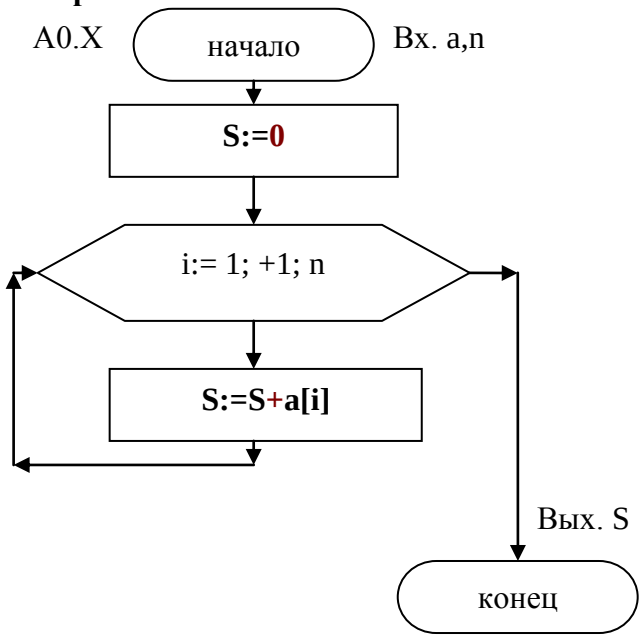
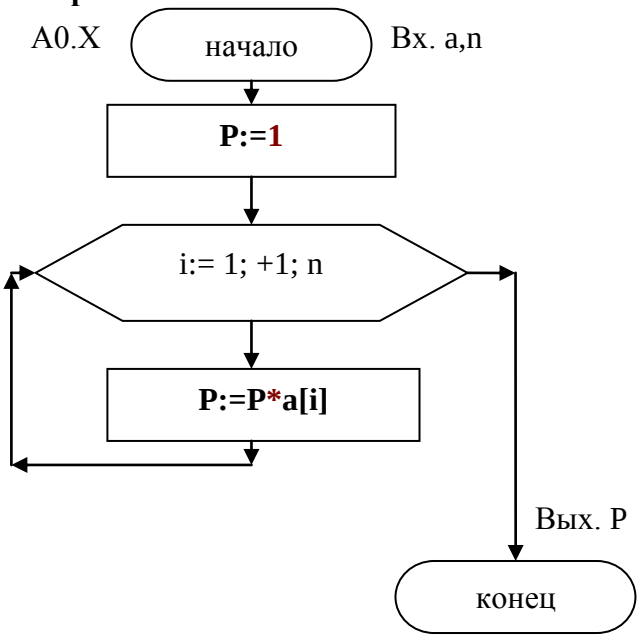
<table>
<tr>
<td><u>Сумма</u>.
Найти сумму S элементов массива a(n)</td>
<td><u>Произведение</u>.
Найти произведение P элементов массива a(n)</td>
</tr>
<tr>
<td>Метод.
Отделим ввод-вывод от обработки и рассмотрим только обработку.

Пусть i – номер текущего элемента массива a
$s=\sum_{i=1}^{n}a[i]=0+\sum_{i=1}^{n}a[i]=$
$=(...((0+a[1])+a[2])+...)+a[n]$
Примем за начальное значение суммы число 0 (S:=0), т.к. S=0+S
Теперь по очереди с первого по последний элемент (i=1..n):
Просматриваем элементы
И добавляем их к сумме (S:=S+a[i])

Полученная сумма будет искомой.</td>
<td>Метод.
Отделим ввод-вывод от обработки и рассмотрим только обработку.

Пусть i – номер текущего элемента массива a
$p=\prod_{i=1}^{n}a[i]=1*\prod_{i=1}^{n}a[i]=$
$=(...(1*a[1])*a[2])*...)*a[n]$
Примем за начальное значение произведения число 1 (P:=1), т.к. P=1*P
Теперь по очереди с первого по последний элемент (i=1..n):
Просматриваем элементы
И изменяем произведение (P:=P*a[i])

Полученное произведение будет искомым.</td>
</tr>
<tr>
<td>Алгоритм.
 
</td>
<td>Алгоритм.
 
</td>
</tr>
<tr>
<td>Программный код (фрагмент)

S:=0;
For i:=1 to n do
S:=S+a[i];</td>
<td>Программный код (фрагмент)

P:=1;
For i:=1 to n do
P:=P*a[i];</td>
</tr>
</table>

Сумма с условием

Найти сумму S отрицательных элементов массива a(n)

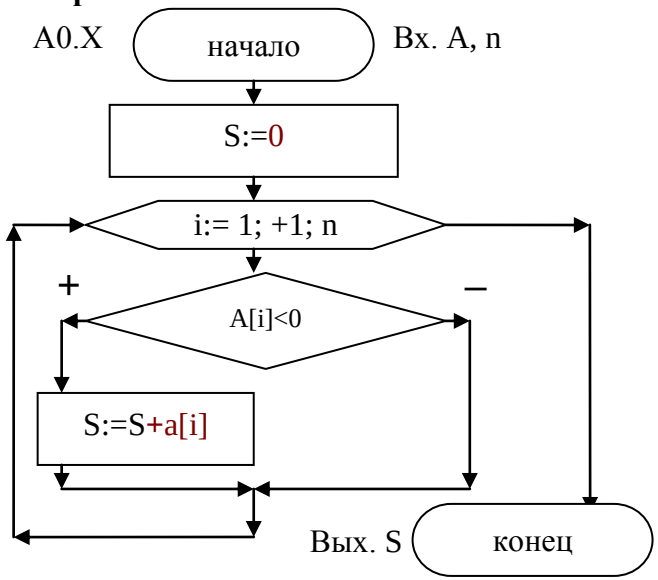
Метод. Аналогичен выше рассмотренному.

Отделим ввод-вывод от обработки и рассмотрим только обработку.

Пусть i – номер 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=1..n):
- Просматриваем элементы и
- Если отрицателен (a[i]<0), то
  - добавляем его к сумме (S:=S+a[i])

Полученная сумма будет искомой.

Алгоритм.



Программный код (фрагмент)

```
S:=0;
For i:=1 to n do
        If a[i]<0 then
                S:=S+a[i];
```

Произведение с условием

Найти произведение P отрицательных элементов массива a(n)

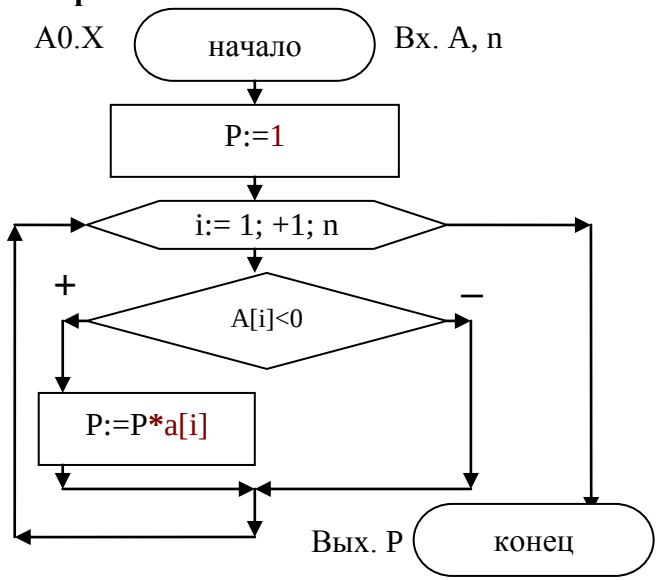
Метод. Аналогичен выше рассмотренному.

Отделим ввод-вывод от обработки и рассмотрим только обработку.

Пусть i – номер 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=1..n):
- Просматриваем элементы и
- Если отрицателен (a[i]<0), то
  - Изменяем произведение (P:=P*a[i])

Полученное произведение искомое.

Алгоритм.



Программный код (фрагмент)

```
P:=1;
For i:=1 to n do
        If a[i]<0 then
                P:=P*a[i];
```

Количество с условием

Найти количество Kol отрицательных элементов массива a(n)

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

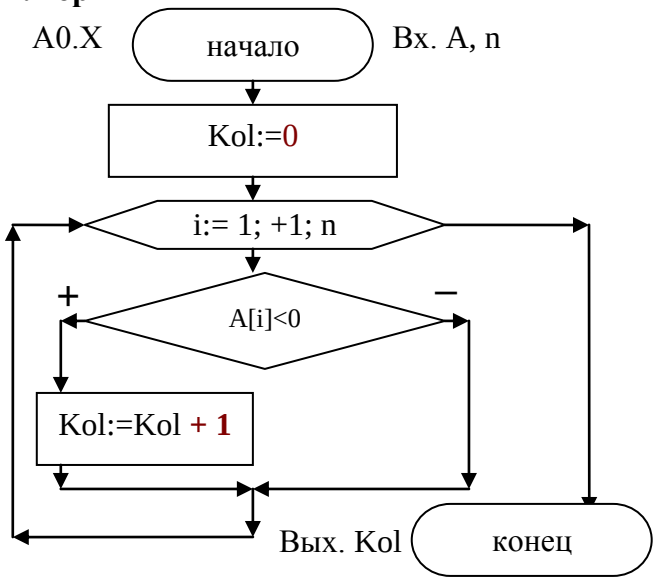
Пусть i – номер текущего элемента массива a.

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=1..n):
- Просматриваем элементы и
- Если отрицателен (a[i]<0), то
  - Изменяем кол-во (Kol:=Kol + 1)
  - Еще один нашли!

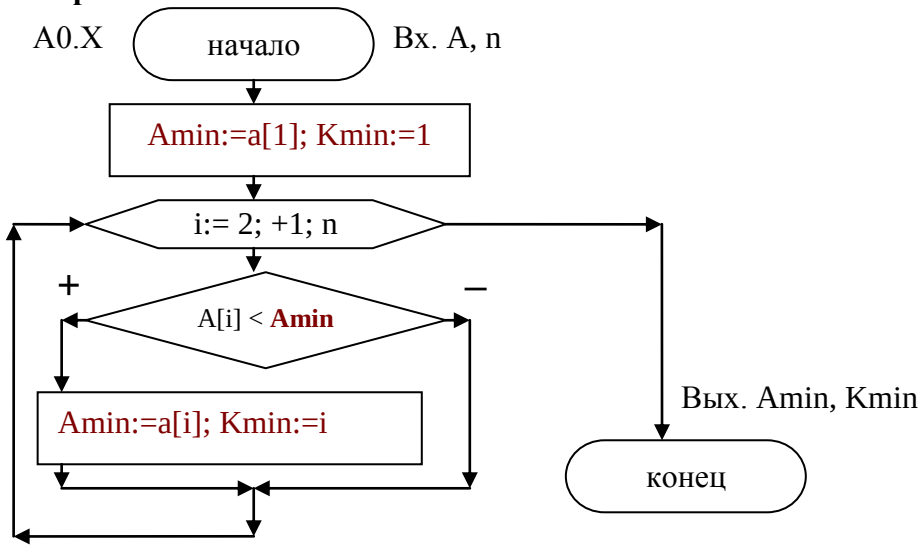
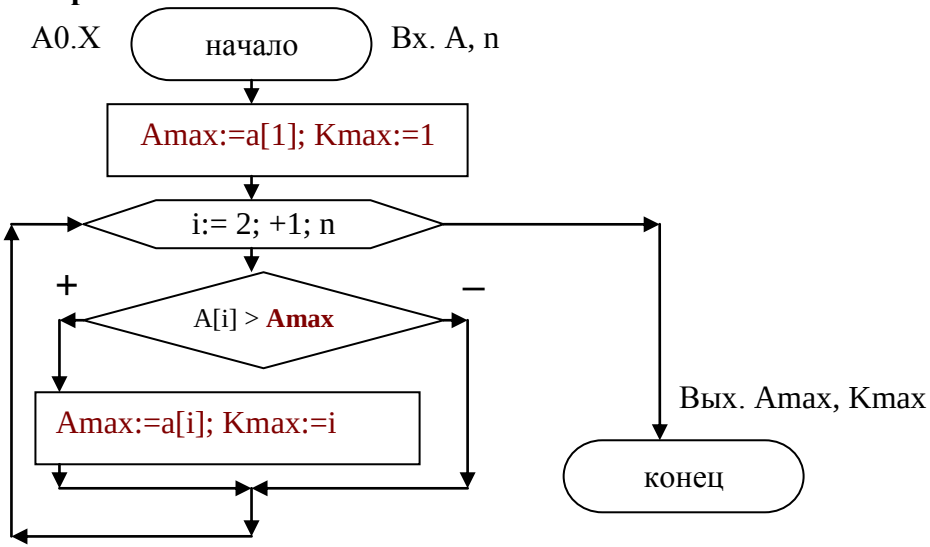
Полученное кол-во искомое.

Алгоритм



Программный код (фрагмент)

```
Kol:=0;
For i:=1 to n do
        If a[i]<0 then
                Kol:=Kol+1;
```

Минимум Найти значение Amin и номер Kmin первого из минимальных элементов массива a(n)	**Максимум** Найти значение AMax и номер KMax первого из максимальных элементов массива a(n)
Метод. Отделим ввод-вывод от обработки и рассмотрим только обработку. Пусть i – номер текущего элемента массива a. Примем за начальное значение минимума первый элемент (Amin:=a[1]; Kmin:=1), может меньшего и не найдем. Теперь по очереди со второго по последний элемент (i=2...n): Просматриваем элементы и Если он меньше текущего минимума (a[i]<Amin), то Изменяем минимум (Amin:=a[i]; Kmin:=i) Новый минимум! Полученный минимум – искомый.	**Метод.** Отделим ввод-вывод от обработки и рассмотрим только обработку. Пусть i – номер текущего элемента массива a. Примем за начальное значение максимума первый элемент (Amax:=a[1]; Kmax:=1), может большего и не найдем. Теперь по очереди со второго по последний элемент (i=2...n): Просматриваем элементы и Если он больше текущего максимума (a[i]>Amax), то Изменяем максимум (Amax:=a[i]; Kmax:=i) Новый максимум! Полученный максимум – искомый.
Алгоритм. 	**Алгоритм.** 

Программный код (фрагмент)
Amin:=a[1]; Kmin:=1;
For i:=2 to n do
If a[i] < Amin then
begin
Amin:=a[i]; Kmin:=i;
end;
	Программный код (фрагмент)
	Amax:=a[1]; Kmax:=1;
	For i:=2 to n do
	If a[i] > Amax then
	begin
	Amax:=a[i]; Kmax:=i;
	end;

Поиск максимума с условием

Найти максимальный элемент (номер (*KMax*) 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.

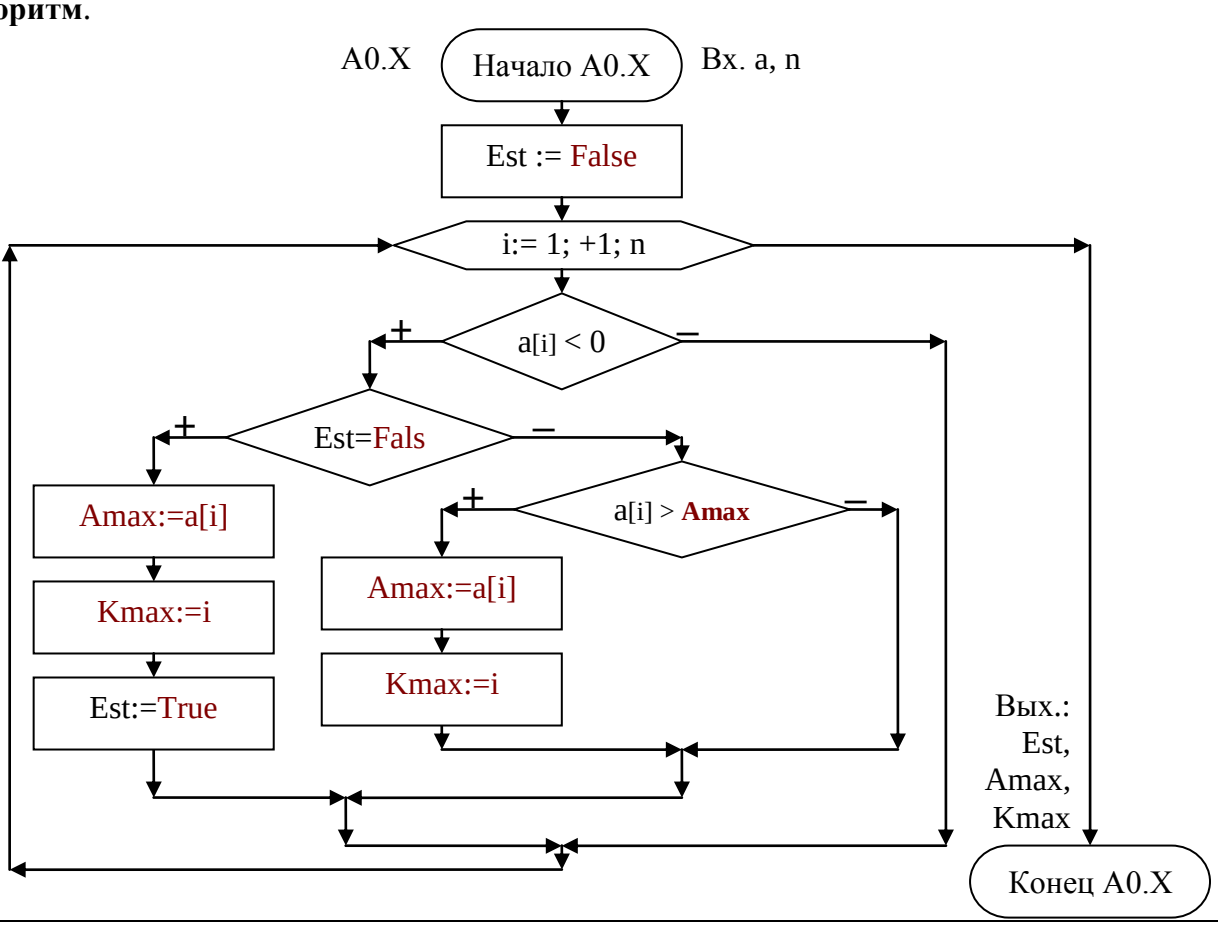
Метод.

Отделим ввод-вывод от обработки и рассмотрим только обработку.
Пусть i – номер 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=1..n):

- 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.

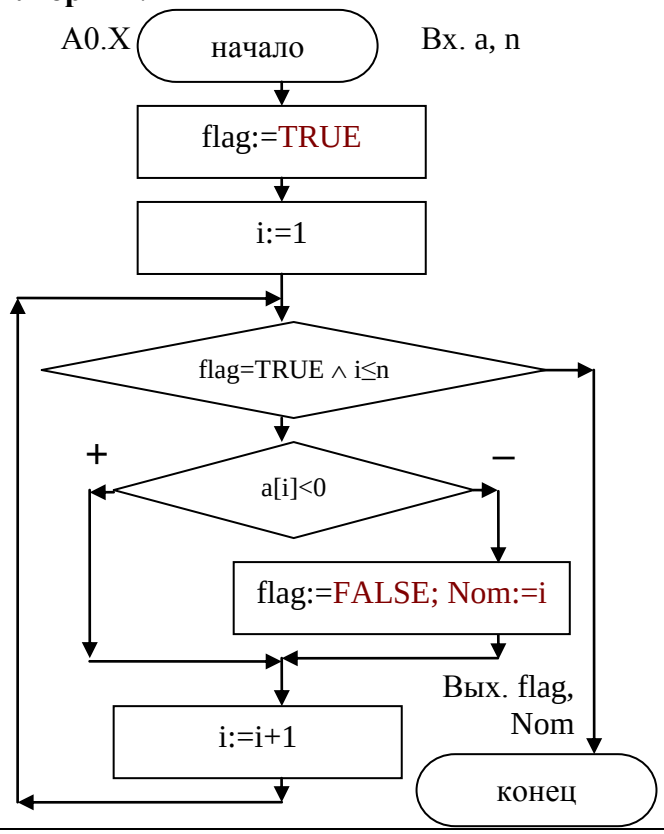
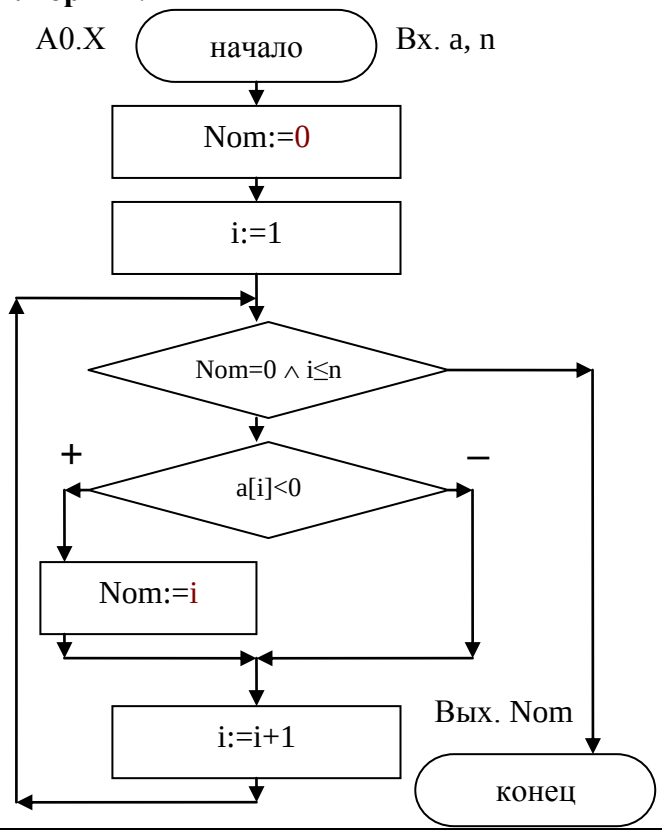
Алгоритм.



Программный код (фрагмент)

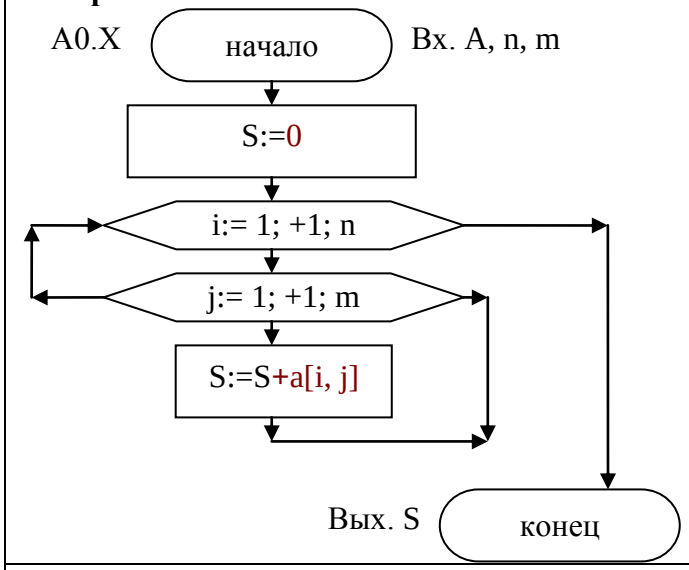
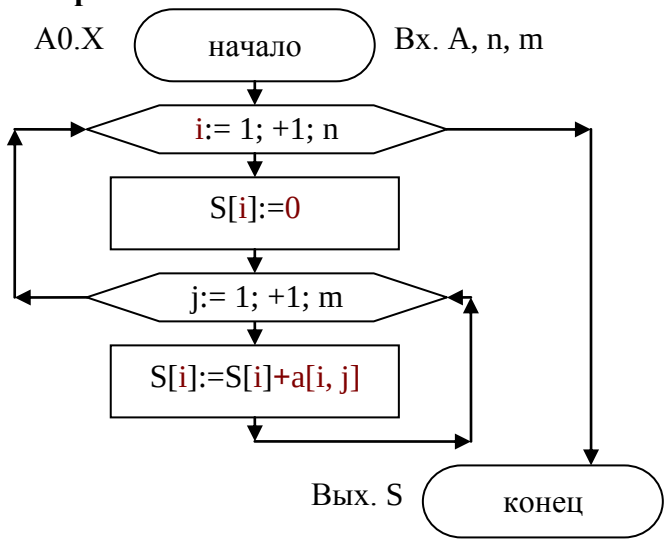
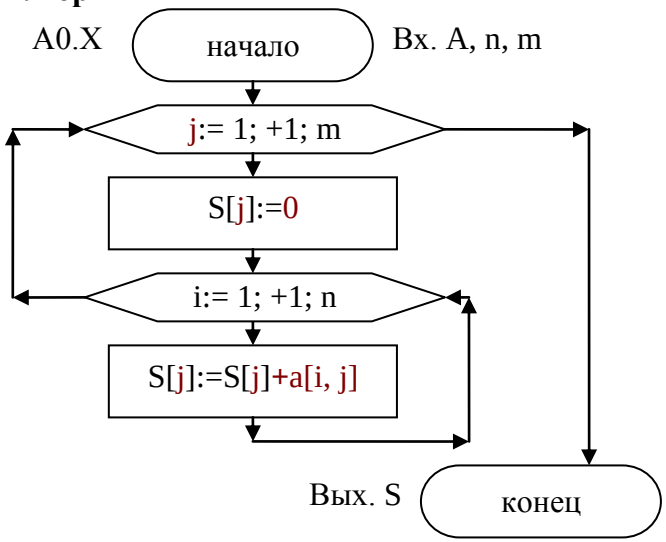
```
Est := False;
For i:=1 to n do
  If a[i]<0 then
        If Est = False then // первое отрицательное
        Begin
              Amax := a[i];  Kmax := i;  Est := True;
        End
        Else   If a[i] > Amax then
              Begin
                    Amax:=a[i];  Kmax:=i;
              End;
```

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* – номер текущего элемента массива *a*. Пусть *flag* – логического типа (boolean) – 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* – номер текущего элемента массива *a*. Пусть *Nom* – номер первого из отрицательных элементов в массиве *a*. Примем за начальное значение номера число 0 (*Nom*:=0), и 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 нулевым и цикл просмотра элементов массива завершится после просмотра всех *n* элементов.
Алгоритм.  	**Алгоритм.** 

Программный код (фрагмент)
flag:=True; i:=1;
while (flag=True) and (i<=n) do
begin if a[i]<0 then
else begin flag:=False; Nom:=i end;
inc(i);
end;
	Программный код (фрагмент)
	Nom:=0; i:=1;
	while (Nom=0) and (i<=n) do
	begin
	if a[i]<0 then Nom:=i;
	inc(i);
	end;

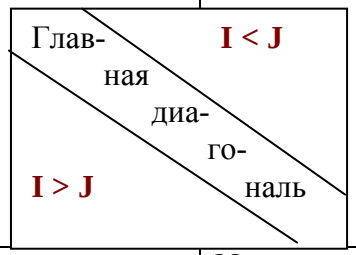
Для **матриц** каждый из перечисленных алгоритмов применим как для **целой матрицы**, так и для **отдельных ее частей.**

Сумма элементов всей матрицы	**Суммы элементов строк**	**Суммы элементов столбцов**
Найти сумму S (простая переменная) всех элементов матрицы a(n*m)	Найти суммы S (одномерный массив) элементов всех строк матрицы a(n*m) по отдельности.	Найти суммы S (одномерный массив) элементов всех столбцов матрицы a(n*m) по отдельности.
Метод. Отделим ввод-вывод от обработки и рассмотрим только обработку. Пусть i – номер строки, а j – номер 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 последнюю строки (i=1..n): Просматриваем с первого по последний элемент (j=1..m) 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 – номер строки, а j – номер столбца текущего элемента двумерного массива a. **Для каждой строки** (i=1..n): За начальное значение суммы i-ой строки берем 0 (**S[i]:=0**), Просматриваем элементы строки (j=1..m) 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 – номер строки, а j – номер столбца текущего элемента двумерного массива a. **Для каждого столбца** (j=1..m): За начальное значение суммы j-ого столбца берем 0 (**S[j]:=0**), Просматриваем элементы столбца (i=1..n) Изменяем сумму (**S[j]:=S[j]+a[i, j]**) Полученные **m** сумм будут искомыми.
Алгоритм. 	**Алгоритм**. 	**Алгоритм** 

Программный код (фрагмент)
S:=0;
For i:=1 to n do
For j:=1 to m do
S:=S+a[i, j];
	Программный код (фрагмент)
	For i:=1 to n do
	Begin S[i]:=0;
	For j:=1 to m do
	S[i]:=S[i]+a[i, j];
	End;
		Программный код (фрагмент)
		For j:=1 to m do
		Begin S[j]:=0;
		For i:=1 to n do
		S[j]:=S[j]+a[i, j];
		End;

Сумма элементов ниже главной диагонали

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



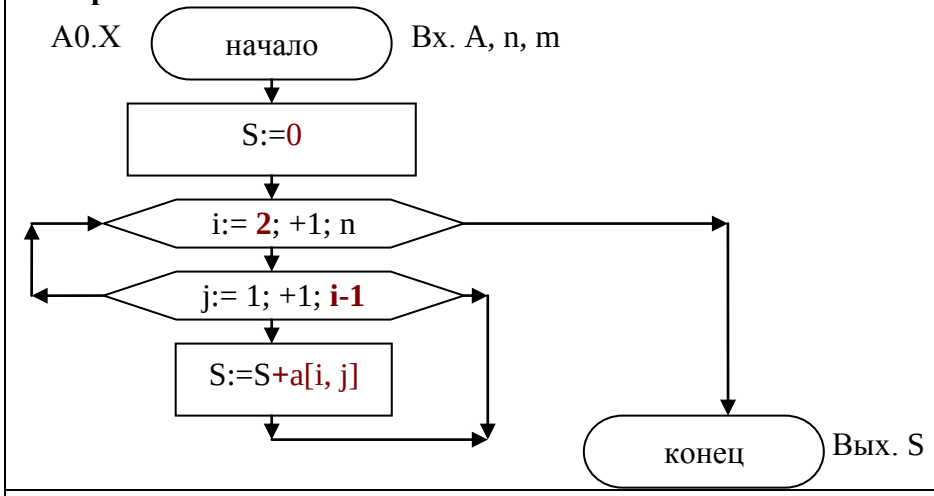
Метод.
Отделим ввод-вывод от обработки и рассмотрим только обработку.

Пусть i – номер строки, а j – номер 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=**2**..n):
  Просматриваем с первого по **(i-1)-й** элемент (j=1..(**i-1**))
    Добавляем элемент к сумме (**S:=S+a[i, j]**)

Полученная сумма будет искомой.

Алгоритм.



Программный код (фрагмент)

```
S:=0;
For i:=2 to n do
      For j:=1 to i-1 do
            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

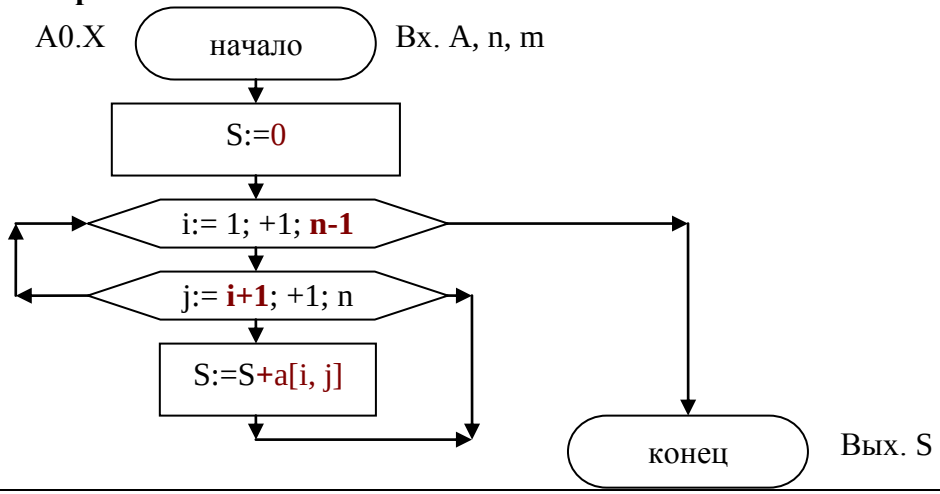
Метод.
Отделим ввод-вывод от обработки и рассмотрим только обработку.

Пусть i – номер строки, а j – номер 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=1..(**n-1**)):
  Просматриваем с **(i+1)-го** по последний элемент (j=(**i+1**)..n)
    Добавляем элемент к сумме (**S:=S+a[i, j]**)

Полученная сумма будет искомой.

Алгоритм.



Программный код (фрагмент)

```
S:=0;
For i:=1 to n-1 do
      For j:=i+1 to n do
            S:=S+a[i, j];
```