**Министерство образования и науки Российской Федерации**

**САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ,   
МЕХАНИКИ И ОПТИКИ**

Факультет программной инженерии и компьютерной техники  
Направление подготовки 09.03.04 Программная инженерия

Дисциплина «Aлгоритмы и структуры данных»

**ОТЧЁТ**

по лабораторной работе №4 (Week 4 Openedu)

Студенка Кузенкова Елизавета группы P3217

Преподаватель Муромцев Дмитрий Ильич

Санкт-Петербург

2019 г.

Содержание

[Задача 1 Стек 3](#_Toc3287484)

[Исходный код к задаче 1 3](#_Toc3287485)

[Бенчмарк к задаче 1 4](#_Toc3287486)

[Задача 2 Очередь 5](#_Toc3287487)

[Исходный код к задаче 2 6](#_Toc3287488)

[Бенчмарк к задаче 2 6](#_Toc3287489)

[Задача 3 Скобочная последовательность 7](#_Toc3287490)

[Исходный код к задаче 3 8](#_Toc3287491)

[Бенчмарк к задаче 3 9](#_Toc3287492)

[Задача 4 Очередь с минимумом 9](#_Toc3287493)

[Исходный код к задаче 4 10](#_Toc3287494)

[Бенчмарк к задаче 4 11](#_Toc3287495)

[Задача 5 Quack 13](#_Toc3287496)

[Исходный код к задаче 5 16](#_Toc3287497)

[Бенчмарк к задаче 5 20](#_Toc3287498)

# Задача 1 Стек

|  |  |
| --- | --- |
| Имя входного файла: | input.txt |
| Имя выходного файла: | output.txt |
| Ограничение по времени: | 2 секунды |
| Ограничение по памяти: | 256 мегабайт |

Реализуйте работу стека. Для каждой операции изъятия элемента выведите ее результат.

На вход программе подаются строки, содержащие команды. Каждая строка содержит одну команду. Команда — это либо "+ N”", либо "−". Команда "+ " означает добавление в стек числа , по модулю не превышающего . Команда "−" означает изъятие элемента из стека. Гарантируется, что не происходит извлечения из пустого стека. Гарантируется, что размер стека в процессе выполнения команд не превысит элементов.

#### Формат входного файла

В первой строке входного файла содержится ( ) — число команд. Каждая последующая строка исходного файла содержит ровно одну команду.

#### Формат выходного файла

Выведите числа, которые удаляются из стека с помощью команды "−", по одному в каждой строке. Числа нужно выводить в том порядке, в котором они были извлечены из стека. Гарантируется, что изъятий из пустого стека не производится.

#### Пример

|  |  |
| --- | --- |
| input.txt | output.txt |
| 6 + 1 + 10 - + 2 + 1234 - | 10 1234 |

## Исходный код к задаче 1

class Lab4\_1

{

public static void Main(string[] args)

{

var app = new Lab4\_1();

app.DoWork(args);

}

private void DoWork(string[] args)

{

using (StreamWriter sw = new StreamWriter("output.txt"))

{

string[] stdin = File.ReadAllLines("input.txt");

int[] parameters = stdin[0].Split(' ').Select(t => int.Parse(t)).ToArray();

Stack<long> queue = new Stack<long>();

for (int i = 1; i < stdin.Length; i++)

{

string[] temp = stdin[i].Split(' ');

if (temp[0] == "-")

sw.WriteLine(queue.Pop());

else

queue.Push(long.Parse(temp[1]));

}

}

}

}

## Бенчмарк к задаче 1

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **№ теста** | **Результат** | **Время, с** | **Память** | **Размер входного файла** | **Размер выходного файла** |
| Max |  | 0.671 | 98164736 | 13389454 | 5693807 |
| 1 | OK | 0.031 | 11350016 | 33 | 10 |
| 2 | OK | 0.031 | 11366400 | 11 | 3 |
| 3 | OK | 0.046 | 11354112 | 19 | 6 |
| 4 | OK | 0.031 | 11374592 | 19 | 6 |
| 5 | OK | 0.031 | 11362304 | 19 | 6 |
| 6 | OK | 0.031 | 11358208 | 96 | 45 |
| 7 | OK | 0.078 | 11366400 | 85 | 56 |
| 8 | OK | 0.046 | 11358208 | 129 | 11 |
| 9 | OK | 0.031 | 11366400 | 131 | 12 |
| 10 | OK | 0.031 | 11436032 | 859 | 540 |
| 11 | OK | 0.031 | 11370496 | 828 | 573 |
| 12 | OK | 0.031 | 11382784 | 1340 | 11 |
| 13 | OK | 0.031 | 11247616 | 1325 | 12 |
| 14 | OK | 0.015 | 11419648 | 8292 | 5590 |
| 15 | OK | 0.031 | 11431936 | 8212 | 5706 |
| 16 | OK | 0.062 | 11513856 | 13298 | 111 |
| 17 | OK | 0.031 | 11517952 | 13354 | 12 |
| 18 | OK | 0.046 | 14004224 | 82372 | 56548 |
| 19 | OK | 0.046 | 14147584 | 82000 | 56993 |
| 20 | OK | 0.031 | 14671872 | 132796 | 1134 |
| 21 | OK | 0.046 | 14643200 | 133914 | 11 |
| 22 | OK | 0.093 | 27987968 | 819651 | 569557 |
| 23 | OK | 0.093 | 26710016 | 819689 | 569681 |
| 24 | OK | 0.109 | 27983872 | 1328670 | 11294 |
| 25 | OK | 0.093 | 27820032 | 1338543 | 11 |
| 26 | OK | 0.593 | 80633856 | 8196274 | 5693035 |
| 27 | OK | 0.562 | 81190912 | 8193816 | 5693807 |
| 28 | OK | 0.671 | 97771520 | 13286863 | 112020 |
| 29 | OK | 0.656 | 98152448 | 13389454 | 11 |
| 30 | OK | 0.656 | 98164736 | 13388564 | 11 |

# Задача 2 Очередь

|  |  |
| --- | --- |
| Имя входного файла: | input.txt |
| Имя выходного файла: | output.txt |
| Ограничение по времени: | 2 секунды |
| Ограничение по памяти: | 256 мегабайт |

Реализуйте работу очереди. Для каждой операции изъятия элемента выведите ее результат.

На вход программе подаются строки, содержащие команды. Каждая строка содержит одну команду. Команда — это либо «+ N», либо «−». Команда «+ N» означает добавление в очередь числа N, по модулю не превышающего . Команда «−» означает изъятие элемента из очереди. Гарантируется, что размер очереди в процессе выполнения команд не превысит элементов.

#### Формат входного файла

В первой строке входного файла содержится ) — число команд. Каждая последующая строка исходного файла содержит ровно одну команду.

#### Формат выходного файла

Выведите числа, которые удаляются из очереди с помощью команды «−», по одному в каждой строке. Числа нужно выводить в том порядке, в котором они были извлечены из очереди. Гарантируется, что извлечения из пустой очереди не производится.

#### Пример

|  |  |
| --- | --- |
| input.txt | output.txt |
| 4 + 1 + 10 - - | 1 10 |

## Исходный код к задаче 2

class Lab4\_2

{

public static void Main(string[] args)

{

var app = new Lab4\_2();

app.DoWork(args);

}

private void DoWork(string[] args)

{

using (StreamWriter sw = new StreamWriter("output.txt"))

{

string[] stdin = File.ReadAllLines("input.txt");

int[] parameters = stdin[0].Split(' ').Select(t => int.Parse(t)).ToArray();

var queue = new Queue<long>();

for (int i = 1; i < stdin.Length; i++)

{

string[] temp = stdin[i].Split(' ');

if (temp[0] == "-")

sw.WriteLine(queue.Dequeue());

else

queue.Enqueue(long.Parse(temp[1]));

}

}

}

}

## Бенчмарк к задаче 2

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **№ теста** | **Результат** | **Время, с** | **Память** | **Размер входного файла** | **Размер выходного файла** |
| Max |  | 0.687 | 98308096 | 13389454 | 5693807 |
| 1 | OK | 0.046 | 11517952 | 20 | 7 |
| 2 | OK | 0.031 | 11513856 | 11 | 3 |
| 3 | OK | 0.031 | 11501568 | 19 | 6 |
| 4 | OK | 0.031 | 11526144 | 19 | 6 |
| 5 | OK | 0.031 | 11481088 | 96 | 45 |
| 6 | OK | 0.015 | 11571200 | 85 | 56 |
| 7 | OK | 0.031 | 11526144 | 129 | 12 |
| 8 | OK | 0.031 | 11481088 | 131 | 12 |
| 9 | OK | 0.031 | 11509760 | 859 | 538 |
| 10 | OK | 0.062 | 11530240 | 828 | 573 |
| 11 | OK | 0.031 | 11530240 | 1340 | 12 |
| 12 | OK | 0.031 | 11620352 | 1325 | 12 |
| 13 | OK | 0.031 | 11550720 | 8292 | 5589 |
| 14 | OK | 0.015 | 11554816 | 8212 | 5706 |
| 15 | OK | 0.031 | 11591680 | 13298 | 115 |
| 16 | OK | 0.031 | 11636736 | 13354 | 12 |
| 17 | OK | 0.031 | 14065664 | 82372 | 56552 |
| 18 | OK | 0.031 | 14237696 | 82000 | 56993 |
| 19 | OK | 0.046 | 14794752 | 132796 | 1124 |
| 20 | OK | 0.031 | 14807040 | 133914 | 12 |
| 21 | OK | 0.093 | 28143616 | 819651 | 569553 |
| 22 | OK | 0.093 | 26779648 | 819689 | 569681 |
| 23 | OK | 0.109 | 27389952 | 1328670 | 11296 |
| 24 | OK | 0.093 | 27889664 | 1338543 | 12 |
| 25 | OK | 0.625 | 80711680 | 8196274 | 5693025 |
| 26 | OK | 0.609 | 81244160 | 8193816 | 5693807 |
| 27 | OK | 0.671 | 97927168 | 13286863 | 112110 |
| 28 | OK | 0.687 | 98308096 | 13389454 | 10 |
| 29 | OK | 0.687 | 98279424 | 13388564 | 11 |

# Задача 3 Скобочная последовательность

|  |  |
| --- | --- |
| Имя входного файла: | input.txt |
| Имя выходного файла: | output.txt |
| Ограничение по времени: | 2 секунды |
| Ограничение по памяти: | 256 мегабайт |

Последовательность A, состоящую из символов из множества «(», «)», «[» и «]», назовем правильной скобочной последовательностью, если выполняется одно из следующих утверждений:

 A — пустая последовательность;

 первый символ последовательности A — это «(», и в этой последовательности существует такой символ «)», что последовательность можно представить как A=(B)C, где B и C — правильные скобочные последовательности;

 первый символ последовательности A — это «[», и в этой последовательности существует такой символ «]», что последовательность можно представить как A=[B]C , где B и C— правильные скобочные последовательности.

Так, например, последовательности «(())» и «()[]» являются правильными скобочными последовательностями, а последовательности «[)» и «((» таковыми не являются.

Входной файл содержит несколько строк, каждая из которых содержит последовательность символов «(», «)», «[» и «]». Для каждой из этих строк выясните, является ли она правильной скобочной последовательностью.

#### Формат входного файла

Первая строка входного файла содержит число N( - число скобочных последовательностей, которые необходимо проверить. Каждая из следующих N строк содержит скобочную последовательность длиной от 1 до включительно. В каждой из последовательностей присутствуют только скобки указанных выше видов.

#### Формат выходного файла

Для каждой строки входного файла выведите в выходной файл «YES», если соответствующая последовательность является правильной скобочной последовательностью, или «NO», если не является.

#### Пример

|  |  |
| --- | --- |
| input.txt | output.txt |
| 5 ()() ([]) ([)] ((]] )( | YES YES NO NO NO |

## Исходный код к задаче 3

class Lab4\_3

{

public static void Main(string[] args)

{

var app = new Lab4\_3();

app.DoWork(args);

}

private void DoWork(string[] args)

{

using (StreamWriter sw = new StreamWriter("output.txt"))

{

string[] stdin = File.ReadAllLines("input.txt");

for (int i = 1; i < stdin.Length; i++)

{

var isCorrect = true;

Stack<char> brackets = new Stack<char>();

for (int j = 0; j < stdin[i].Length; j++)

switch (stdin[i][j])

{

case '(':

brackets.Push(stdin[i][j]);

break;

case '[':

brackets.Push(stdin[i][j]);

break;

case ')':

if (brackets.Count == 0 || brackets.Peek() != '(')

isCorrect = false;

if (brackets.Count != 0)

brackets.Pop();

break;

case ']':

if (brackets.Count == 0 || brackets.Peek() != '[')

isCorrect = false;

if (brackets.Count != 0)

brackets.Pop();

break;

}

if (isCorrect && brackets.Count == 0)

sw.WriteLine("YES");

else

sw.WriteLine("NO");

}

sw.Dispose();

}

}

}

## Бенчмарк к задаче 3

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **№ теста** | **Результат** | **Время, с** | **Память** | **Размер входного файла** | **Размер выходного файла** |
| Max |  | 0.140 | 27168768 | 5000885 | 2133 |
| 1 | OK | 0.031 | 10633216 | 31 | 22 |
| 2 | OK | 0.031 | 10838016 | 15 | 16 |
| 3 | OK | 0.031 | 10711040 | 68 | 66 |
| 4 | OK | 0.031 | 10682368 | 324 | 256 |
| 5 | OK | 0.031 | 10772480 | 1541 | 1032 |
| 6 | OK | 0.031 | 10764288 | 5880 | 2128 |
| 7 | OK | 0.031 | 10989568 | 50867 | 2129 |
| 8 | OK | 0.093 | 13393920 | 500879 | 2110 |
| 9 | OK | 0.125 | 27164672 | 5000884 | 2120 |
| 10 | OK | 0.140 | 27168768 | 5000885 | 2133 |

# Задача 4 Очередь с минимумом

|  |  |
| --- | --- |
| Имя входного файла: | input.txt |
| Имя выходного файла: | output.txt |
| Ограничение по времени: | 2 секунды |
| Ограничение по памяти: | 256 мегабайт |

Реализуйте работу очереди. В дополнение к стандартным операциям очереди, необходимо также отвечать на запрос о минимальном элементе из тех, которые сейчас находится в очереди. Для каждой операции запроса минимального элемента выведите ее результат.

На вход программе подаются строки, содержащие команды. Каждая строка содержит одну команду. Команда — это либо «+ N», либо «−», либо «?». Команда «+ N» означает добавление в очередь числа N, по модулю не превышающего . Команда «−» означает изъятие элемента из очереди. Команда «?» означает запрос на поиск минимального элемента в очереди.

#### Формат входного файла

В первой строке содержится )— число команд. В последующих строках содержатся команды, по одной в каждой строке.

#### Формат выходного файла

Для каждой операции поиска минимума в очереди выведите её результат. Результаты должны быть выведены в том порядке, в котором эти операции встречаются во входном файле. Гарантируется, что операций извлечения или поиска минимума для пустой очереди не производится.

#### Пример

|  |  |
| --- | --- |
| input.txt | output.txt |
| 7 + 1 ? + 10 ? - ? - | 1 1 10 |

## Исходный код к задаче 4

using System;

using System.Collections.Generic;

using System.IO;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Openedu.Week4

{

class Lab4\_4

{

public static void Main(string[] args)

{

var app = new Lab4\_4();

app.DoWork(args);

}

private void DoWork(string[] args)

{

using (StreamWriter sw = new StreamWriter("output.txt"))

{

string[] stdin = File.ReadAllLines("input.txt");

int[] parameters = stdin[0].Split(' ').Select(t => int.Parse(t)).ToArray();

var queue = new Queue<int>();

var minimums = new LinkedList<int>();

for (var i = 0; i < stdin.Length; i++)

{

var temp = stdin[i].Split(' ');

switch (temp[0])

{

case "+":

{

var r = int.Parse(temp[1]);

queue.Enqueue(r);

while (minimums.Count > 0 && minimums.First() > r)

{

minimums.RemoveFirst();

}

minimums.AddFirst(r);

break;

}

case "-":

{

var b = queue.Dequeue();

if (minimums.Last() == b)

{

minimums.RemoveLast();

}

break;

}

case "?":

{

sw.WriteLine(minimums.Last());

break;

}

default:

{

break;

}

}

}

}

}

}

}

## Бенчмарк к задаче 4

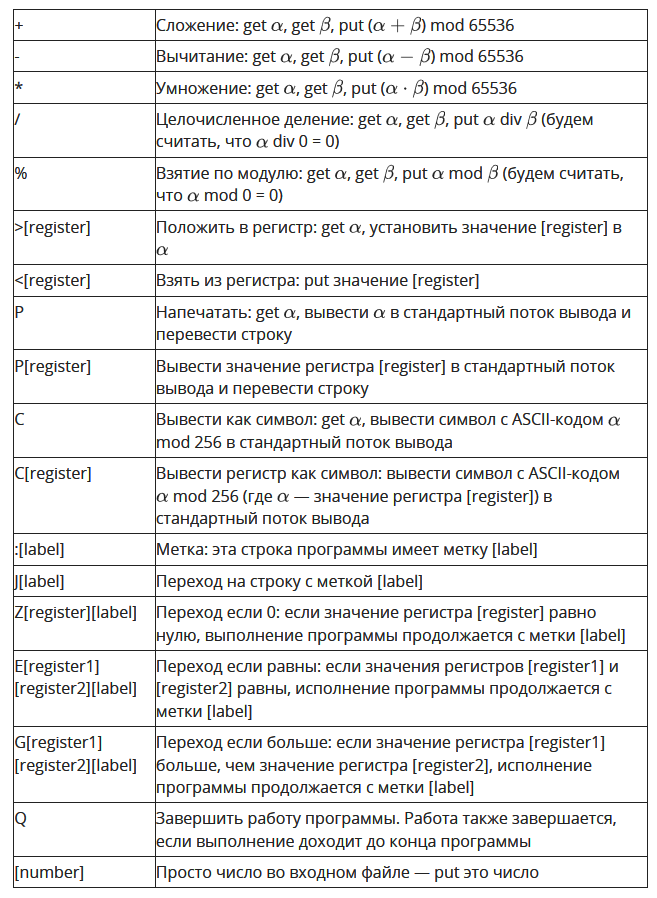
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **№ теста** | **Результат** | **Время, с** | **Память** | **Размер входного файла** | **Размер выходного файла** |
| Max |  | 1.203 | 145141760 | 13389342 | 4002151 |
| 1 | OK | 0.031 | 11444224 | 29 | 10 |
| 2 | OK | 0.031 | 11386880 | 11 | 3 |
| 3 | OK | 0.031 | 11431936 | 22 | 6 |
| 4 | OK | 0.031 | 11415552 | 22 | 6 |
| 5 | OK | 0.046 | 11448320 | 36 | 9 |
| 6 | OK | 0.015 | 11472896 | 48 | 12 |
| 7 | OK | 0.031 | 11448320 | 76 | 35 |
| 8 | OK | 0.031 | 11427840 | 129 | 12 |
| 9 | OK | 0.031 | 11440128 | 67 | 48 |
| 10 | OK | 0.046 | 11440128 | 44 | 9 |
| 11 | OK | 0.031 | 11481088 | 45 | 9 |
| 12 | OK | 0.031 | 11464704 | 44 | 9 |
| 13 | OK | 0.031 | 11378688 | 45 | 9 |
| 14 | OK | 0.031 | 11411456 | 721 | 384 |
| 15 | OK | 0.031 | 11390976 | 1340 | 12 |
| 16 | OK | 0.031 | 11444224 | 640 | 407 |
| 17 | OK | 0.031 | 11444224 | 445 | 90 |
| 18 | OK | 0.046 | 11427840 | 456 | 100 |
| 19 | OK | 0.031 | 11522048 | 445 | 90 |
| 20 | OK | 0.031 | 11464704 | 456 | 100 |
| 21 | OK | 0.031 | 11661312 | 6616 | 3812 |
| 22 | OK | 0.031 | 11776000 | 13389 | 12 |
| 23 | OK | 0.031 | 11640832 | 6461 | 4008 |
| 24 | OK | 0.031 | 11599872 | 4896 | 1140 |
| 25 | OK | 0.046 | 11628544 | 5007 | 1250 |
| 26 | OK | 0.046 | 11616256 | 4896 | 1140 |
| 27 | OK | 0.062 | 11603968 | 5007 | 1250 |
| 28 | OK | 0.046 | 13471744 | 64907 | 39589 |
| 29 | OK | 0.031 | 15343616 | 133814 | 12 |
| 30 | OK | 0.062 | 13516800 | 64675 | 39996 |
| 31 | OK | 0.031 | 14102528 | 53897 | 13890 |
| 32 | OK | 0.031 | 13545472 | 55008 | 15000 |
| 33 | OK | 0.031 | 14131200 | 53897 | 13890 |
| 34 | OK | 0.046 | 13598720 | 55008 | 15000 |
| 35 | OK | 0.093 | 28278784 | 645271 | 404305 |
| 36 | OK | 0.109 | 28430336 | 1338956 | 12 |
| 37 | OK | 0.078 | 28520448 | 646300 | 400008 |
| 38 | OK | 0.109 | 28590080 | 588898 | 163890 |
| 39 | OK | 0.078 | 28454912 | 600009 | 175000 |
| 40 | OK | 0.109 | 28585984 | 588898 | 163890 |
| 41 | OK | 0.093 | 28475392 | 600009 | 175000 |
| 42 | OK | 0.531 | 80834560 | 6465010 | 4002151 |
| 43 | OK | 0.734 | 91480064 | 13389342 | 12 |
| 44 | OK | 0.531 | 80977920 | 6462989 | 4000004 |
| 45 | OK | 0.734 | 107687936 | 6388899 | 1888890 |
| 46 | OK | 0.531 | 80949248 | 6500010 | 2000000 |
| 47 | OK | 0.781 | 108339200 | 6388899 | 1888890 |
| 48 | OK | 0.531 | 80965632 | 6500010 | 2000000 |
| 49 | OK | 0.734 | 91447296 | 13388086 | 12 |
| 50 | OK | 0.031 | 11403264 | 55 | 16 |
| 51 | OK | 0.031 | 11448320 | 705 | 225 |
| 52 | OK | 0.031 | 11649024 | 6506 | 2000 |
| 53 | OK | 0.031 | 14249984 | 65007 | 20000 |
| 54 | OK | 0.125 | 28815360 | 650008 | 200000 |
| 55 | OK | 0.812 | 107147264 | 6675213 | 2000000 |
| 56 | OK | 0.031 | 11423744 | 117 | 12 |
| 57 | OK | 0.031 | 11427840 | 1327 | 12 |
| 58 | OK | 0.046 | 11763712 | 13417 | 12 |
| 59 | OK | 0.046 | 15310848 | 133845 | 12 |
| 60 | OK | 0.156 | 30552064 | 1339319 | 12 |
| 61 | OK | 1.203 | 145141760 | 13388955 | 12 |

# Задача 5 Quack

|  |  |
| --- | --- |
| Имя входного файла: | input.txt |
| Имя выходного файла: | output.txt |
| Ограничение по времени: | 2 секунды |
| Ограничение по памяти: | 256 мегабайт |

Язык Quack — забавный язык, который фигурирует в одной из задач с [Internet Problem Solving Contest](http://ipsc.ksp.sk). В этой задаче вам требуется написать интерпретатор языка Quack.

Виртуальная машина, на которой исполняется программа на языке Quack, имеет внутри себя очередь, содержащую целые числа по модулю 65536 (то есть, числа от 0 до 65535, соответствующие беззнаковому 16-битному целому типу). Слово get в описании операций означает извлечение из очереди, put — добавление в очередь. Кроме того, у виртуальной машины есть 26 регистров, которые обозначаются буквами от 'a' до 'z'. Изначально все регистры хранят нулевое значение. В языке Quack существуют следующие команды (далее под a и b подразумеваются некие абстрактные временные переменные):



#### Формат входного файла

Входной файл содержит синтаксически корректную программу на языке Quack. Известно, что программа завершает работу не более чем за

шагов. Программа содержит не менее одной и не более

инструкций. **Метки имеют длину от 1 до 10 и состоят из цифр и латинских букв.**

#### Формат выходного файла

Выведите содержимое стандартного потока вывода виртуальной машины в выходной файл.

#### Примеры

|  |  |
| --- | --- |
| input.txt | output.txt |
| 100 0 :start >a Zaend <a <a 1 + - >b <b Jstart :end P | 5050 |

Второй пример подразумевает UNIX-переводы строки в ответе (один символ с кодом 10).

|  |  |
| --- | --- |
| input.txt | output.txt |
| 58 49 10 62 97 10 80 97 10 90 97 50 10 60 97 10 74 49 10 58 50 10 48 10 58 51 10 62 97 10 90 97 52 10 67 97 10 74 51 10 58 52 10 0 :1 >a Pa Za2 <a J1 :2 0 :3 >a Za4 Ca J3 :4 | 58 49 10 62 97 10 80 97 10 90 97 50 10 60 97 10 74 49 10 58 50 10 48 10 58 51 10 62 97 10 90 97 52 10 67 97 10 74 51 10 58 52 10 0 :1 >a Pa Za2 <a J1 :2 0 :3 >a Za4 Ca J3 :4 |

## Исходный код к задаче 5

class Lab4\_5

{

public static void Main(string[] args)

{

var app = new Lab4\_5();

app.DoWork(args);

}

private void DoWork(string[] args)

{

StreamWriter sw = new StreamWriter("output.txt");

Console.OutputEncoding = Encoding.ASCII;

Console.SetOut(sw);

string[] stdin = File.ReadAllLines("input.txt");

new QuackMachine(stdin).Run();

sw.Dispose();

}

}

public class QuackMachine

{

private static ushort[] \_registers = new ushort[26];

private static int \_cursor = 0;

private static Dictionary<string, int> \_labels = new Dictionary<string, int>();

private static string[] \_program;

private static Queue<ushort> \_queue = new Queue<ushort>();

public QuackMachine(string[] input)

{

\_program = input;

this.RegisterLabels();

}

public void Run()

{

for (\_cursor = 0; \_cursor < \_program.Length; \_cursor++)

{

switch (\_program[\_cursor][0])

{

case '+': this.Sum(); break;

case '-': this.Substract(); break;

case '\*': this.Multiply(); break;

case '/': this.Divide(); break;

case '%': this.Mod(); break;

case '>': this.GetRegister(\_program[\_cursor][1]); break;

case '<': this.SetRegister(\_program[\_cursor][1]); break;

case 'P':

if (\_program[\_cursor].Length == 1)

this.PrintValueFromStack();

else

this.PrintValueFromRegister(\_program[\_cursor][1]);

break;

case 'C':

if (\_program[\_cursor].Length == 1)

this.PrintCharFromStack();

else

this.PrintCharFromRegister(\_program[\_cursor][1]);

break;

case ':': break;

case 'J': this.GoTo(\_cursor); break;

case 'Z': this.GoToIfZeroEqual(\_cursor); break;

case 'E': this.GoToIfEquals(\_cursor); break;

case 'G': this.GoToIfMoreThan(\_cursor); break;

case 'Q': this.Exit(); break;

default: \_queue.Enqueue(ushort.Parse(\_program[\_cursor])); break;

}

}

}

private void RegisterLabels()

{

for (int i = 0; i < \_program.Length; i++)

if (\_program[i][0] == ':')

\_labels.Add(\_program[i].Remove(0, 1), i);

}

private void Sum()

{

ushort a = \_queue.Dequeue();

ushort b = \_queue.Dequeue();

\_queue.Enqueue((ushort)((a + b) % 65536));

}

private void Substract()

{

ushort a = \_queue.Dequeue();

ushort b = \_queue.Dequeue();

\_queue.Enqueue((ushort)((a - b) % 65536));

}

private void Multiply()

{

ushort a = \_queue.Dequeue();

ushort b = \_queue.Dequeue();

\_queue.Enqueue((ushort)(a \* b));

}

private void Divide()

{

ushort a = \_queue.Dequeue();

ushort b = \_queue.Dequeue();

\_queue.Enqueue((ushort)(a / b));

}

private void Mod()

{

ushort a = \_queue.Dequeue();

ushort b = \_queue.Dequeue();

\_queue.Enqueue((ushort)(a % b));

}

private void GetRegister(char register)

{

ushort a = \_queue.Dequeue();

\_registers[register - 'a'] = a;

}

private void SetRegister(char register)

{

\_queue.Enqueue(\_registers[register - 'a']);

}

private void PrintValueFromStack()

{

ushort a = \_queue.Dequeue();

Console.WriteLine(a);

}

private void PrintValueFromRegister(char register)

{

Console.WriteLine(\_registers[register - 'a']);

}

private void PrintCharFromStack()

{

ushort a = \_queue.Dequeue();

Console.Write((char)(a % 256));

}

private void PrintCharFromRegister(char register)

{

Console.Write((char)(\_registers[register - 'a'] % 256));

}

private void GoTo(int index)

{

\_cursor = \_labels[\_program[index].Remove(0, 1)];

}

private void GoToIfZeroEqual(int index)

{

if (\_registers[\_program[index][1] - 'a'] == 0)

\_cursor = \_labels[\_program[index].Remove(0, 2)];

}

private void GoToIfEquals(int index)

{

if (\_registers[\_program[index][1] - 'a'] == \_registers[\_program[index][2] - 'a'])

\_cursor = \_labels[\_program[index].Remove(0, 3)];

}

private void GoToIfMoreThan(int index)

{

if (\_registers[\_program[index][1] - 'a'] > \_registers[\_program[index][2] - 'a'])

\_cursor = \_labels[\_program[index].Remove(0, 3)];

}

private void Exit()

{

\_cursor = int.MaxValue;

}

}

## Бенчмарк к задаче 5

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **№ теста** | **Результат** | **Время, с** | **Память** | **Размер входного файла** | **Размер выходного файла** |
| Max |  | 0.109 | 24506368 | 1349803 | 250850 |
| 1 | OK | 0.015 | 11001856 | 69 | 6 |
| 2 | OK | 0.031 | 11005952 | 232 | 218 |
| 3 | OK | 0.031 | 10915840 | 3 | 0 |
| 4 | OK | 0.015 | 10944512 | 100 | 19 |
| 5 | OK | 0.031 | 11800576 | 56 | 58890 |
| 6 | OK | 0.046 | 11366400 | 67 | 30000 |
| 7 | OK | 0.031 | 11431936 | 67 | 30000 |
| 8 | OK | 0.062 | 11423744 | 55 | 30000 |
| 9 | OK | 0.046 | 10952704 | 461 | 60 |
| 10 | OK | 0.031 | 11186176 | 11235 | 21 |
| 11 | OK | 0.015 | 11395072 | 23748 | 42 |
| 12 | OK | 0.031 | 12091392 | 66906 | 8905 |
| 13 | OK | 0.015 | 11055104 | 7332 | 954 |
| 14 | OK | 0.031 | 11149312 | 4611 | 602 |
| 15 | OK | 0.031 | 11591680 | 37968 | 5424 |
| 16 | OK | 0.031 | 10956800 | 14 | 2 |
| 17 | OK | 0.015 | 10944512 | 70 | 10 |
| 18 | OK | 0.031 | 11075584 | 350 | 50 |
| 19 | OK | 0.031 | 11018240 | 1750 | 250 |
| 20 | OK | 0.031 | 11145216 | 8750 | 1250 |
| 21 | OK | 0.031 | 11706368 | 43750 | 6250 |
| 22 | OK | 0.046 | 14839808 | 218750 | 31250 |
| 23 | OK | 0.031 | 11567104 | 34606 | 4721 |
| 24 | OK | 0.078 | 18477056 | 683180 | 7 |
| 25 | OK | 0.046 | 18472960 | 683102 | 0 |
| 26 | OK | 0.062 | 24506368 | 1349803 | 0 |
| 27 | OK | 0.078 | 19132416 | 491572 | 247791 |
| 28 | OK | 0.062 | 19095552 | 491488 | 249618 |
| 29 | OK | 0.062 | 19144704 | 491600 | 249600 |
| 30 | OK | 0.062 | 19144704 | 491502 | 250850 |
| 31 | OK | 0.062 | 19103744 | 491416 | 249477 |
| 32 | OK | 0.078 | 19111936 | 491520 | 250262 |
| 33 | OK | 0.078 | 19103744 | 491317 | 246859 |
| 34 | OK | 0.046 | 19165184 | 491514 | 248199 |
| 35 | OK | 0.109 | 19152896 | 491557 | 249601 |