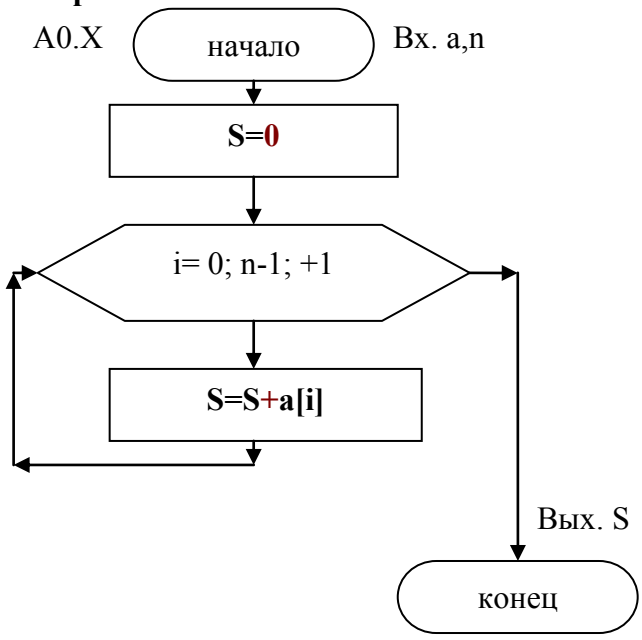
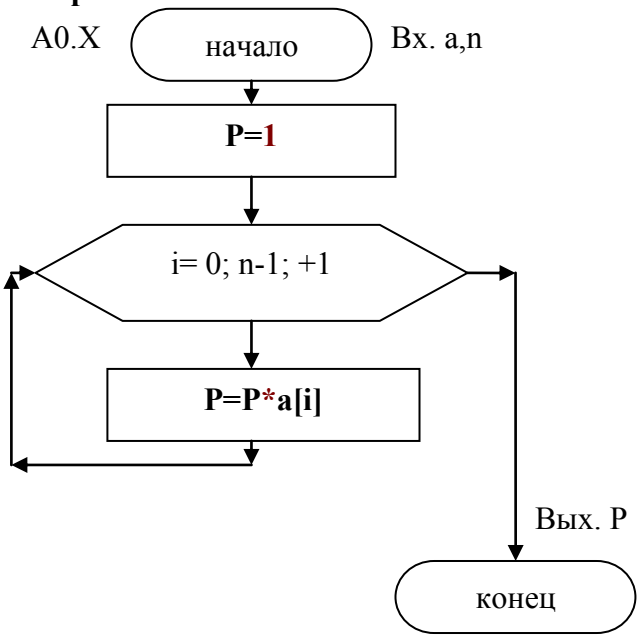
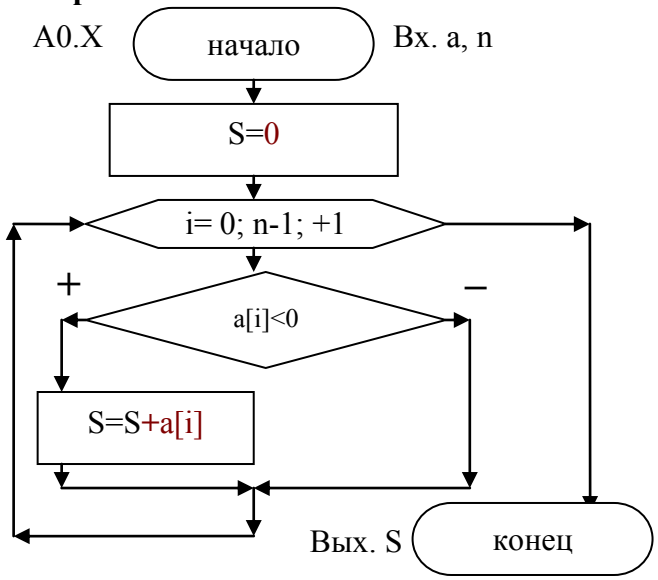
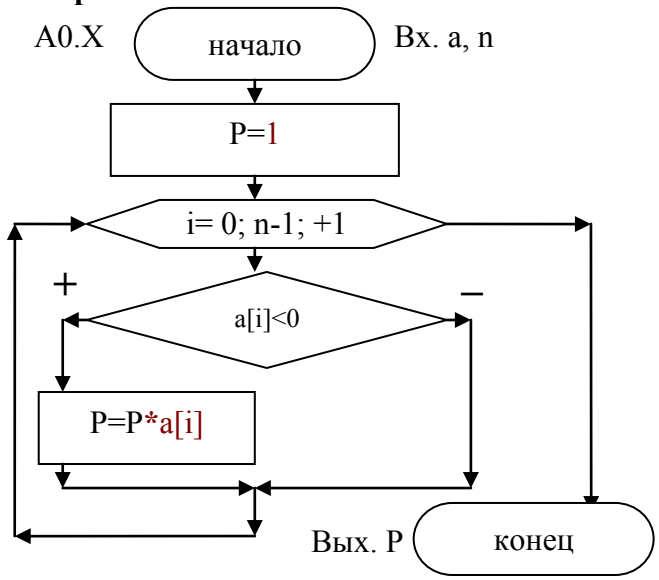
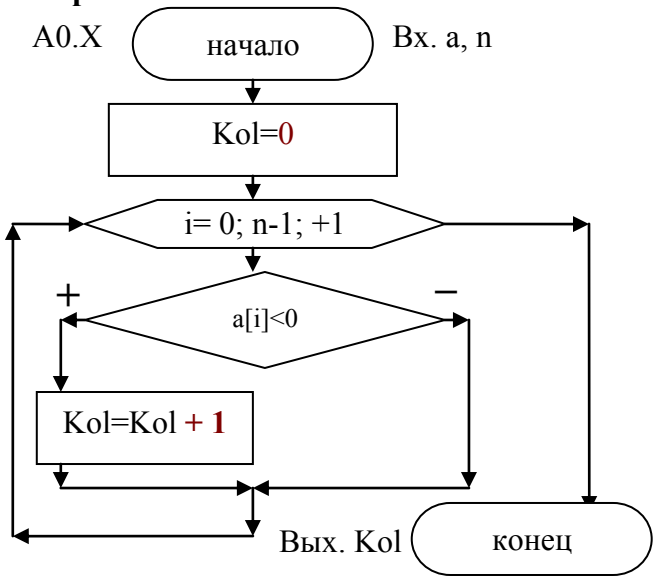
Сумма. Найти сумму S элементов массива a(n)	**Произведение**. Найти произведение P элементов массива a(n)
Метод. Отделим ввод-вывод от обработки и рассмотрим только обработку. Пусть i – индекс текущего элемента массива a $s = \sum_{i=0}^{n-1} a[i] = 0 + \sum_{i=0}^{n-1} a[i] =$ $= (...((\ 0 + a[0]) + a[1]) + ...) + a[n-1]$ Примем за начальное значение суммы число 0 (S=0), т.к. S=0+S Теперь по очереди с первого по последний элемент (i=0..n-1), учитывая индексацию с 0: Просматриваем элементы И добавляем их к сумме (S=S+a[i]) Полученная сумма будет искомой.	**Метод**. Отделим ввод-вывод от обработки и рассмотрим только обработку. Пусть i – индекс текущего элемента массива a $p = \prod_{i=0}^{n-1} a[i] = 1 * \prod_{i=0}^{n-1} a[i] =$ $= (...(\ 1 * a[0]) * a[1]) * ...) * a[n-1]$ Примем за начальное значение произведения число 1 (P=1), т.к. P=1*P Теперь по очереди с первого по последний элемент (i=0..n-1) , учитывая индексацию с 0: Просматриваем элементы И изменяем произведение (P=P*a[i]) Полученное произведение будет искомым.
Алгоритм.  	**Алгоритм**. 
Программный код (фрагмент) **S=0;** **for(i=0; i<n; i++)** **S=S+a[i]; // S+=a[i];**	**Программный код (фрагмент)** **P=1;** **for(i=0; i<n; i++)** **P:=P*a[i]; // или P*=a[i];**

Сумма с условием Найти сумму S отрицательных элементов массива a(n)	**Произведение с условием** Найти произведение P отрицательных элементов массива a(n)	**Количество с условием** Найти количество Kol отрицательных элементов массива a(n)
Метод. Аналогичен выше рассмотренному. Отделим ввод-вывод от обработки и рассмотрим только обработку. Пусть i – индекс текущего элемента массива a. Примем за начальное значение суммы число 0 (S=0), т.к. S=0+S Теперь по очереди с первого по последний элемент (i=0..n-1) , учитывая индексацию с 0: Просматриваем элементы и Если отрицателен (a[i]<0), то добавляем его к сумме (S=S+a[i]) Полученная сумма будет искомой.	**Метод**. Аналогичен выше рассмотренному. Отделим ввод-вывод от обработки и рассмотрим только обработку. Пусть i – индекс текущего элемента массива a. Примем за начальное значение число 1 (P=1), т.к. P=1*P Теперь по очереди с первого по последний элемент (i=0..n-1) , учитывая индексацию с 0: Просматриваем элементы и Если отрицателен (a[i]<0), то Изменяем произведение (P=P*a[i]) Полученное произведение искомое.	**Метод**. Отделим ввод-вывод от обработки и рассмотрим только обработку. Пусть i – индекс текущего элемента массива a. Примем за начальное значение кол-ва число 0 (Kol=0), т.к. пока найдено 0 элементов. Теперь по очереди с первого по последний элемент (i=0..n-1) , учитывая индексацию с 0: Просматриваем элементы и Если отрицателен (a[i]<0), то Изменяем кол-во (Kol=Kol + 1) Еще один нашли! Полученное кол-во искомое.
Алгоритм.  	**Алгоритм**.  	**Алгоритм** 
Программный код (фрагмент) **S=0;** **for(i=0; i<n; i++)** **if (a[i]<0)** **S=S+a[i];**	**Программный код (фрагмент)** **P=1;** **for(i=0; i<n; i++)** **if (a[i]<0)** **P=P*a[i];**	**Программный код (фрагмент)** **Kol=0;** **for(i=0; i<n; i++)** **if (a[i]<0)** **Kol=Kol+1;**

Минимум

Найти значение Amin и номер (номер = индекс Kmin+1) первого из минимальных элементов массива a(n)

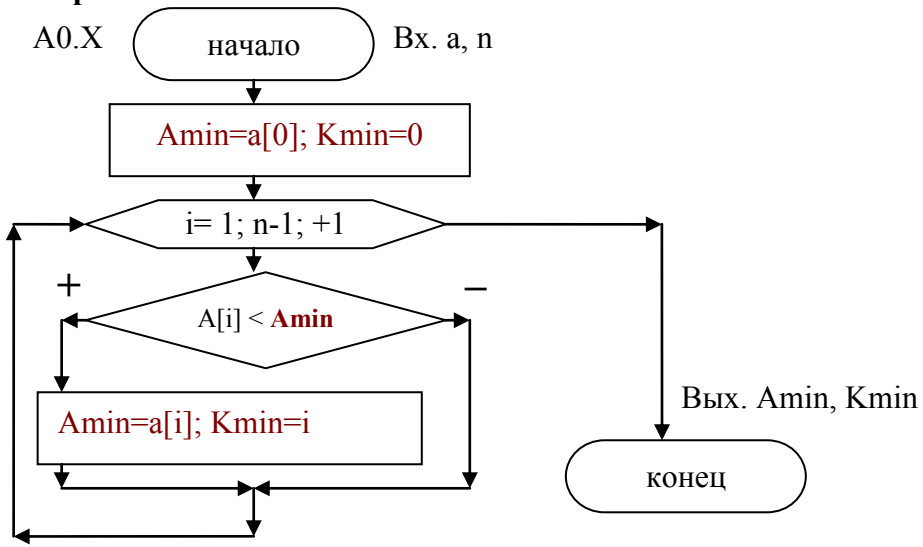
Метод.
Отделим ввод-вывод от обработки и рассмотрим только обработку.
Пусть i – индекс текущего элемента массива a.

Примем за начальное значение минимума первый элемент (Amin=a[0]; Kmin=0;), может меньшего и не найдем.

Теперь по очереди со второго по последний элемент (i=1...n-1):
- Просматриваем элементы и
- Если он меньше текущего минимума (a[i]<Amin), то
  - Изменяем минимум
  - (Amin=a[i]; Kmin=i;)
  - Новый минимум!

Полученный минимум – искомый. Номер на 1 больше , чем индекс надо вывести.

Алгоритм.



Программный код (фрагмент)

```
Amin=a[0]; Kmin=0;
for(i=1; i<n; i++)
        if (a[i] < Amin ){
                Amin=a[i]; Kmin=i;
        }
```

Максимум

Найти значение AMax и номер (номер = индекс KMax +1) первого из максимальных элементов массива a(n)

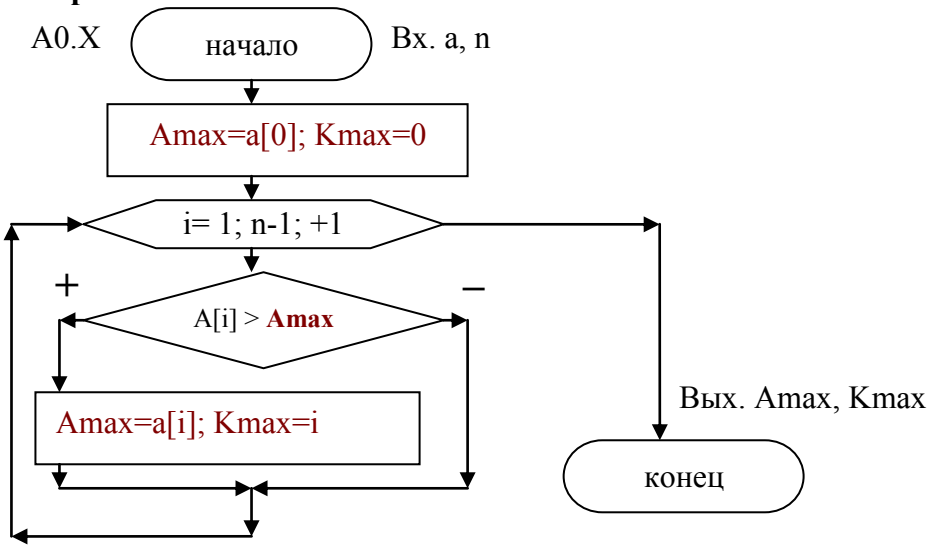
Метод.
Отделим ввод-вывод от обработки и рассмотрим только обработку.
Пусть i – индекс текущего элемента массива a.

Примем за начальное значение максимума первый элемент (Amax=a[0]; Kmax=0;), может большего и не найдем.

Теперь по очереди со второго по последний элемент (i=1...n-1):
- Просматриваем элементы и
- Если он больше текущего максимума (a[i]>Amax), то
  - Изменяем максимум
  - (Amax=a[i]; Kmax=i;)
  - Новый максимум!

Полученный максимум – искомый. Номер на 1 больше , чем индекс надо вывести.

Алгоритм.



Программный код (фрагмент)

```
Amax=a[0]; Kmax=0;
for(i=0; i<n; i++)
        if (a[i] > Amax ){
                Amax=a[i]; Kmax=i;
        }
```

Поиск максимума с условием

Найти в a(n) максимальный элемент (номер (*KMax+1*) и значение (*Amax*)) среди отрицательных.
Если невозможно (*Est*) найти ни одного отрицательного элемента, вывести сообщение об этом.
Если элементов с максимальным значением несколько, то найти номер первого из них.

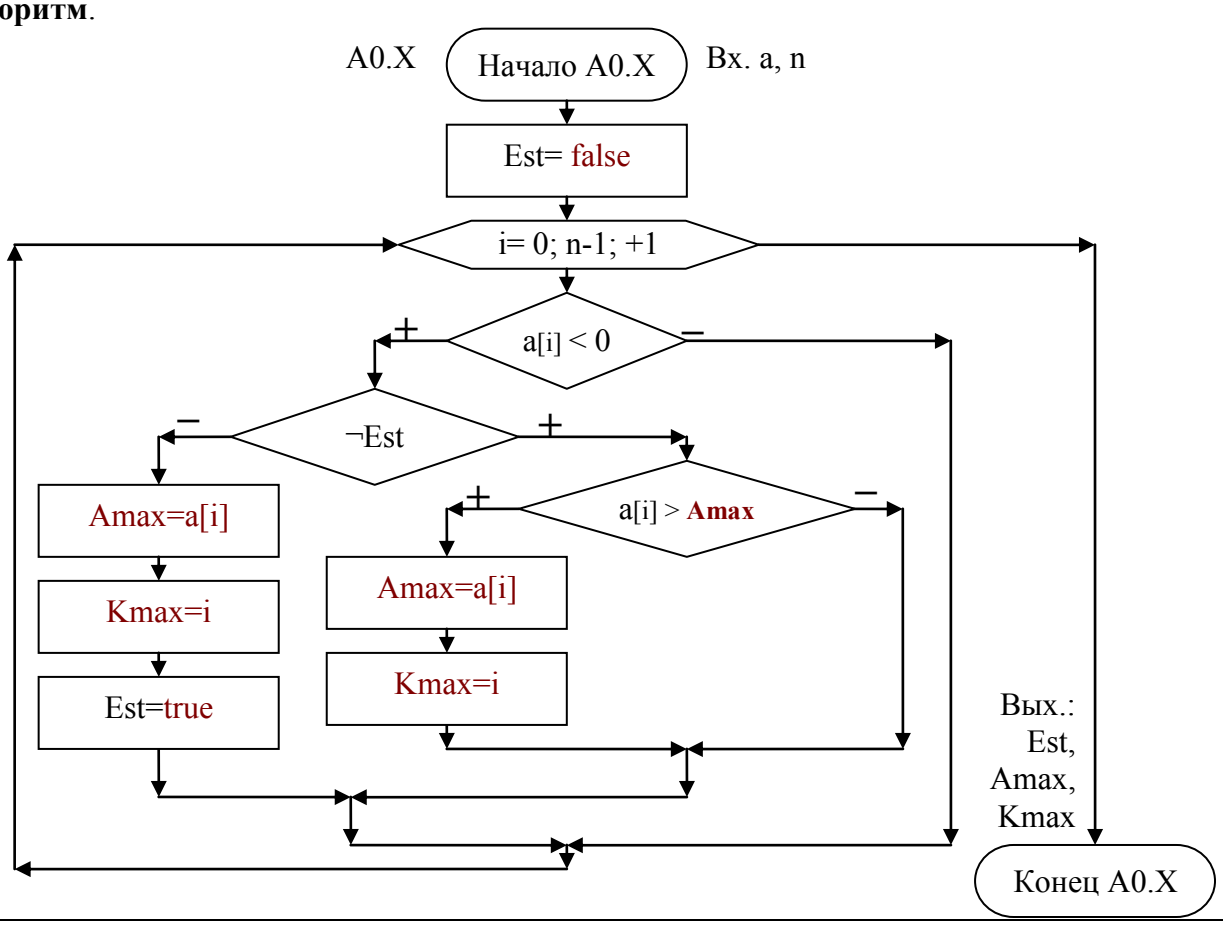
Метод.

Отделим ввод-вывод от обработки и рассмотрим только обработку.
Пусть i – индекс текущего элемента массива a.
Пусть Est=true, если найден хотя бы один отрицательный элемент и известно текущее значение максимума, и Est=false, если пока не найдено ни одного отрицательного элемента.
Начальное значение Est=true;
Перебираем по очереди со первого по последний элемент (i=0..n-1), учитывая индексацию с 0:

Если i-ый элемент отрицателен (a[i]<0),
То Если Est=False, то найдено начальное значение максимума: Amax=a[i]; Kmax=i; Est=true;
Иначе есть с чем сравнить: Если a[i] больше текущего максимума (a[i]>Amax), то
Изменяем максимум: Amax=a[i]; Kmax=i;
Новый максимум!

Найденный (если Est=True) после просмотра всех элементов максимум – искомый.
Если не найдено ни одного отрицательного элемента, то максимум не найден.

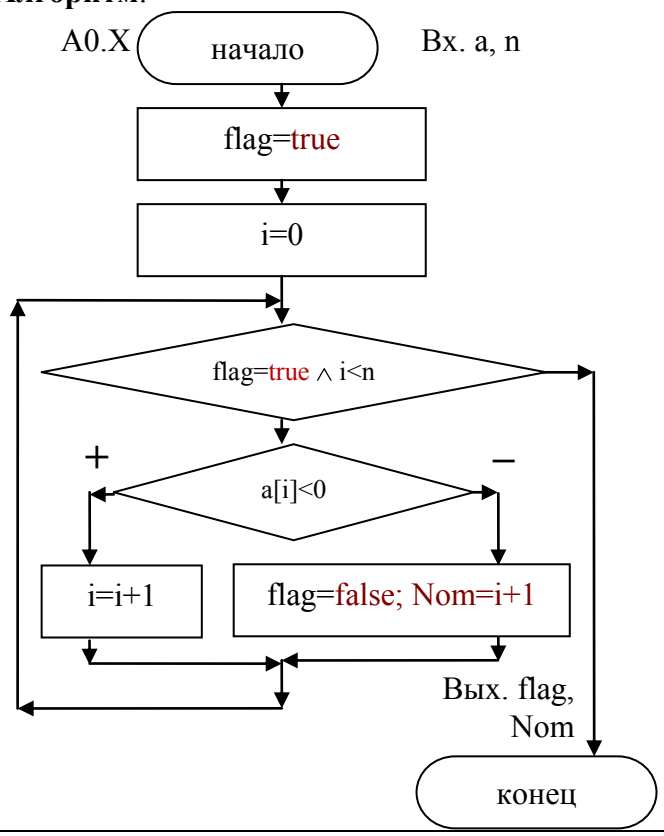
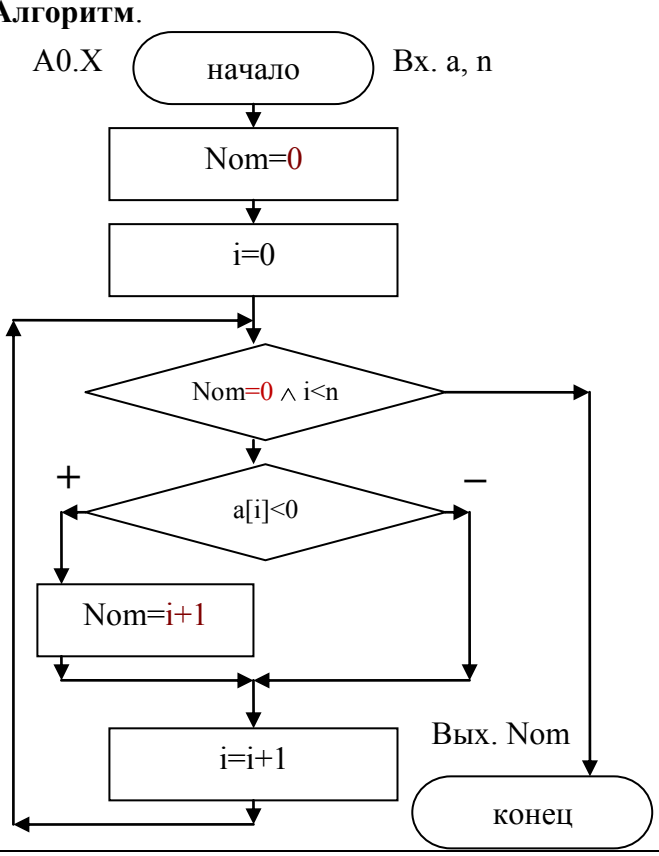
Алгоритм.



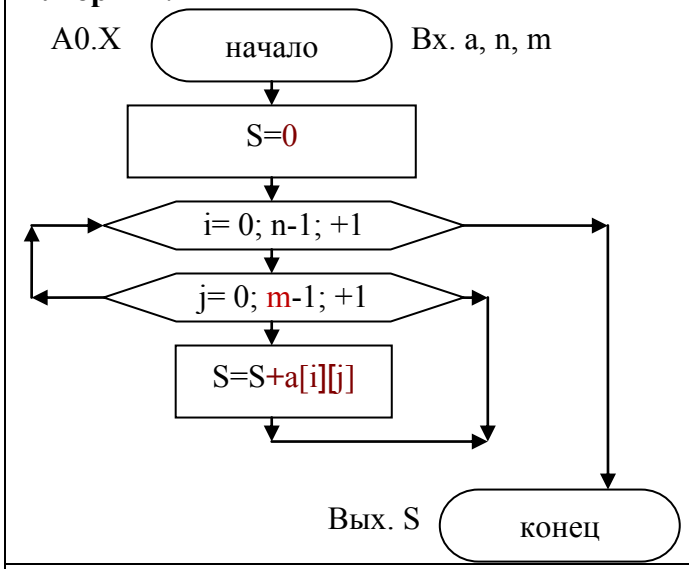
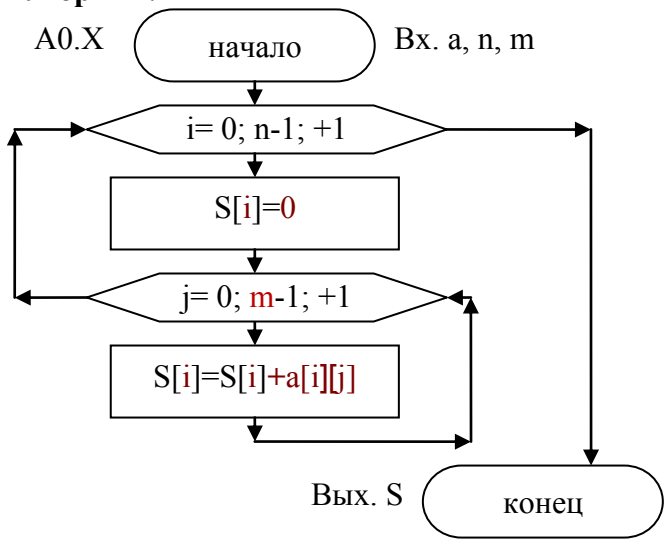
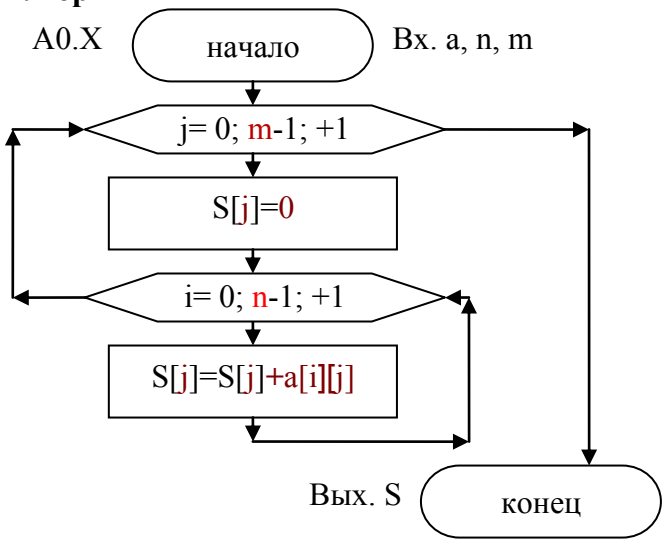
Программный код (фрагмент)

```
Est = false;
for(i=0; i<n; i++)
  if (a[i]<0 ){
        if( !Est ){ // первое отрицательное, до этого не было
              Amax = a[i];  Kmax = i;  Est = true;
        } //if2
        else   if( a[i] > Amax ){
                    Amax=a[i];  Kmax=i;
              } // if3 else if2
  }// if1 for
```

Поиск элемента по условию

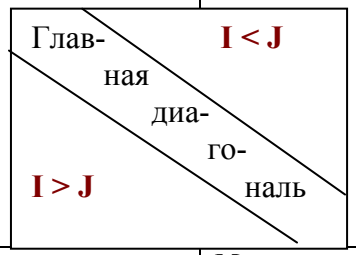
Проверить, ВСЕ ЛИ элементы массива *a*(*n*) отрицательны. Если не все, найти указать номер Nom первого из неотрицательных.	**Проверить**, есть ли в массиве *a*(*n*) **ХОТЯ БЫ ОДИН** отрицательный элемент. Если есть, то найти **номер первого** *Nom* из таких элементов.
Метод. Отделим ввод-вывод от обработки и рассмотрим только обработку. Пусть *i* – индекс текущего элемента массива *a*. Пусть *flag* – логического типа (*bool*) – результат проверки условия: *true* (истина), если все элементы массива отрицательны, и *false* (ложь), если есть хотя бы один неотрицательный элемент в массиве. Примем за начальное значение результата *true* (*flag=true*), но как только встретим хотя бы один неподходящий по условию элемент, изменим наше первоначальное предположение на *false*, запомним его номер и завершим поиск. Если в массиве все элементы отрицательны, наше первоначальное предположение (*flag=true*) останется в силе до конца перебора всех *n* элементов массива.	**Метод**. Отделим ввод-вывод от обработки и рассмотрим только обработку. Пусть *i* – индекс текущего элемента массива *a*. Пусть *Nom* – номер (индекс+1) первого из отрицательных элементов в массиве *a*, или 0, если нет отрицательных. Примем за начальное значение номера число 0 (*Nom*=0), затем, как только найдем подходящий по условию элемент, запомним его номер (*Nom=i*) и завершим поиск. Если в массиве не найдется ни одного отрицательного элемента, значение номера останется 0 и цикл просмотра элементов массива завершится после просмотра всех *n* элементов.
Алгоритм. 	**Алгоритм**. 
Программный код (фрагмент) **flag=true; i=0;** **while (flag && i<n){ //или for(i=0; flag&& i<n;){** **if (a[i]<0) i++;** **else {** **flag=false; Nom=i+1;** **}**	**Программный код (фрагмент)** **Nom=0; i=0;** **while (Nom==0 && i<n) {** **if(a[i]<0) Nom=i+1;** **i++;** **}** **//или цикл for(; !Nom && i<n; i++){**

Для **матриц** каждый из перечисленных алгоритмов применим как для **целой матрицы**, так и для **отдельных ее частей**.

Сумма элементов всей матрицы	**Суммы элементов строк**	**Суммы элементов столбцов**
Найти сумму S (простая переменная) всех элементов матрицы a(n*m)	Найти суммы S(n) (одномерный массив) элементов всех строк матрицы a(n*m) по отдельности.	Найти суммы S(m) (одномерный массив) элементов всех столбцов матрицы a(n*m) по отдельности.
Метод. Отделим ввод-вывод от обработки и рассмотрим только обработку. Пусть i – индекс строки, j – индекс столбца текущего элемента двумерного массива a. Примем за начальное значение суммы число 0 (**S=0**), т.к. S=0+S По очереди с первой по последнюю строки (i=0..n-1): Просматриваем для каждого столбца (j=0..m-1) Добавляем элемент к сумме (**S=S+a[i][j]**) Полученная сумма будет искомой.	**Метод**. Отделим ввод-вывод от обработки и рассмотрим только обработку. Пусть i – индекс строки, j – индекс столбца текущего элемента двумерного массива a. **Для каждой строки** (i=0..n-1): За начальное значение суммы i-ой строки берем 0 (**S[i]=0**), Просматриваем элементы строки (j=0..m-1) Изменяем сумму (**S[i]=S[i]+a[i][j]**) Полученные **n** сумм будут искомыми.	**Метод**. Отделим ввод-вывод от обработки и рассмотрим только обработку. Пусть i – индекс строки, а j – индекс столбца текущего элемента двумерного массива a. **Для каждого столбца** (j=0..m-1): За начальное значение суммы j-ого столбца берем 0 (**S[j]=0**), Просматриваем элементы столбца (i=0..n-1) Изменяем сумму (**S[j]=S[j]+a[i][j]**) Полученные **m** сумм будут искомыми.
Алгоритм. 	**Алгоритм**. 	**Алгоритм** 
Программный код (фрагмент) **S=0;** **for(i=0; i<n; i++)** **for(j=0; j<m; j++)** **S=S+a[i][j];**	**Программный код (фрагмент)** **for(i=0; i<n; i++){** **S[i]=0;** **for(j=0; j<m; j++) S[i]=S[i]+a[i][j];** **}**	**Программный код (фрагмент)** **for(j=0; j<m; j++){** **S[j]=0;** **for(i=0; i<n; i++){** **S[j]=S[j]+a[i][j];** **}} // for2 for1**

Сумма элементов ниже главной диагонали

Найти сумму S (простая переменная) всех элементов квадратной матрицы a(n*n), лежащих ниже главной диагонали



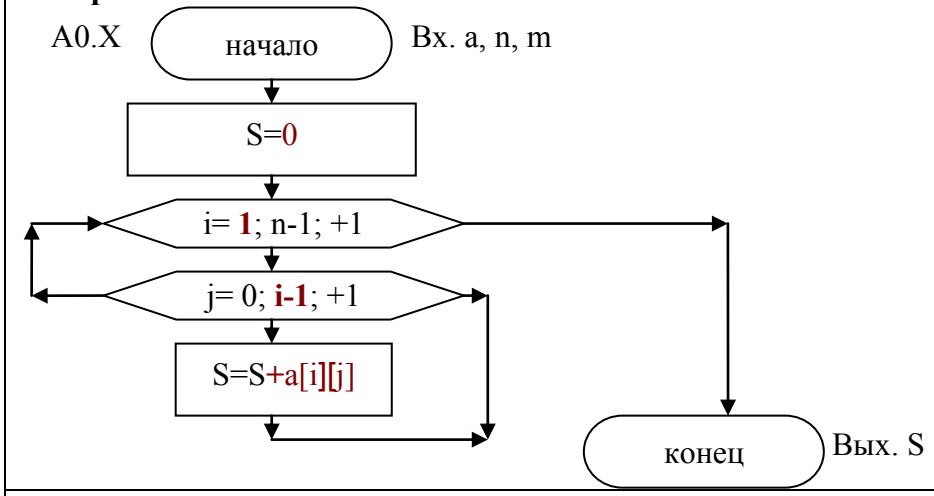
Метод.
Отделим ввод-вывод от обработки и рассмотрим только обработку.

Пусть i – индекс строки, j – индекс столбца текущего элемента двумерного массива a.

Примем за начальное значение суммы число 0 (**S=0**), т.к. S=0+S
Теперь по очереди **со второй** по последнюю строки (i=**1**..n-1):
  Просматриваем с первого по **(i-1)-й** элемент (j=0..(**i-1**))
    Добавляем элемент к сумме (**S=S+a[i][j]**)

Полученная сумма будет искомой.

Алгоритм.



Программный код (фрагмент)

```
S=0;
for(i=1; i<n; i++)
        for( j=0 ;j< i; j++)
                S=S+a[i][j];
```

Сумма элементов выше главной диагонали

Найти сумму S (простая переменная) всех элементов квадратной матрицы a(n*n), лежащих выше главной диагонали

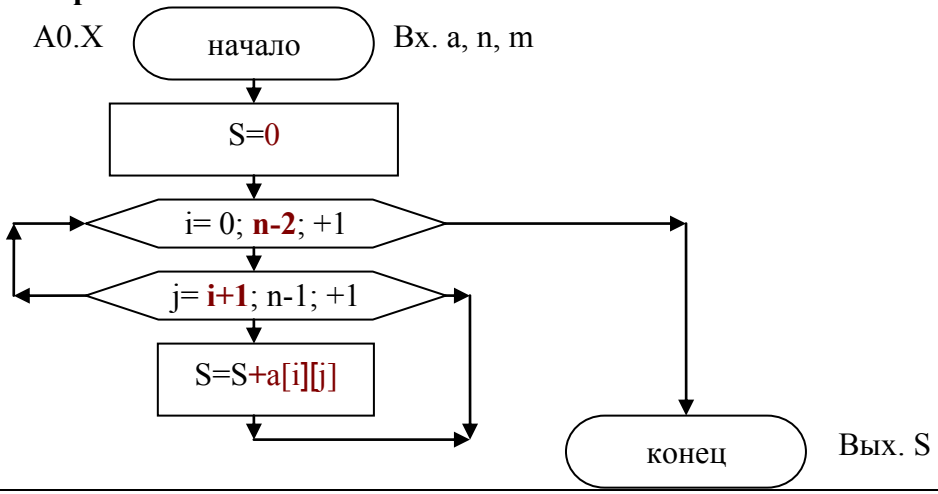
Метод.
Отделим ввод-вывод от обработки и рассмотрим только обработку.

Пусть i – индекс строки, j – индекс столбца текущего элемента двумерного массива a.

Примем за начальное значение суммы число 0 (**S=0**), т.к. S=0+S
Теперь по очереди с первой по **пред**последнюю строки (i=0..(**n-2**)):
  Просматриваем с **(i+1)-го** по последний элемент (j=(**i+1**)..n-1)
    Добавляем элемент к сумме (**S=S+a[i][j]**)

Полученная сумма будет искомой.

Алгоритм.



Программный код (фрагмент)

```
S=0;
for(i=0; i<n-1; i++)
        for (j=i+1; j<n; j++ )
                S=S+a[i][j];
```