Цикл «Статьи для начинающих»

Как организовать ввод и вывод данных (статья 2)

Файловый ввод / вывод
Когда Вы отсылаете в проверяющую систему исходный текст Вашей программы, этот исходный текст компилируется выбранным Вами на странице отправки решения компилятором. Если компиляция прошла успешно, Ваше решение будет последовательно исполняться на тестах. Для этого оно должно уметь прочитать входные данные и сформировать выходные данные.

Разные проверяющие системы поддерживают разные способы представления входных и выходных данных.
Один из способов состоит в том, что входные данные для решения подаются в стандартный поток ввода, а выходные данные должны быть сформированы в стандартном потоке вывода.
Другой способ предполагает, что и входные данные содержатся в файле с одним заранее определенным именем, а выходные данные формируются в файле с другим заранее определенным именем. Имена этих файлов, как правило, указываются в условии задачи.

В проверяющей системе на contest.uni-smr.ac.ru используется второй способ. При этом имя файла, содержащего входные данные, всегда (независимо от задачи) **input.txt**, а имя файла, содержащего выходные данные, всегда (независимо от задачи) **output.txt**.

Приведем примеры решения задачи A+B, в которой требуется сложить два целых числа, на разных языках.

Примеры ввода / вывода на языках семейства Pascal

Установленный в системе компилятор — Free Pascal (актуальную версию уточняйте на странице отправки решения).

Программа на Free Pascal

```
var
    a, b: Longint;
    inf, ouf: text;
begin
    assign(inf, 'input.txt');
    assign(ouf, 'output.txt');
    reset(inf);
    rewrite(ouf);
    read(inf, a, b);
    write(ouf, a + b);
    close(inf);
    close(ouf);
end.
```

Можно использовать для файловых переменных имена input и output:

```
var
    a, b: integer;
    input, output: text;
```

```
begin
    assign(input, 'input.txt');
    assign(output, 'output.txt');
    reset(input);
    rewrite(output);
    read(input, a, b);
    write(output, a + b);
    close(input);
    close(output);
end.
```

Заметим, что программа на Pascal ABC будет выглядеть аналогично.
Также можно перенаправить стандартные потоки ввода / вывода (как это делается, написано чуть ниже, в примере для Delphi; это же код может быть отправлен как код на Free Pascal).

Программа на Delphi, заголовок программы указывать не обязательно.
Программы на Delphi компилируются с опцией MDelphi

```
var
fin, fout: TextFile;  //переменные для работы с текстовыми файлами
a, b, c: Longint;  // исходные данные и результат

begin
{$APPTYPE CONSOLE}

AssignFile(fin, 'input.txt'); // связываем файловую переменную с текстовым
Reset(fin); // файлом, содержащим входные данные, и открываем его

read(fin, a, b); // прочитываем из файла входные данные

CloseFile(fin); // завершение работы с входным файлом

c:=a+b; // собственно решение задачи

AssignFile(fout, 'output.txt'); // связываем файловую переменную с текстовым
Rewrite(fout); // файлом, в который запишем результат

write(fout, c); // вывод решения

CloseFile(fout); // завершение работы с выходным файлом - обязательно!

end.
```

Обратите внимание, что при чтении входных данных файл открывается процедурой Reset, а при записи выходных данных файл открывается процедурой Rewrite. Если, к примеру, открыть файл, предназначенный для записи результата, процедурой Reset, то в файл ничего не запишется.

Еще один способ состоит в перенаправлении стандартных потоков ввода / вывода

```
var
    a, b: LongInt;
begin
    assign(input, 'input.txt');
    assign(output, 'output.txt');
    reset(input);
    rewrite(output);
    read(a, b);
```

```
    write(a + b);
    close(input);
    close(output);
end.
```

Примеры ввода / вывода на языке Python

В настоящий момент в системе установлены компиляторы для Python 2 и Python 3 (актуальные версии уточняйте на странице отправки решения)
Язык Python входит в дополнительную группу языков, поэтому существование полных решений для всех задач, предлагаемых на официальных соревнованиях, не гарантируется.

```
infile = open('input.txt', 'r')
outfile = open('output.txt', 'w')
a, b = [int(x) for x in infile.readline().split()]
outfile.write(str(a+b) + '\n');
outfile.close();
infile.close();
```

Вместо строки
`outfile.write(str(a+b) + '\n');`
можно использовать
`print(x+y, file = outfile)`

Примеры ввода / вывода на языке C++

```
#include <iostream>
#include <cstdio>

using namespace std;

int main()
{
	FILE *in = freopen("input.txt", "r", stdin);
	FILE *out = freopen("output.txt", "w", stdout);
	//------------------------------------------------------------

	int a, b;
	cin >> a >> b;
	cout << a + b;
	/*
	Или:
	scanf("%d%d", &a, &b);
	printf("%d", a + b);
	*/

	//------------------------------------------------------------
	fclose(in);
	fclose(out);
	return 0;
}
```

В случае большого объема входных (или выходных) данных использование scanf (printf) вместо cin (cout) может оказаться более уместным (чтение с помощью cin требует большего времени).

Примеры ввода / вывода на языке Java

Обратите внимание, что в отличие от ряда других проверяющих систем, в системе, установленной на contest.uni-smr.ac.ru необходимо указывать пакет (package), имя которого не должно содержать точки.

```
package aplusb;

import java.util.*;
import java.io.*;

class A_plus_B {

    int a;
    int b;
    String Solution;

    void readData() throws IOException {
        FileReader fin = new FileReader("input.txt");
        Scanner scr = new Scanner(fin);

        a = scr.nextInt();
        b = scr.nextInt();

        fin.close();
    }

   void solution(){
       int sol = a + b;
       Solution = String.valueOf(sol);
   }

   void writeData() throws IOException {

       FileWriter fout = new FileWriter("output.txt");
       fout.write(Solution);
       fout.flush();
       fout.close();
   }
}

public class AplusB {

    /**
     * @param args the command line arguments
     * @throws java.io.IOException
     */
    public static void main(String[] args) throws IOException {
        A_plus_B ab = new A_plus_B();
        ab.readData();
        ab.solution();
        ab.writeData();
    }
}
```

В случае, когда необходимо прочитать достаточно большой объем входных данных, рекомендуется использовать класс StreamTokenizer вместо класса Scanner и методы этого класса для чтения. Пример использования этого класса приведен ниже.
Если необходимо выводить большой объем данных, рекомендуется делать это, избегая

излишнего формирования новых строк (т. е. вместо вывода вида a + " " + b выводить сначала a, потом пробельный символ, потом b).

```
package aplusb;

import java.util.*;
import java.io.*;

class A_plus_B {

    int a;
    int b;
    int Solution;
    StreamTokenizer stk;

    void readData() throws IOException {
        stk =
           new StreamTokenizer(new BufferedReader(new FileReader("input.txt")));

        a = nextInt();
        b = nextInt();
    }

    int nextInt() throws IOException{
        stk.nextToken();
        return (int)stk.nval;
    }

    void solution(){
       Solution = = a + b;;
    }

    void writeData() throws IOException {

       PrintWriter fout = new PrintWriter("output.txt");
       fout.println(Solution);
       fout.flush();
       fout.close();
    }
}

public class AplusB {

    /**
     * @param args the command line arguments
     * @throws java.io.IOException
     */
    public static void main(String[] args) throws IOException {
        A_plus_B ab = new A_plus_B();
        ab.readData();
        ab.solution();
        ab.writeData();
    }

}
```

Разумеется, существуют и другие возможности организовать ввод / вывод. В настоящей статье рассматриваются только базовые возможности, которые позволят Вам начать работу с системой.