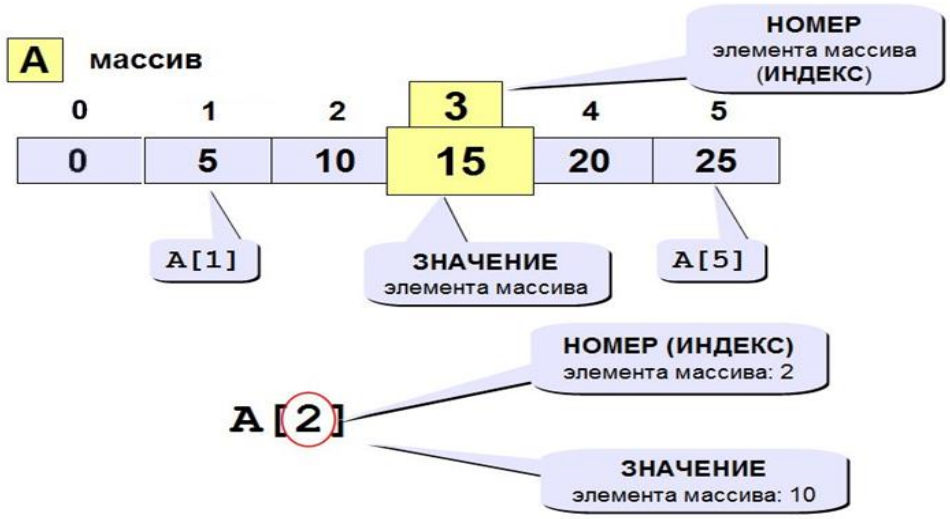
Массивы в C++

Немного теории

При решении многих задач возникает необходимость хранить всю последовательность или ее часть. Конечно, можно завести много переменных для этого: a, b, c и т.д. Но в этом случае **а)** быстро кончатся буквы, **б**) неизвестно, сколько именно этих переменных нам понадобится, причем от запуска к запуску программы, это количество должно меняться, но главное **в)** для обработки последовательности нельзя будет использовать циклы!

Для хранения большого количества данных одного типа используют массивы. Массив – это переменная, в которой хранится много значений. При создании массива указывается его размер. Каждый элемент массива определяется именем, совпадающим с именем массива, а также индексом. Индекс — это величина, характеризующая положение элемента в массиве.



В языке C++ нумерация элементов массива начинается с нуля, таким образом номер последнего элемента массива соответствует значению n-1, где n – количество элементов массива.

Решение задач с массивами часто связано с чтением и выводом большого количества данных, что может вызывать проблемы с временем выполнения программы. Для ускорения работы операторов ***cin*** и ***cout*** можно использовать следующие команды:

```
ios::sync_with_stdio(0); cin.tie(0); cout.tie(0);
```

Линейный массив (vector)

Линейный массив (**vector**) – ведет себя как динамически расширяемый массив, с элементами которого можно работать как с элементами обычного одномерного массива. Требуется подключение библиотеки: **#include <vector>**

Создание массива

vector <тип элементов> имя;

1 способ

```
cin >> n;
vector <int> a (n);          //создать вектор из n элементов
```

2 способ Количество элементов указывать не обязательно

```
vector <int> a;              // создать пустой вектор
```

Можно создать массивы «чего угодно», объектов любых типов. Например, можно создать массив строк.

Ввод (считывание) элементов массива

```
cin >> n;
vector <int> a (n);
for (int i = 0; i < n; i++) {
       cin >> a[i];
    }
```

2 способ

```
cin >> n;
vector <int> a;     // создать пустой вектор
for (int j=0; j<n; j++) {
        cin >> x;
        a.push_back(x);     //ввод
    }
```

Данный способ формирования вектора является не самым оптимальным, так как при добавлении нового элемента требуется проверка выделенной памяти и, при необходимости, ее расширение. Если возможно, лучше сразу зарезервировать нужное количество памяти.

Вывод элементов массива

```
for (int i = 0; i < n; i++) {
      cout << a[i] << ' ';
    }
```

Заметьте, что счетчик нашего цикла начинается с нуля, а заканчивается n-1. Если вместо оператора строгого неравенства — i < n использовать оператор «меньше, либо равно» — i ≤ n, то на последней итерации программа обратится к несуществующему элементу массива . Это может привести к ошибкам сегментации и аварийному завершению программы. Будьте внимательны — подобные ошибки бывает сложно отловить.

Сумма элементов массива

```
long long s=0;
for(i=0; i<n; i++){
   s+=a[i];                 //s=s+a[i];
  }
```

Поиск максимума

```
int imax = 0;
for (int i = 0; i < n; i++){
      if (a[i] > a[imax]) {imax = i;}
    }
cout << a[imax];
```

Поиск индекса заданного числа в массиве

```
int r = -1; // индекс элемента, равного x
int i = 0;
while (_____ && i < n){
   if (a[i] == x) r = i;
   i++;
  }
```

Если r после цикла останется -1 – элемент не найден.

Чтобы не рассматривать отдельно конец последовательности, когда элемент в массиве отсутствует, можно дополнить его фиктивным «барьерным» элементом (для этого в массиве должна быть обязательно хотя бы одна «лишняя» ячейка):

```
a[n] = x;
int r = 0;
while (a[r]!= x) {
  r++;
 }
```

Теперь мы в любом случае «найдем» x. Отсутствие элемента теперь не особый случай – x не обнаружен в последовательности, если r станет равным n.

Задача №69. Переставить элементы в обратном порядке

Напишите программу, которая переставляет элементы массива в обратном порядке без использования дополнительного массива. Программа должна считать массив, поменять порядок его элементов, затем вывести результат (просто вывести элементы массива в обратном порядке – недостаточно!)

Входные данные

Сначала задано число N — количество элементов в массиве (1≤N≤35). Далее через пробел записаны N чисел — элементы массива. Массив состоит из целых чисел.

Выходные данные

Необходимо вывести массив, полученный после перестановки элементов.

Примеры

Входные данные

```
6
4 5 3 4 2 3
```

Выходные данные

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

1 способ.

Надо переставить его элементы, так чтобы нулевой элемент встал на N-1-е место, а N-1-й на нулевое.

a[0]↔a[n-1]

a[1]↔a[n-2]

…

a[i]↔a[__?__]

```
for (int i = 0; i < _____; i++) {
      swap(a[i], a[N - i - 1];
   }
```

Подумайте сами условие выхода из цикла

2 способ

Можно воспользоваться интересным приемом – **ввести два указателя**.

Заведем две переменных **left** и **right**, в которых будем хранить индексы обмениваемых ячеек.

Они вначале будут равны 0 и N-1 соответственно.

int left = 0;

int right = N-1;

В цикле будем обменивать ячейки с индексами **left** и **right** и переходить к следующим.

Удобно использовать цикл while:

```
while(_________________) {
    swap(a[left], a[right]);
    left++;
    right--;
  }
```

Подумайте сами условие выхода из цикла

Методы класса vector:

vector < int > v	–	описание динамического массива
v.push_back(x)	–	добавление элемента в конец вектора
v.size()	–	возвращает количество элементов в векторе
v.empty()	–	возвращает логическое значение: пуст ли вектор?
v.capacity()	–	возвращает число элементов, которые могут поместиться в векторе без расширения
v.resize(n)	–	изменяет размер вектора
v.reserve(n)	–	резервирует память под n элементов
v.assign(k, x)	–	присваивает первым k элементам значение x
v.clear()	–	очищает содержимое вектора
v.insert(v.begin()+k, x)	–	вставка элемента x в к-ю позицию

Сортировка массива – это процесс распределения всех элементов массива в определенном порядке.

Если в решении задачи требуется просто отсортировать массив по возрастанию, конечно, следует использовать стандартную функцию sort

Для использования встроенной функции сортировки необходимо подключить библиотеку алгоритмов:

#include <algoritm>

Сортировка по возрастанию	**sort(a.begin(), a.end());**
Сортировка по убыванию	**sort(a.begin(), a.end(), greater<int>());**

Сортировка выбором, сортировка пузырьком

АЛГОРИТМЫ ОБРАБОТКИ МАССИВОВ

ЗАДАЧИ ДЛЯ САМОСТОЯТЕЛЬНОГО РЕШЕНИЯ

Для решения задач необходимо зарегистрироваться на сайте
https://informatics.msk.ru/
Задачи указаны с номерами.

Задача №2768 Прыжки с трамплина

На соревнованиях по прыжкам на лыжах с трамплина техника прыжка оценивается пятью судьями. Каждый судья ставит оценку от 1 до 20, после чего одна наименьшая и одна наибольшая оценки отбрасываются. Вам нужно написать программу, которая будет демонстрировать результаты прыжка для телетрансляции.

Она должна выводить пять оценок, которые поставили судьи, не меняя их порядка, а затем их сумму, и при этом брать в скобки те оценки, которые не учитываются при расчете суммы

Входные данные

На вход подается 5 натуральных чисел от 1 до 20, разделенных пробелом.

Выходные данные

Выведите те же числа в том же порядке, взяв в скобки минимальное (а если их несколько – самое левое из них) и максимальное (а если их несколько – самое правое из них) число, а также сумму всех чисел, не взятых в скобки. Все числа (включая сумму) должны быть напечатаны в одной строке и разделены одним пробелом (внутри скобок пробелов быть не должно). Перед суммой должен стоять знак равенства, отделенный слева и справа одним пробелом. Порядок оценок должен быть такой же, как и во входных данных.

Примеры

Входные данные
10 11 10 11 10
Выходные данные
(10) 11 10 (11) 10 = 31

Задача №1572 Уникальные элементы

На вход программе сначала подается значение n ≤ 100 — количество элементов в массиве. В следующей строке входных данных расположены сами элементы массива — целые числа, по модулю не превосходящие 30000. Распечатайте только те значения элементов массива, которые встречаются в нем ровно один раз. Элементы следует распечатывать в том порядке, в котором они встречаются в массиве. Создавать новые массивы нельзя.

Примеры

Входные данные
8

4 3 5 2 5 1 3 5

Выходные данные

4 2 1

Задача №3017. Последовательности

Рассмотрим последовательности чисел. Первая последовательность состоит из одного числа K. Каждая следующая последовательность чисел описывает предыдущую по такому правилу.

Просматриваем описываемую последовательность слева направо и разбиваем на отрезки, состоящие из подряд идущих равных чисел (причем все идущие подряд одинаковые числа всегда объединяем в один отрезок). Далее каждый такой отрезок описываем двумя числами — первое число говорит, сколько раз повторяется одно и то же число, второе число говорит, какое именно число повторяется. Записываем эти пары последовательно в соответствии с отрезками слева направо, и получаем новую последовательность (см. примеры ниже).

Например, для K=2 последовательности получатся такими:

№	**Последовательность**	**Как ее читать** (слова в описании соответствуют числам текущей последовательности слева направо, и описывают предыдущую последовательность)
1	2	Исходная последовательность
2	1 2	Одна «двойка»
3	1 1 1 2	Одна «единица», одна «двойка»
4	3 1 1 2	Три «единицы», одна «двойка»
5	1 3 2 1 1 2	Одна «тройка», две «единицы», одна «двойка»
6	1 1 1 3 1 2 2 1 1 2	Одна «единица», одна «тройка», одна «двойка», две «единицы», одна «двойка»

Напишите программу, которая по исходному числу K напечатает N-ую получающуюся последовательность.

Входные данные

Вводится число K ($1 \le K \le 9$) и число N ($1 \le N \le 15$).

Выходные данные

Ваша программа должна печатать N-ую последовательность, полученную из начальной последовательности, состоящей из одного числа K. Числа при выводе следует разделять пробелами.

Примеры

Входные данные

2
6

Выходные данные

1 1 1 3 1 2 2 1 1 2

Задача №1461. Шарики

В одной компьютерной игре игрок выставляет в линию шарики разных цветов. Когда образуется непрерывная цепочка из трех и более шариков одного цвета, она удаляется из линии. Все шарики при этом сдвигаются друг к другу, и ситуация может повториться.

Напишите программу, которая по данной ситуации определяет, сколько шариков будет "уничтожено". Естественно, непрерывных цепочек из трех и более одноцветных шаров в начальный момент может быть не более одной.

Входные данные

Сначала вводится количество шариков в цепочке (не более 1000) и цвета шариков (от 0 до 9, каждому цвету соответствует свое целое число).

Выходные данные

Требуется вывести количество шариков, которое будет "уничтожено".

Примеры

Входные данные

5 1 3 3 3 2

Выходные данные

3

Задача №3016. Списывание

На контрольной работе N учеников сидят в ряд. Для каждого ученика известно, какую оценку он получил бы, если бы писал эту контрольную самостоятельно (оценка — это число от 2 до 5). Однако ученики могут писать контрольную не только самостоятельно, но и списывать у своего соседа, но только если сосед пишет контрольную самостоятельно. В этом случае списывающий получит такую же оценку, какую получит тот, у кого он списал.

А именно (правила применяются строго в указанном порядке):

Школьники, которые знают материал на 5, будут писать контрольную самостоятельно.

Школьник, который знает материал на 4, если он сидит рядом с тем, кто знает на 5, будет списывать у него, а в противном случае будет писать самостоятельно.

Школьник, который знает на 3, если он сидит рядом с тем, кто знает на 5, будет списывать у него. Если среди его соседей знающего на 5 нет, но есть тот, кто знает на 4, и при этом пишет самостоятельно, то троечник будет списывать у него. В противном случае будет писать самостоятельно.

Аналогично школьник, знающий на 2 — из соседей, которые пишут самостоятельно, выберет того, кто знает лучше, и спишет у него. А если таких нет (или оба его соседа также знают на 2), то будет писать самостоятельно.

Определите, кто какую оценку в итоге получит.

Входные данные

Вводится число N (1≤ N≤ 10) - количество учеников, и далее последовательность из N чисел, описывающая, кто на какую оценку может написать контрольную, если будет писать самостоятельно.

Выходные данные

Выведите N чисел - оценки, которые получат ученики за контрольную.

Примечания к примерам тестов

1. Первый и пятый ученики будут писать самостоятельно. Второй спишет у первого, а четвертый — у пятого (в итоге также получат пятерки). Третьему не у кого списывать, так как его соседи будут писать работу не самостоятельно.

2. Второй и четвертый спишут у третьего, пятый — у шестого.

Примеры

Входные данные

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

Выходные данные

```
5 5 3 5 5
```

Задача №111160. Обувной магазин

В обувном магазине продается обувь разного размера. Известно, что одну пару обуви можно надеть на другую, если она хотя бы на три размера больше. В магазин пришел покупатель. Требуется определить, какое наибольшее количество пар обуви сможет предложить ему продавец так, чтобы он смог надеть их все одновременно.

Входные данные

Сначала вводится размер ноги покупателя (обувь меньшего размера он надеть не сможет), в следующей строке — размеры каждой пары обуви в магазине через пробел. Размер — натуральное число, не превосходящее 100.

Выходные данные

Выведите единственное число — максимальное количество пар обуви, которое сможет надеть покупатель.

Примеры

Входные данные

```
26
30 35 40 41 42
```

Выходные данные

```
3
```

Задача №734. Такси

После затянувшегося совещания директор фирмы решил заказать такси, чтобы развезти сотрудников по домам. Он заказал N машин – ровно столько, сколь у него сотрудников. Однако когда они подъехали, оказалось, что у каждого водителя такси свой тариф за 1 километр.

Директор знает, какому сотруднику сколько километров от работы до дома (к сожалению, все сотрудники живут в разных направлениях, поэтому нельзя отправить двух сотрудников на одной машине). Теперь директор хочет определить, какой из сотрудников на каком такси должен поехать домой, чтобы суммарные затраты на такси (а их несет фирма) были минимальны.

Входные данные

Первая строка входных данных содержит натуральное число N ($1 \le N \le 1000$) – количество сотрудников компании (совпадающее с количеством вызванных машин такси). Далее записано N чисел, задающих расстояния в километрах от работы до домов сотрудников компании (первое число – для первого сотрудника, второе – для второго и т.д.). Все расстояния – положительные целые числа, не превышающие 1000. Далее записано еще N чисел – тарифы за проезд одного километра в такси (первое число – в первой машине такси, второе – во второй и т.д.). Тарифы выражаются положительными целыми числами, не превышающими 10000.

Выходные данные

Программа должна вывести N чисел. Первое число – номер такси, в которое должен сесть первый сотрудник, второе число – номер такси, в которое должен сесть второй и т.д., чтобы суммарные затраты на такси были минимальны. Если вариантов рассадки сотрудников, при которых затраты минимальны, несколько, выведите любой из них.

Примеры

Входные данные

```
3
10 20 30
50 20 30
```

Выходные данные

```
1 3 2
```

Задача №3594. Злые свинки или Anti Angry Birds

Вы никогда не задумывались, почему в Angry Birds у птиц нет крыльев? Тем же вопросом задались разработчики новой игры. В их версии смысл игры прямо противоположен Angry Birds: зеленая свинка стреляет по злым птицам из лазерного ружья (завязка явно не хуже исходной игры).

Птицы в игре представляются точками на плоскости. Выстрел сбивает только ближайшую птицу, находящуюся на линии огня. При этом сбитая птица, падая, сбивает всех птиц, находящихся ровно под ней. Две птицы не могут находиться в одной точке. По заданному расположению птиц необходимо определить, какое минимальное количество выстрелов необходимо, чтобы все птицы были сбиты.

Входные данные

Первая строка входного файла содержит единственное целое число N ($1 \le N \le 1000$)— количество птиц.

Следующие N строк содержат по два натуральных числа каждая x_i, y_i — координаты *i*-ой птицы ($0 < x, y \le 10^9$). Свинка находится в точке с координатами (0, 0).

Выходные данные

Единственная строка выходного файла должна содержать одно целое число — минимальное количество выстрелов, необходимое для того, чтобы сбить всех птиц.

Примеры

Входные данные

```
6
1 1
2 2
3 3
2 1
3 2
3 1
```

Выходные данные

```
3
```

Задача №113612. Подборка новостей

Игорь читает новостную ленту в своей любимой социальной сети, состоящую из n последовательно расположенных записей. Каждая запись в ленте имеет свою характеристику — позитивность a_i, заданную натуральным числом. После прочтения i-й записи настроение Игоря улучшается на a_i.

Игорь читает записи в том порядке, в котором они показаны, не пропуская никакие из них. Приложение социальной сети устроено так, что после просмотра последней записи Игорь перемещается в начало ленты и видит первую запись. Игорь может начать просмотр новостной ленты с любой из записей и прочитать подряд любое количество записей, не превосходящее n.

Игорь чувствует себя счастливым, если его настроение после прочтения новостей улучшилось хотя бы на p. Он хочет почувствовать себя счастливым, прочитав как можно меньше записей, ведь он уже опаздывает на олимпиаду! Помогите ему выбрать запись, с которой следует начать просмотр, и количество записей, которые нужно просмотреть, или определите, что записей в ленте недостаточно, чтобы стать счастливым.

Входные данные

Первая строка содержит два числа n и p ($1 \le n \le 1000$, $1 \le p \le 10^7$) — количество записей в новостной ленте и величину, на которую Игорь хочет увеличить своё настроение.

Вторая строка содержит n целых неотрицательных чисел a_i ($1 \le a_i \le 10^4$) — позитивности записей в ленте. Соседние числа разделены ровно одним пробелом.

Выходные данные

Если Игорь сможет стать счастливым, выведите два числа — номер записи k, с которой следует начать просмотр, и количество записей c, которые нужно посмотреть. Если возможных ответов с минимальным с несколько, выведите тот, у которого k минимально.

Если же решения нет, выведите - 1.

Примеры

Входные данные

```
9 10
1 2 3 4 5 4 3 2 1
```

Выходные данные

```
3 3
```