Создать класс для объекта стек. Стек хранит целые числа. Имеет характеристики: наименование (строка, не более 10 символов) и размер (целое). Размер стека больше или равно 1. Функционал стека:  
-    добавить элемент и вернуть признак успеха (логическое);  
-    извлечь элемент и вернуть признак успеха (логическое);  
-    получить имя стека (строка);  
-    получить размер стека (целое);  
-    получить текущее количество элементов в стеке (целое).   
В классе определить параметризированный конструктор, которому передается имя стека и размер. При переполнении стека очередной элемент не добавлять и определяется соответствующий признак успеха.  
В основной программе реализовать алгоритм:  
1.    Ввести имя и размер для первого стека.  
2.    Создать объект первого стека.  
3.    Ввести имя и размер для второго стека.  
4.    Создать объект второго стека.  
5.    В цикле:  
     5.1.    Считывать очередное значение элемента.  
     5.2.    Добавлять элемент в первый стек, при переполнении завершить цикл.  
     5.3.      Добавлять элемент во второй стек, при переполнении завершить цикл.  
6.    Построчно вывести содержимое стеков.

**Входные данные**

Первая строка:  
«имя стека 1» «размер стека»  
Вторая строка:  
«имя стека 2» «размер стека»  
Третья строка:  
Последовательность целых чисел, разделенных пробелами, в количестве не менее чем размер одного из стеков + 1.

**Выходные данные**

Первая строка:  
«имя стека 1» «размер»  
Вторая строка:  
«имя стека 2» «размер»  
Третья строка:  
«имя стека 1» «имя стека 2»  
Каждое имя стека в третьей строке занимает поле длины 15 позиции и прижата к левому краю.  
Четвертая строка и далее построчно, вывести все элементы стеков:  
«значение элемента стека 1» «значение элемента стека 2»  
Вывод значений элементов стеков производиться последовательным извлечением.  
Каждое значение занимает поле из 15 позиции и прижата к правому краю.

Используя потоке Ввода/Вывода - cin/cout

Используя bool stacks::push(int value) для добавить элемент и вернуть признак успеха (логическое).  
Используя bool stacks::peek() для извлечь элемент и вернуть признак успеха (логическое).  
Используя string stacks::getNameStack() для получить имя стека (строка).  
Используя int stacks::getCapacity() для получить размер стека (целое).  
Используя int stacks::getSize() для получить текущее количество элементов в стеке (целое).

stacks::stacks(string nameStack, int capacity)

|  |  |  |  |
| --- | --- | --- | --- |
| № шага | Предикат | Действе | № перехода |
| 1 |  | this->capacity = capacity; | 2 |
| 2 |  | this->nameStack = nameStack; | Ø |

bool stacks::isFull()

|  |  |  |  |
| --- | --- | --- | --- |
| № шага | Предикат | Действе | № перехода |
| 1 | if (top >= capacity - 1) | return true; | Ø |
| else | return false; | Ø |

bool stacks::isEmpty()

|  |  |  |  |
| --- | --- | --- | --- |
| № шага | Предикат | Действе | № перехода |
| 1 | if (top == -1) | return true; | Ø |
| else | return false; | Ø |

bool stacks::push(int value)

|  |  |  |  |
| --- | --- | --- | --- |
| № шага | Предикат | Действе | № перехода |
| 1 | if (isFull()) | return false; | Ø |
| else | ++top; | 2 |
| 2 |  | stack[top] = value; | 3 |
| 3 |  | return true; | Ø |

bool stacks::peek()

|  |  |  |  |
| --- | --- | --- | --- |
| № шага | Предикат | Действе | № перехода |
| 1 | if (isEmpty() == false) | return true; | Ø |
| else | return false; | Ø |

void stacks::showStack(int i)

|  |  |  |  |
| --- | --- | --- | --- |
| № шага | Предикат | Действе | № перехода |
| 1 | if (i >= 0 && i < capacity) | cout << setw(15) << right << stack[i]; | Ø |

string stacks::getNameStack()

|  |  |  |  |
| --- | --- | --- | --- |
| № шага | Предикат | Действе | № перехода |
| 1 |  | return nameStack; | Ø |

int stacks::getCapacity()

|  |  |  |  |
| --- | --- | --- | --- |
| № шага | Предикат | Действе | № перехода |
| 1 |  | return capacity; | Ø |

int stacks::getSize()

|  |  |  |  |
| --- | --- | --- | --- |
| № шага | Предикат | Действе | № перехода |
| 1 |  | return top + 1; | Ø |

int main()

|  |  |  |  |
| --- | --- | --- | --- |
| № шага | Предикат | Действе | № перехода |
| 1 |  | string nameStack1, nameStack2; | 2 |
| 2 |  | int capacity1, capacity2; | 3 |
| 3 |  | cin >> nameStack1 >> capacity1; | 4 |
| 4 |  | stacks stack1 = stacks(nameStack1, capacity1); | 5 |
| 5 |  | cin >> nameStack2 >> capacity2; | 6 |
| 6 |  | stacks stack2 = stacks(nameStack2, capacity2); | 7 |
| 7 | while (true) |  | 8 |
| false |  | 12 |
| 8 |  | int value; | 9 |
| 9 |  | cin >> value; | 10 |
| 10 | if (stack1.isFull() == false) | stack1.push(value); | 11 |
| else | break; | 12 |
| 11 | if (stack2.isFull() == false) | stack2.push(value); | 7 |
| else | break; | 12 |
| 12 |  | cout << stack1.getNameStack() << " " << stack1.Capacity() << endl; | 13 |
| 13 |  | cout << stack2.getNameStack() << " " << stack2.Capacity() << endl; | 14 |
| 14 |  | cout << setw(15) << left << stack1.getNameStack() << setw(15) << left << stack2.getNameStack(); | 15 |
| 15 |  | int distance = capacity1 - capacity2; | 16 |
| 16 | if (distance <= 0) |  | 17 |
| else |  | 20 |
| 17 | for (int i = capacity1 - 1; i >= 0; i--) | cout << endl; | 18 |
| i < 0 |  | Ø |
| 18 |  | stack1.showStack(i); | 19 |
| 19 |  | stack2.showStack(i); | 17 |
| 20 | for (int i = capacity2; i >= 0; i--) | cout << endl; | 21 |
| i < 0 |  | Ø |
| 21 |  | stack1.showStack(i); | 22 |
| 22 |  | stack2.showStack(i-1); | 20 |