 2: алгоритм Евклида

- Задача: найдите наибольший общий делитель двух заданных чисел.

```
1  int gcd (int a, int b)
2  {
3      if (a == 0)
4          return b;
5      return gcd (b % a, a);
6  }
```

Как работает вызов `gcd (7, 5)`?

```
gcd (a:7, b:5) line 5
  gcd (a:5, b:7) line 3
```

Пример 2: алгоритм Евклида

- Задача: найдите наибольший общий делитель двух заданных чисел.

```
1  int gcd (int a, int b)
2  {
3      if (a == 0)
4          return b;
5      return gcd (b % a, a);
6  }
```

Как работает вызов `gcd (7, 5)`?

```
gcd (a:7, b:5) line 5
  gcd (a:5, b:7) line 5
```

Пример 2: алгоритм Евклида

- Задача: найдите наибольший общий делитель двух заданных чисел.

```
1  int gcd (int a, int b)
2  {
3      if (a == 0)
4          return b;
5      return gcd (b % a, a);
6  }
```

Как работает вызов `gcd (7, 5)`?

```
gcd (a:7, b:5) line 5
  gcd (a:5, b:7) line 5
    gcd (a:2, b:5) line 3
```

Пример 2: алгоритм Евклида

- Задача: найдите наибольший общий делитель двух заданных чисел.

```
1  int gcd (int a, int b)
2  {
3      if (a == 0)
4          return b;
5      return gcd (b % a, a);
6  }
```

Как работает вызов `gcd (7, 5)`?

```
gcd (a:7, b:5) line 5
  gcd (a:5, b:7) line 5
    gcd (a:2, b:5) line 5
```

Пример 2: алгоритм Евклида

- Задача: найдите наибольший общий делитель двух заданных чисел.

```
1  int gcd (int a, int b)
2  {
3      if (a == 0)
4          return b;
5      return gcd (b % a, a);
6  }
```

Как работает вызов `gcd (7, 5)`?

```
gcd (a:7, b:5) line 5
  gcd (a:5, b:7) line 5
    gcd (a:2, b:5) line 5
      gcd (a:1, b:2) line 3
```

Пример 2: алгоритм Евклида

- Задача: найдите наибольший общий делитель двух заданных чисел.

```
1  int gcd (int a, int b)
2  {
3      if (a == 0)
4          return b;
5      return gcd (b % a, a);
6  }
```

Как работает вызов `gcd (7, 5)`?

```
gcd (a:7, b:5) line 5
  gcd (a:5, b:7) line 5
    gcd (a:2, b:5) line 5
      gcd (a:1, b:2) line 5
```

Пример 2: алгоритм Евклида

- Задача: найдите наибольший общий делитель двух заданных чисел.

```
1  int gcd (int a, int b)
2  {
3      if (a == 0)
4          return b;
5      return gcd (b % a, a);
6  }
```

Как работает вызов `gcd (7, 5)`?

```
gcd (a:7, b:5) line 5
  gcd (a:5, b:7) line 5
    gcd (a:2, b:5) line 5
      gcd (a:1, b:2) line 5
        gcd (a:0, b:1) line 3
```

Пример 2: алгоритм Евклида

- Задача: найдите наибольший общий делитель двух заданных чисел.

```
1  int gcd (int a, int b)
2  {
3      if (a == 0)
4          return b;
5      return gcd (b % a, a);
6  }
```

Как работает вызов `gcd (7, 5)`?

```
gcd (a:7, b:5) line 5
  gcd (a:5, b:7) line 5
    gcd (a:2, b:5) line 5
      gcd (a:1, b:2) line 5
        gcd (a:0, b:1) line 4 → 1
```

Пример 2: алгоритм Евклида

- Задача: найдите наибольший общий делитель двух заданных чисел.

```
1  int gcd (int a, int b)
2  {
3      if (a == 0)
4          return b;
5      return gcd (b % a, a);
6  }
```

Как работает вызов `gcd (7, 5)`?

```
gcd (a:7, b:5) line 5
  gcd (a:5, b:7) line 5
    gcd (a:2, b:5) line 5
      gcd (a:1, b:2) line 5 → 1
```

Пример 2: алгоритм Евклида

- Задача: найдите наибольший общий делитель двух заданных чисел.

```
1  int gcd (int a, int b)
2  {
3      if (a == 0)
4          return b;
5      return gcd (b % a, a);
6  }
```

Как работает вызов `gcd (7, 5)`?

```
gcd (a:7, b:5) line 5
  gcd (a:5, b:7) line 5
    gcd (a:2, b:5) line 5 → 1
```

Пример 2: алгоритм Евклида

- Задача: найдите наибольший общий делитель двух заданных чисел.

```
1  int gcd (int a, int b)
2  {
3      if (a == 0)
4          return b;
5      return gcd (b % a, a);
6  }
```

Как работает вызов `gcd (7, 5)`?

```
gcd (a:7, b:5) line 5
  gcd (a:5, b:7) line 5 → 1
```

Пример 2: алгоритм Евклида

- Задача: найдите наибольший общий делитель двух заданных чисел.

```
1  int gcd (int a, int b)
2  {
3      if (a == 0)
4          return b;
5      return gcd (b % a, a);
6  }
```

Как работает вызов `gcd (7, 5)`?

`gcd (a:7, b:5) line 5` $\rightarrow$ `1`

Как написать правильное решение

- Придумать решение
- Оценить время работы
- Подумать про крайние случаи
- Постараться придумать общее решение
- Придумать, как написать код
- Написать его
- Скомпилировать
- Разобраться с предупреждениями компилятора
- Запустить на примерах
- Запустить на крайних случаях
- Запустить на максимальных тестах
- Послать на проверку

Что могло пойти не так

- Ошибки случаются, часто и много
  - Значит, нужно уметь на них реагировать
- Программу приходится читать чаще, чем писать
  - Значит, её должно быть возможно прочитать

Как написать читаемую программу?

- Отступы
- Разбиение на строки
- Разбиение на функции
- Делайте одинаковые вещи одинаково

Вопросы?

Вопросы?