Памятка участника муниципального этапа Всероссийской олимпиады школьников по информатике, Липецк, 2014/15

Организация тура

Перед началом тура компьютеры участников будут *включены*, клавиатура будет лежать на столе или системном блоке. Участникам *строго запрещено* трогать компьютер, клавиатуру до начала олимпиады.

О начале тура будет объявлено дежурными.

Если у участника возник вопрос по задаче, он должен взять у дежурного бланк вопроса, заполнить его и отдать дежурному.

Вопрос должен быть сформулирован таким образом, чтобы подразумевать ответ «да» или «нет».

Если у участника возникли проблемы, вопросы, а также если он хочет передать свой вопрос (записанный на бланке) жюри, участник должен обратится к любому из дежурных.

Во время туров участникам олимпиады запрещается пользоваться любыми электронными устройствами, в том числе личными компьютерами, калькуляторами, электронными записными книжками, средствами связи (пейджерами, мобильными телефонами и т. п.), принесенными электронными носителями информации (дискетами, CD- и DVD-дисками, модулями флэш-памяти и т.п.), а также учебной литературой и заготовленными личными записями.

По истечении времени тура участникам олимпиады запрещается выполнять любые действия на компьютере.

Участникам разрешается приносить с собой воду или питание при условии соблюдения порядка пользования компьютерными классами. Централизованного питания во время тура не предусмотрено.

Конфигурация компьютеров

В распоряжение участников олимпиады будут предоставлены рабочие места, оснащенные современными компьютерами.

На компьютере каждого участника олимпиады будет установлено следующее программное обеспечение.

- Borland Delphi или Free Pascal или Pascal ABC.Net
- MinGW (GNU C/C++) или Microsoft Visual Studio

На некоторых компьютерах могут также быть установлены Far Manager, Microsoft Visual Studio, Java, Python, C#. Если необходимое ПО не установлено, участник может попросить дежурных пересадить его за другой компьютер. В каждой точке проведения оргкомитет постарается обеспечить возможность всем писать на их языке программирования.

Информация о рабочем каталогах, в которые установлено программное обеспечение зависит от места нахождения участника.

Проверка решений

Проверка решений участников может производиться с использованием специализированной проверяющей программной среды.

Во время тура участники отсылают решения задач на предварительную проверку.

Предварительная проверка решений проводится во время всего тура. В результате предварительной проверки решение либо принимается на проверку, либо отклоняется.

Решением каждой задачи является программа. Во время предварительной проверки она запускается на тестах из условия задачи. Если в условии задачи не оговорено обратное, программа принимается на проверку, если она успешно прошла все тесты.

Решение должно читать данные из файла, указанного в условии задачи и выводить результат в выходной файл.

Результат предварительной проверки сообщается участнику. Если решение отклонено, то участнику сообщается причина этого.

После завершения тура производится окончательная проверка решений, принятых на проверку. Если у участника по какой-либо задаче принято на проверку несколько решений, то оценивается последнее из них. Решения, не принятые на проверку, не оцениваются.

Максимальное количество баллов, которое может набрать участник по результатам проверки задачи, указывается в ее условии.

Оценка правильности решения осуществляется путем исполнения программы с заранее подготовленными и неизвестными участникам входными файлами с последующим анализом получаемых в результате этого выходных файлов.

Оценка полученного участником решения каждой задачи осуществляется по результатам прохождения каждого теста из набора тестов для этой задачи.

Количество баллов, получаемое участником по результатам прохождения каждого теста, определяется утвержденной жюри системой начисления баллов.

Результатами многократного исполнения решения с одними и теми же входными файлами должны быть одинаковые выходные файлы, вне зависимости от времени запуска программы и ее программного окружения. Жюри вправе произвести неограниченное количество запусков программы участника и выбрать наихудший результат по каждому из тестов.

Размер файла с исходным текстом не должен превышать 256 килобайт. Время компиляции программы не должно превышать одной минуты.

Памятка участника муниципального этапа Всероссийской олимпиады школьников по информатике, Липецк, 2014/15

Максимальное время работы и объем используемой памяти будут указаны в условиях задач. Временем работы программы считается суммарное время работы процесса на всех ядрах процессора. Память, используемая приложением, включает всю память, которая выделена процессу операционной системой, включая память кода и стек.

Участникам олимпиады разрешается использование в решениях задач любых внешних модулей и заголовочных файлов, включенных в стандартную поставку соответствующего компилятора.

Жюри использует следующие командные строки для компиляции решений.

Компилятор	Командная строка
Borland Delphi	`dcc32 -cc <исходный файл>`
Free Pascal	`fpc <исходный файл>`
Pascal ABC.Net	`pabcnetcc <исходный файл>`
Visual C	`cl /O2 /TC <исходный файл>`
GNU C (MinGW)	`gcc -O2 -Wl,--stack=67108864 -x c <исходный файл>`
Visual C++	`cl /O2 /EHs /TP <исходный файл>`
GNU C++ (MinGW)	`g++ -O2 -Wl,--stack=67108864 -x c++ <исходный файл>`
Java	`javac <исходный файл>`
C#	`csc /optimize <исходный файл>`
Python 3	`компиляция не производится`

Жюри оставляет за собой право изменять команды компиляции решений в процессе проведения соревнований, о чем участники олимпиады информируются перед началом тура.

Возможные результаты предварительной проверки решений перечислены в таблице.

Результат	Тест	Комментарий	Возможные причины
Compilation error	Нет	Исполняемый файл не был создан при компиляции	• Синтаксическая ошибка в программе; • Неправильно указано расширение файла или язык программирования.
Security Violation	Да	Программа нарушает правила олимпиады	• Ошибка в программе; • Попытка срыва работы проверяющей системы.
Memory limit exceeded	Да	Программа превысила лимит используемой памяти.	• Неэффективное решение; • Ошибка в программе.
Time limit exceeded	Да	Программа превысила лимит времени работы.	• Неэффективное решение; • Ошибка в программе.
Idleness limit exceeded	Да	Программа перешла в состояние ожидания и не выполняет никаких действий.	• Чтение с клавиатуры. • Намеренный переход в состояние ожидания (sleep).
Runtime error	Да	Программа завершилась с ненулевым кодом возврата или сгенерировала исключительную ситуацию.	• Ошибка времени исполнения; • Не хватает «`return 0`» в программе на C/C++; • «`exit(не-ноль)`» в C/C++; • «`halt(не-ноль)`» в Delphi; • «`System.exit(не-ноль)`» в Java; • Неперехваченное исключение.
Presentation error	Да	Проверяющая программа не может проверить правильность вывода, потому что он не соответствует принятому формату.	• Формат вывода некорректен; • Программа не создала выходного файла или создала файл с неверным именем.
Wrong answer	Да	Неверный ответ.	• Неверный алгоритм; • Ошибка в реализации алгоритма.
Accepted	Нет	Программа прошла предварительные тесты и принята на проверку.	Программа корректна.

В решениях задач участникам запрещается:

- создавать каталоги и временные файлы при работе программы;
- осуществлять чтение и запись векторов прерываний;
- любое использование сетевых средств;
- любые другие действия, нарушающие работу проверяющей системы.

Под конец тура очередь на тестирование может быть довольно большой, поэтому результаты будут приходить с задержкой. Вы можете продолжать решать задачи во время ожидания результата предварительной проверки.

Работа с проверяющей программной средой

Чтобы запустить клиент проверяющей программной среды, запустите браузер и перейдите на страницу входа в проверяющую среду (адрес будет указан дежурным).

Введите *имя пользователя* и *пароль*, указанные на листе, находящемся под клавиатурой.

Основная страница клиента проверяющей программной среды состоит из трех разделов:

- *Информация* — содержит информацию об участнике и олимпиаде.
- *Отправка решения* — позволяет отправить решение на предварительную проверку.
- *Результаты проверки* — отображает результаты проверки отправленных решений.

Для того, чтобы отправить решение на предварительную проверку, укажите задачу, которую вы решили в поле *Выберите задачу*. Затем укажите язык на котором решена задача в поле *Выберите язык*. Далее укажите имя файла, содержащего решения в поле *Выберите файл*. Проверьте правильность введенной информации и нажмите на кнопку *Отправить на проверку*.

Памятка по работе с файлами

Ваши решения должны читать входные данные из входного файла и выводить результат работы в выходной файл. Имена файлов указаны в решении задачи. Ниже приведен пример программы на всех разрешенных языках программирования, которая решает задачу «даны два числа, выведите их сумму», считывая числа из файла «sum.in» и записывая результат в файл «sum.out».

Паскаль

Чтобы процедуры «read» и «readln» читали из файла, в начале программы напишите

```
assign(input, 'sum.in');
reset(input);
```

Чтобы процедуры «write» и «writeln» писали в файл, в начале программы напишите

```
assign(output, 'sum.out');
rewrite(output);
```

Переменные input и output заводить не надо – это системные переменные. После выполнения этих команд ваша программа будет читать данные из указанного файла и выводить данные в указанный файл. Если вы пишете с использованием Free Pascal, то в конце программы файлы надо закрыть, написав

```
close(input);
close(output);
```

Таким образом, полностью программа выглядит так:

```
var
 a, b: longint;
begin
 assign(input, 'sum.in');
 reset(input);
 assign(output, 'sum.out');
 rewrite(output);
 read(a, b);
 writeln(a + b);
 close(input);
 close(output);
end.
```

Borland Delphi

В Borland Delphi все почти также, но вместо assign надо писать assignfile. Можно также сразу открыть файл на чтение с помощью reset и на запись с помощью rewrite, указав имя файла. Закрывать файлы в Delphi не требуется. Получается следующая программа.

```
var
 a, b: longint;
begin
 reset(input, 'sum.in');
 rewrite(output, 'sum.out');
 read(a, b);
 writeln(a + b);
```

```
end.
```

C++

Во всех компиляторах C++ чтобы читать и писать из файлов надо переназначить дескрипторы stdin и stdout на соответствующие файлы. Это делается с помощью процедуры freopen.

Программа:

```
#include <cstdio>
using namespace std;
int main() {
 freopen("sum.in", "r", stdin);
 freopen("sum.out", "w", stdout);
 int a, b;
 scanf("%d%d", &a, &b);
 printf("%d\n", a + b);
 fclose(stdin);
 fclose(stdout);
 return 0;
}
```

Python

Файлы открываются с помощью процедуры open. Процедуре print после этого можно передать в качестве именного параметра file дескриптор файла.

Необходимо закрывать файл после записи.

```
inf = open("sum.in", "r")
ouf = open("sum.out", "w")
a, b = map(int, inf.readline().split())
print(a + b, file = ouf)
ouf.close() Java
```

Рекомендуем создать Scanner для чтения и PrintWriter для записи

Программа:

```
import java.util.*;
import java.io.*;
public class Sum {
 public static void main(String[] s) throws IOException {
 Scanner in = new Scanner(new File("sum.in"));
 int a = in.nextInt();
 int b = in.nextInt();
 in.close();
 PrintWriter out = new PrintWriter("sum.out");
 out.println(a + b);
 out.close();
 }
}
```

C#

Программа:

```
using System;
using System.IO;

public class Sum {
 public static void Main(string[] args) {
 TextReader inf = new StreamReader("sum.in");
 TextWriter ouf = new StreamWriter("sum.out");
 string[] s = inf.ReadLine().Split();
 int a = Int32.Parse(s[0]);
 int b = Int32.Parse(s[1]);
 ouf.WriteLine(a + b);
 inf.Close();
 ouf.Close();
 }
}
```