Пример спецификации к Лабораторным работам №2-4 (Вариант 0)
Абстракция А0

1. Постановка задачи (ПЗ).

Задание: Написать программу в виде консольного приложения для одномерного массива(ов) в соответствии с условием.

(полностью задание и требования в Лабораторной работе №2)

Условие: Координатами *x,y* заданы *n* точек плоскости. Найти процент точек, удаление которых от начала координат больше заданной величины *r* и притом обе координаты положительны, а также подсчитать среднее удаление всех точек от начала координат.

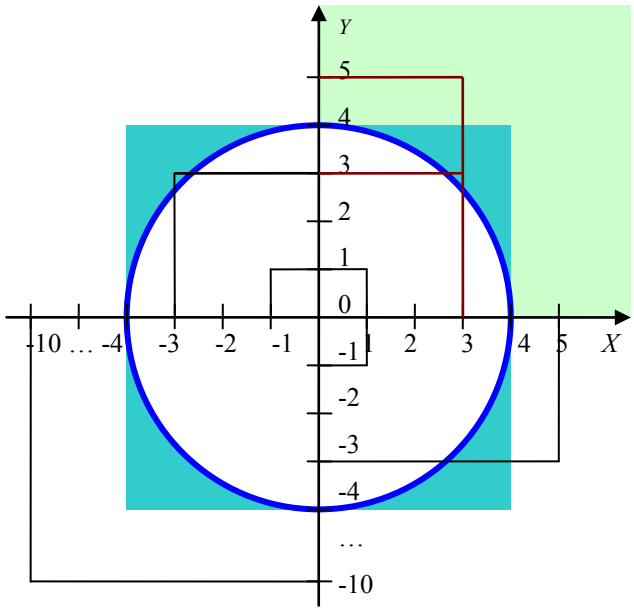
2. Уточненная постановка задачи (*добавить недостающие имена, определить типы и структуры, добавить альтернативные (если есть – «отрицательные», если нужен особый ответ – «положительные») решения).*

Даны два ***одномерных вещественных массива*** *x* и *y* из *n* ($0<n\le20$) элементов – координаты точек плоскости, и ***целая*** величина *r*.

Найти:

p **–** процент точек, удаление которых от начала координат больше заданной величины *r* и притом обе координаты положительны; (*Пример особого «положительного» ответа: Если p=100%, то вывести сообщение «ВСЕ 100% точек лежат в заданной области»*)

sr **–** среднее удаление всех точек от начала координат. (В данном условии нет отрицательного ответа, т.к. всегда можно найти значение sr при $0<n\le20$. *Пример «отрицательного» ответа: Если бы среднее удаление нужно было найти не среди ВСЕХ точек, а только среди НЕпопавших в заданную для поиска p область, то следовало бы добавить следующее предложение: Если нет точек, НЕпопавших в заданную область, то вывести сообщение «Невозможно вычислить среднее удаление SR»*)



3. Пример с иллюстрацией (станет одним из тестов): *(графическая иллюстрация в примере не всегда возможна, но всегда желательна)*

Возьмем ***r*** = 4, ***n*** = 10. И точки из разных четвертей и областей (вне круга, в описанном квадрате, вне квадрата, на границах области, их пересечении):

x	1	1	-1	3	-3	0	3	5	5	-10
y	1	-1	1	3	3	4	5	-3	0	-10

Есть *две* точки с положительными координатами вне круга с радиусом *r* = 4. Это точки (3,3) и (3,5). Значит, искомый процент ***p*** = 2/10*100% = 20%

Среднее удаление *всех* точек от начала координат ***sr*** = $\frac{3\sqrt{1^2+1^2}+2\sqrt{9^2+9^2}+4+2\sqrt{25^2+9^2}+5+\sqrt{2*100^2}}{10}\approx 4,753196$

4. Таблица данных

(начало таблицы данных)

Класс	**Имя**	**Описание (смысл, диапазон, точность)**	**Тип**	**Структура**	**Формат в/в**
Входные данные	***n***	число точек плоскости, $0 < n \le 20$	цел	простая переменная	XX (:2)
	x	абсциссы точек плоскости, $\|x_i\|<=10$, точн. 0.1	вещ	одномерный массив (20)	+XX.X (:5:1)
	y	ординаты точек плоскости, $\|y_i\|<=10$, точн. 0.1	вещ	одномерный массив (20)	+XX.X (:5:1)
	r	критическое удаление, $1 \le r \le 14$ т.к. $0 < r < \sqrt{10^2+10^2} \approx 14,14$	цел	простая переменная	XX (:2)

(продолжение таблицы данных)

Класс	Имя	Описание (смысл, диапазон, точность)	Тип	Структура	Формат в/в
Выходные данные	***p***	искомый процент точек, $0 \le p \le 100$, точн. 0.1 (см.тесты 4,5)	вещ	простая переменная	XXX.X (:5:1)
	sr	среднее удаление всех точек от начала координат, $0 \le sr \le \sqrt{10^2 + 10^2} \le 10\sqrt{2} < 15$; точн. 0.00001 (см. тесты 6 и 7)	вещ	простая переменная	XX.XXXXX (:8:5)
Промежу-точные данные*	***i***	индекс текущего элемента, $-1 \le i \le 20$ *(Индексы с 0 до 19 и выход при -1 или 20 при переборе с конца/ с начала)*	цел	простая переменная	---
					---

**Промежуточные данные заполняются не сразу, а по мере необходимости использовать дополнительные переменные. Оставляйте место для них*

5. Входная форма:

```
  1.1   Количество точек n:
обр1    _____
  1.2   <n>
        _____
  2.1   Критическое удаление r:
обр2    _____
  2.2   <r>
        _____
обр3.1  Абсциссы и ординаты:
        _____
        <x[0]> <y[0]>
Обр3.2  <x[1]> <y[1]>
          ......
        <x[n-1]> <y[n-1]>
```

Образцы 1.1, 2.1 и 3 - приглашения для ввода n, r и координат точек в режиме диалога.
В лабораторной работе 4 *(ввод из собственного файла) их можно не записывать в файл вовсе:*

```
  1.2   <n>
  2.2   <r>
        <x[0]> <y[0]>
Обр3.2  <x[1]> <y[1]>
          ......
        <x[n-1]> <y[n-1]>
```

6. **Выходная форма:**

35 пробелов перед текстом (это примерно центр экрана при его ширине 80). Также полезно учитывать высоту экрана – 25 строк

```
Обр4                Лаб.2
        _____
Обр5    Количество точек n = <n>
обр6    Критическое удаление r = <r>
        _____
Обр7    Абсциссы и ординаты:
        _____
          <x[0]>      <y[0]>
обр8      <x[1]>      <y[1]>
                ......
          <x[n-1]>      <y[n-1]>
        _____
обр9    Процент точек p = <p>
обр10   Среднее удаление sr = <sr>
        _____
Обр11   Сюда нужно добавить
        альтернативные положительные и
        отрицательные ответы, чтобы
        пронумеровать образцы ответов.
```

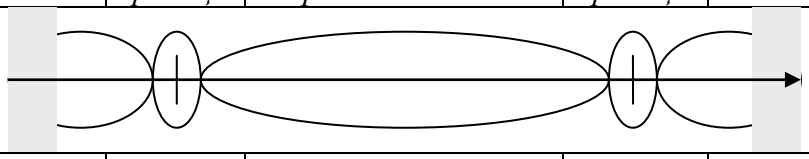
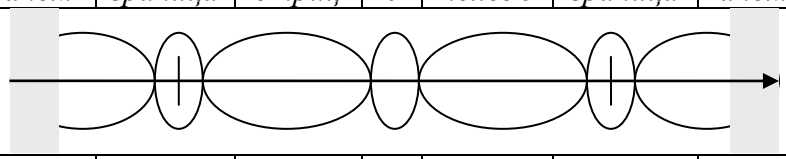
7. Аномалии

Пока не рассматриваем, но можете сделать (n<1, n>20, r<1, r>14, $\exists i$:|x[i]|>10, $\exists i$:|y[i]>10),например:

№	Описание	Условие возникновения**	Реакция на аномалию
1	***n*** меньше минимально допустимого значения	n<1	Сообщение: «Некорректное n: n<1» (обр.А1) Действие: Завершение работы программы
2	…		

**В условиях желательно НЕ использовать связку ∨ (ИЛИ), объединяя разные случаи воедино, и выводить *разные* сообщения, чтобы реакция была более конкретная.

8. Функциональные тесты

Исходные данные								**Результаты**		**Тест №**
	аном	*граница*	*Средние значения*			*граница*	*аном*	***p***	*макс* =100	2
									*мин* =/=0: 5%, 5.**2**6% (для **вещ**.чисел: два-три минимума по **абсолют**. значению, НЕ равных нулю, в каком знаке отличие)	4,5
n	<1	1	[2, 19]			20	>20			
Тест№	---	3	1			2	---		*сред* = (0,100)	1
	аном	*граница*	*Средние значения*			*граница*	*аном*		*0*	3
r	<1	1	[2, 13]			14	>14		*не сущ* = не возможно	---
Тест№	---	3	1			2	---		*Макс.выч.нагрузка* = 20 точек из 20 попали	2
	аном	*граница*	*отриц*	0	*полож*	*граница*	*аном*			
								***sr***	*макс* = $10\sqrt{2}$	2
									мин =/= 0: 0.005, 0.005**26** → точн.10^{-5} ((для **вещ**.чисел: два-три минимума по **абсолют**. значению, НЕ равных нулю, в каком знаке отличие))	6,7
x[i]	<-10	-10	(-10,0)	0	(0,10)	10	>10			
Тест№	---	1	1	1	1	2	---		*сред* = (0, $10\sqrt{2}$)	1
	аном	*граница*	*сред*	*0*	*сред*	*граница*	*аном*		*0*	3
y[i]	<-10	-10	(-10,0)	0	(0,10)	10	>10		*не сущ* = не возможно	---
Тест№	---	1	1	1	1	2	---		*Макс.выч.нагрузка* = 20 точек по (±10,±10)	2

*При создании тестов можно выделить больше **классов эквивалентности** для выходных и исходных данных, например:*

А) для вещественных значений: если при совпадении минимума или максимума с 0, искать минимум по абсолютной величине, отличный от 0. Это поможет проверить точность выходных данных.Например, у нас минимумы p и sr совпали с 0. Выбираем новые минимумы, лучше сразу два-три, ближайших по модулю к нулю – см.тесты 4,5 и 6,7.

*Б) Даже если минимум/максимум не совпадает с нулем, можно в любом случае вместо **максимума** делать тесты на граничные значения результата (максимальное значение и максимальное по абсолютной величине отрицательное значение), а в качестве **минимума** – отрицательное и/или положительное значения, по абсолютной величине ближайшие к 0;*

*В) Для исходных данных при проверке аномальных ситуаций можно рекомендовать для проверки брать значение, ближайшее к граничному – **пограничное**, чтобы четче проверять границу аномального и нормального значения переменной;*

*Г) При проверке аномальных ситуаций, если 0 (ноль) входит в область аномальной ситуации, **проверять отдельно нули** и НЕнули. Например, для (n<1) отдельно тесты для 0 и -1;*

*Д) Можно расширить проверку исходных данных на аномалии, когда введены **не цифры, а буквы** или «лишние» символы присутствуют во веденном значении. Или, как частный случай, вместо точки в*

*качестве разделителя дробной части использована **запятая**, или наоборот. В лабораторной работе 9 будут проверки ошибок **при открытии и чтении из файла**. Ошибки при чтении как раз могут быть из-за некорректных символов и попытки ввода букв вместо цифр.*

ВАЖНО! Ожидаемые результаты необходимо считать САМИМ вручную (на калькуляторе или другим способом, например, в *Excel*), чтобы затем **сравнивать** с тем, что выдаст ваша программа. А не записывать, что она (программа) выдала!

№ теста	Входные данные	Ожидаемый результат	Смысл теста
1	*n*=10, *r* =4 x: 1, 1, -1, **3**, -3, 0, **3**, 5, 5, -10 y: 1, -1, 1, **3**, 3, 4, **5**, -3, 0, -10	*p* =20 (обр9) *sr* =4,753196 (обр10)	Данные взяты из примера. **Разнообразное** расположение точек по четвертям и областям
2	*n* =20; *r* =14 x: **10**, **10**, **…**, **10** y: **10**, **10**, **…**, **10**	*p* =100 (обр9) *sr* =$10\sqrt{2}$ = =14,14214 (обр10)	**Максимальное** значение всех исходных данных и результатов. **Максимальная вычислительная нагрузка** для p и sr
3	*n*=1; *r* =1 x: 0 y: 0	*p* =0 (обр9) *sr* =0 (обр10)	**Минимальные** значения n,r, p, sr. **Нулевые** значения x,y,p,sr
4,5	*n* =20 r=5 x: 0, 5, 3, 4, **5**, 0, …, 0 y: 5, 0, 4, 3, **5**, 0, …, 0 *n* =19 r=5 x: **5**, 0, …, 0 y: **5**, 0, …, 0	*p=5,0%* *sr=(5+5+5+5+5*$\sqrt{2}$ *)/20* = 27,07107 *p=5,3%* *sr=5*$\sqrt{2}$ */19=0,37216*	***Тесты на точность:*** ***мин* |p| и не 0: 5%, 5.26%** 1/20*100% → 5% 20 кратно 10, поэтому ответ кажется целым, один необходимый знак в дробной части «спрятался», возьмем 19: 1/19*100% → 5,26% или 5,3% без дробной части было бы тоже 5%, а 1/20 не равна 1/19! **Заодно проверим попадание на границу области: $3^2+4^2=5^2$**
6,7	*n* =20 r=1 x: 0, 0, …, 0, 0 y: 0, 0, …, 0, 0,1 *Одну ординату сдвинем на 0,1 в конце* *n* =19 r=1 x: 0,1, 0, …, 0 y: 0, 0, …, 0 *Одну абсциссу сдвигаем на 0,1 в начале Для разнообразия,чтобы увеличить вероятность выявления ошибки работы с первым(0) и последним(19) элементом массива.*	p=0% sr = (0,1 +19*0)/20 = 0,005 - снова проблема с делением на круглое число 20 и с корнем из квадрата 0,1 – пара знаков дробной части могла не проявить себя, возьмем 19: sr = (0,1 + 18*0)/19 = 0,00526	***Тесты на точность:*** ***Мин* |*sr*| и не 0: 0,005 и 0,00526** для различия между минимальными значениями при выводе нужно как минимум 4-5 знаков после запятой – указываем точность в ТД до 5 знака: 0,00001 p=0% в обоих тестах.
8	**Далее тесты на аномальные ситуации, *если их рассматриваете***. Как минимум по одному тесту на каждую аномалию и область эквивалентности, и отдельно на 0, например: n=0 (особая точка и приграничное значение) n=-1 (одна из остальных точек n<1) n=21 (приграничное значение) n=w (не число, ошибка ввода)	Сообщение по обр.А1	Аномалия №1 (некорректное n) … Если в программе нет проверки аномалий, то тестировать отсутствующую проверку бессмысленно

9. Метод

Отделим ввод-вывод от обработки данных (собственно решения задачи).

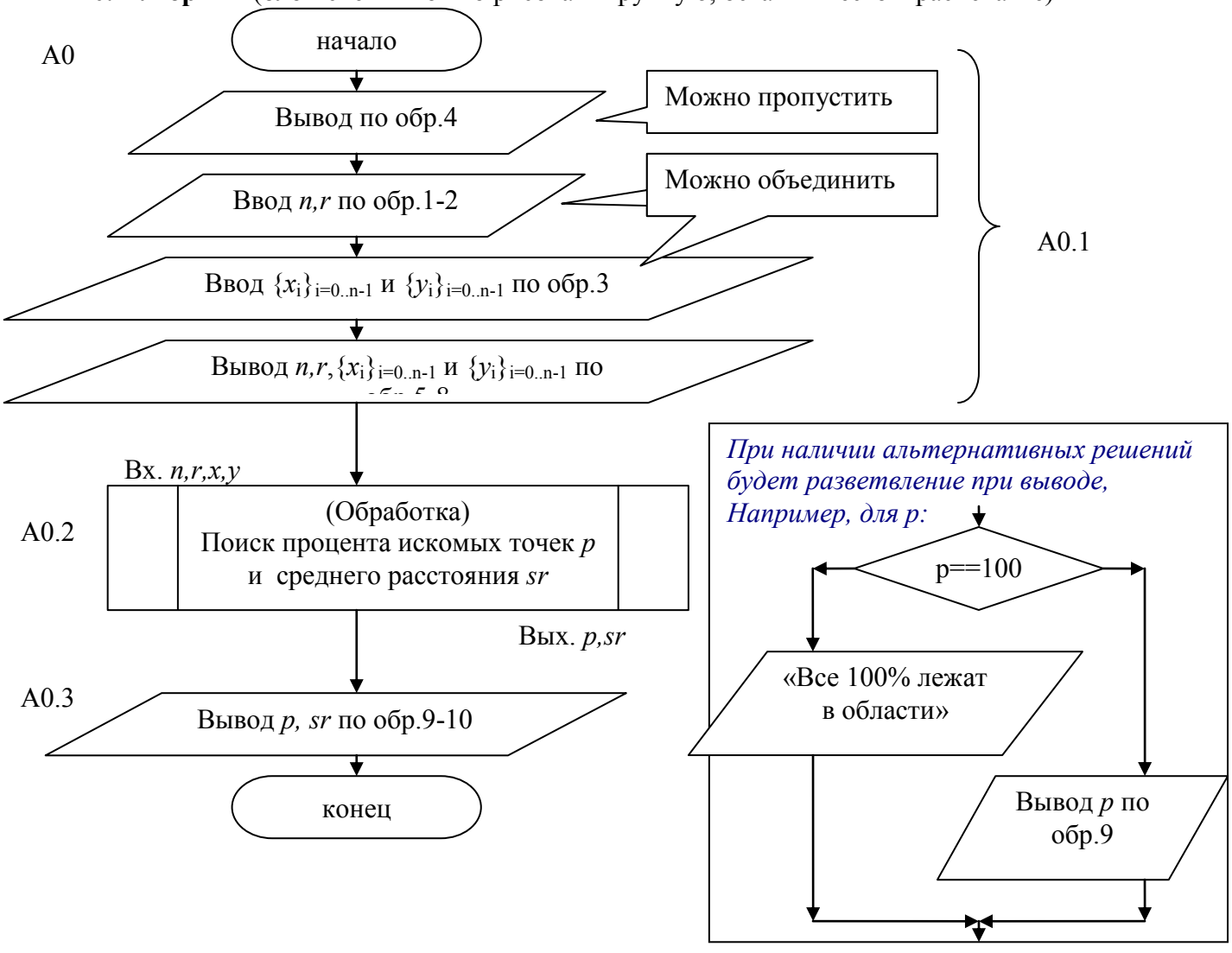
То есть, разделим нашу задачу на три подзадачи:

1. Подзадача А0.1. Ввести исходные данные (обр.1-3) и вывести их на экран (обр.5-8) для визуального подтверждения.
2. Подзадача А0.2. Решение задачи: на основе введенных исходных данных (*n,r,x,y*) найти процент подходящих точек *p* и среднее расстояние *sr*.
3. Подзадача А0.3. Ввести вывести полученные результаты (*p, sr*) на экран (обр.9-10)

Здесь ввод-вывод тривиален и представляет собой чисто техническую задачу; в алгоритмизации нуждается только обработка. Поэтому на нулевом уровне сразу опишем ввод-вывод (подзадачи А0.1 и А0.3), оставив задачу обработки А0.2 в виде абстракции (которую раскроем на следующем уровне проектирования). На данном уровне эту абстракцию в алгоритме заменим ЗАГЛУШКОЙ ***(«Заглушка» – это простейший алгоритм, временно заменяющим реальный алгоритм*** *и помогающий уменьшить область поиска ошибок и отладить уже написанную часть программы, не внося дополнительных ошибок, потенциально содержащихся в том самом реальном алгоритме.* ***Алгоритм «заглушки» должен быть настолько простым****, чтобы в нем не было ошибок или было очень просто их обнаружить и исправить. В простейшем случае, это просто* ***присваивание*** *результата.* ***На следующем шаге разработки «заглушка» заменяется реальным алгоритмом,*** *и, поскольку остальная часть программы уже проверена, то можно с высокой долей вероятности считать, что все вновь обнаруженные* ***ошибки находятся в добавленном алгоритме.)***

Для ввода-вывода массивов потребуется промежуточная переменная – i – цел.,индекс элемента.

10. Алгоритм (блок-схемы можно рисовать вручную, оставив место в распечатке)



11. Программа на *С*. Диалоговый вариант {ввод с клавиатуры, вывод на экран}\
Комментариев в программе можно так много не делать

```
//Лаб.2
#include <stdlib.h> // printf scanf
#include <math.h> // sqrt  M_PI
/*верхняя граница индексов массива – максимальное количество точек*/
#define Nmax 20

int main()
{
   /*описание переменных:*/
   int n,r,i;
   float p, sr;
   float x[Nmax], y[Nmax];

   /*A0.1 - ввод-вывод входных данных*/
   //вывод заголовка в выходной документ:
   printf("%40s\n", "Лаб.2"); // с выравниванием и переходом на след.строку

        //ввод исходных данных:
        printf("Количество точек n:"); scanf("%d",&n); //обр 1.1,1.2
        printf("Критическое удаление r:"); scanf("%d",&r); //обр.2.1,2.2
        //ввод массивов x и y:
        printf("Абсциссы и ординаты:\n"); //обр3.1
        for (i=0; i<n; i++)
                scanf("%f %f", &x[i], &y[i]); //обр3.2:ввод 2 элементов через пробел и переход на след.строку

        //вывод входных данных в выходной документ для подтверждения:
        for(i=1; i<=40; i++) printf(" "); //отделим визуально чертой и строкой введенные и выводимые значения
        printf("\n");

        //вывод n, r по обр5-6
        printf("Количество точек = %d\n", n);
        printf("Критическое удаление = %d\n", r);
        //вывод массивов x и y
        printf("Абсциссы и ординаты: \n"); //обр7
        for( i=0; i<n; i++)
                printf("%10.1f%13.1f\n", x[i], y[i]); //обр8 10=5+5 13=8+5

        /*A0.2 - обработка}*/
   // заглушка - присвоим выходные значения из тестов
        // p=20; sr=4.573196;     //заглушка для теста 1
        p=100; sr=10*sqrt(2);     //заглушка для теста 2
        // p=0; sr=0;       //заглушка для теста 3

        /*A0.3 - вывод результатов*/
        printf("Искомый процент точек = %5.1f\n", p);    //p по обр9
        printf("Среднее удаление = %8.5f\n", sr);          //sr по обр10

   printf("The End");  // getch(); // задержать экран и #include <stdio.h>
   return 0;
}
```

Использование именованной константы ***Nmax*** для границ лучше с точки зрения стиля.

Если константа упомянута несколько раз в коде (допустим есть ещё строка: *if (n>**Nmax**)*,

то изменение значения такой константы позволит изменить значения по всему коду. Без создания именованной константы можно случайно пропустить одно из значений и внести ошибку в программу.

Имя константы произвольно, но должно отражать смысл во избежание путаницы, а значит и ошибок.

Если был бы альтернативный ответ для p=100%:

```
        if (p==100) printf("Все 100% точек попали в область\n");
        else printf("Искомый процент точек = %5.1f\n", p);        //p по обр9
```

Раскрытие абстракции A0.2

*Поскольку условие всей задачи **совпадает** (за исключением необходимости ввода и вывода) с условием выделенной подзадачи, вместо полной спецификации с пунктами 1-11. сделаем просто дополнение к пунктам 4, 9-11.*

9. Метод

Пусть
i – номер текущий точки,
si – удаление точки (x[i], y[i]) от начала координат; тогда

$$si = \sqrt{x[i] * x[i]} + \overline{y[i] * y[i]}$$

Пусть *s* – сумма расстояний всех точек от начала координат; тогда
sr = *s*/*n*; чтобы найти *sr*, надо найти *s*:

$$s = \sum_{i=0}^{n-1} si$$

Пусть *np* – число точек, попавших в область (удаление больше *r*, коорд-ты положительны); тогда

$p = (np/n)*100;$

чтобы найти *p*, надо найти *np*:

$$np = Count_{i=}^{n-}((si > r) \wedge (x[i] > 0) \wedge (y[i] > 0))$$

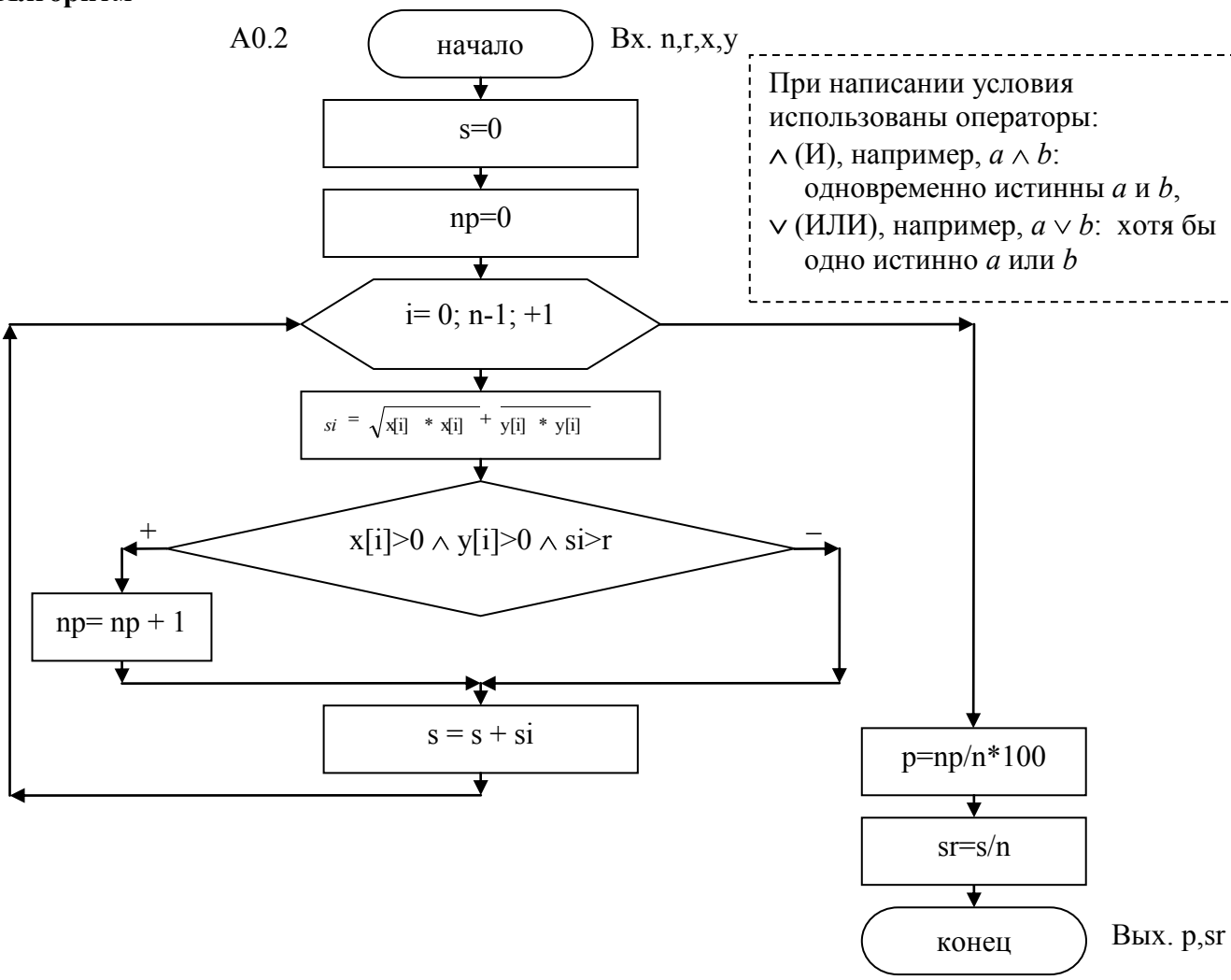
Методы подсчета суммы и количества описаны в файле Base-Algorithmes.doc
Оба алгоритма (поиск суммы и количества) выполним в одном и том же цикле по элементам массивов, иначе расстояние (*si*) каждой точки от центра пришлось бы считать дважды.

4. Таблица данных (продолжение)

(в рукописной спецификации можно дополнить исходную таблицу)

Промежу-точные	*i*	индекс текущего элемента, $-1 \le i \le 20$	цел	простая переменная	---
	np	число точек, удовлетворяющих условию, $0 \le np \le 20$	цел	простая переменная	---
	s	сумма расстояний всех точек от начала координат, $0 \le s \le 20\sqrt{10^2 + 10^2}$ <=20*10* $\sqrt{2}$ <300; точн. 0.001	вещ	простая переменная	---
	si	расстояние до i-ой точки от начала координат, $0 \le si \le \sqrt{10^2 + 10^2} = 10\sqrt{2} \le 15$; точн. 0.001	вещ	простая переменная	---

10. Алгоритм



11. Программный код

Алгоритм не будем оформлять в виде отдельной процедуры, а просто вставим в основной алгоритм вместо заглушки.

Описание ещё трех переменных:

```
    {описания других промежуточных данных: }
  int np=0;
  double si, s=0;  // возьмём двойную точность для промежуточных вычислений
```

Вместо заглушки в раздел операторов программного кода:

```
    {A0.2 - обработка}
for(i=0; i<n; i++){ // перебираем по-очереди все n точек
        si=sqrt(x[i]*x[i] + y[i]*y[i]); //вычисляем удаление точки от центра координат
        s= s+si;                  // и сумму расстояний
        if (x[i]>0 && y[i]>0 &&  si>r)    // и если точка в нужной области, то
                np=np+1; ;                 // количество также увеличиваем
}

p= (double)np/n *100; //или p=100.0*np/n;  // окончательные результаты: процент и сред.удаление
sr= s / n;
```

Получившиеся программы
Лабораторная работа 3:

```
#include <stdlib.h> // printf scanf
#include <math.h> // sqrt  M_PI

/*верхняя граница индексов массива – максимальное количество точек*/
#define Nmax 20

int main()
{
   /*описание переменных:*/
   int n,r,i;
   float p, sr;
   float x[Nmax], y[Nmax];

   /*A0.1 - ввод-вывод входных данных*/
   //вывод заголовка в выходной документ:
   printf("%40s\n", "Лаб.2"); // с выравниванием и переходом на след.строку

        //ввод исходных данных:
        printf("Количество точек n:"); scanf("%d",&n); //обр 1.1,1.2
        printf("Критическое удаление r:"); scanf("%d",&r); //обр.2.1,2.2
        //ввод массивов x и y:
        printf("Абсциссы и ординаты:\n"); //обр3.1
        for (i=0; i<n; i++)
                scanf("%f %f", &x[i], &y[i]); //обр3.2:ввод 2 элементов через пробел и переход на след.строку

        //вывод входных данных в выходной документ для подтверждения:
        for(i=1; i<=40; i++) printf("="); //отделим визуально чертой и строкой введенные и выводимые значения
        printf("\n");

        //вывод n, r по обр5-6
        printf("Количество точек = %d\n", n);
        printf("Критическое удаление = %d\n", r);
        //вывод массивов x и y
        printf("Абсциссы и ординаты: \n"); //обр7
        for( i=0; i<n; i++)
                printf("%10.1f%13.1f\n", x[i], y[i]); //обр8 10=5+5 13=8+5

        /*A0.2 - обработка}*/
        int np=0;
        double si,s=0;
        for(i=0; i<n; i++){ // перебираем по-очереди все n точек
                si=sqrt(x[i]*x[i] + y[i]*y[i]);//вычисляем удаление точки от центра координат
                s= s+si;                    // и сумму
                if (x[i]>0 && y[i]>0 &&  si>r)    // если точка в нужной области, то
                        np=np+1; ;                 // и количество увеличиваем
        }

        p= (double)np/n *100; //или p=100.0*np/n;
        sr= s / n;

        /*A0.3 - вывод результатов*/
        printf("Искомый процент точек = %5.1f\n", p);    //p по обр9
        printf("Среднее удаление = %8.5f\n", sr);          //sr по обр10

   printf("The End");  // getch(); // задержать экран и #include <stdio.h>
   return 0;
}
```

Если был бы альтернативный ответ для p=100%:

```
        if (p==100) printf("Все 100% точек попали в область\n");
        else printf("Искомый процент точек = %5.1f\n", p);        //p по обр9
```

Лабораторная работа 4: (ЕЩЕ НЕ ИСПРАВЛЕН)- КОД НА ПАСКАЛЕ

```
#include <stdlib.h> // printf scanf FILE fprintf fscanf
#include <math.h> // sqrt  M_PI
#include <locale.h> // setlocale, LC_ALL

/*верхняя граница индексов массива – максимальное количество точек*/
#define Nmax 20

int main( int argc, char* argv[])  // количество переданных программе параметров и их значения(стрОки)
{
   /*описание переменных:*/
   int n,r,i;
   float p, sr;
   float x[Nmax], y[Nmax];
   FILE *fin,*fout;   { файловые переменные (указатели на дескрипторы файлов), для работы с текстов. файлами}
   setlocale(LC_ALL, "");  //– один из способов руссификации, или system("chcp 65001"); UTF8 , ("chcp 1251"); ANSI
   //вывод заголовка на экран:
   printf("%40s\n", "Лаб.2 запущена. Проверка параметров:"); // с выравниванием и переходом на след.строку

   if (argc<3) { printf("Не заданы параметры – имена файлов"); return 1;}
   printf("Количество переданных параметров =%d\nЗапущена программа %s\n", argc, argv[0]);
   for(i=1; i<argc; i++){ printf("%d) %s\n", i, argv[i]);}
   //открываем 1й файл для чтения, 2й создаём для записи "w" (или добавления"a")
   fout=fopen( argv[2], "w");
   if (fout==NULL) { printf("Не удалось создать файл %s", argv[2]); return 2;}
   fin=fopen( argv[1], "r");
   if (fin==NULL) { printf("Не удалось открыть файл %s", argv[1]); fclose(fout); return 3;}

   /*A0.1 - ввод-вывод входных данных  ТЕПЕРЬ ИЗ ФАЙЛА*/
         //ввод исходных данных БЕЗ ПРИГЛАШЕНИЙ:
         fscanf(fin,"%d",&n); //обр 1.1,1.2
         fscanf(fin, "%d",&r); //обр.2.1,2.2
         //ввод массивов x и y:
         for (i=0; i<n; i++)
               fscanf(fin,"%f %f", &x[i], &y[i]); //обр3.2:ввод 2 элементов через пробел и переход на след.строку

         //вывод n, r по обр5-6
         fprintf(fout, "Количество точек = %d\n", n);
         fprintf(fout, "Критическое удаление = %d\n", r);
         //вывод массивов x и y
         fprintf(fout, "Абсциссы и ординаты: \n"); //обр7
         for( i=0; i<n; i++)
               fprintf(fout, "%10.1f%13.1f\n", x[i], y[i]); //обр8 10=5+5 13=8+5
         //конец вывода входных данных в выходной документ для подтверждения:
         for(i=1; i<=40; i++) fprintf(fout, "-"); //отделим визуально чертой и строкой исходные данные и результат
         fprintf(fout, "\n");

         /*A0.2 - обработка}*/
         int np=0;
         double si,s=0;
         for(i=0; i<n; i++){ // перебираем по-очереди все n точек
               si=sqrt(x[i]*x[i] + y[i]*y[i]);//вычисляем удаление точки от центра координат
               s= s+si;                    // и сумму
               if (x[i]>0 && y[i]>0 &&  si>r)    // если точка в нужной области, то
                     np=np+1; ;                  // и количество увеличиваем
         }

         p= (double)np/n *100; //или p=100.0*np/n;
         sr= s / n;
         /*A0.3 - вывод результатов*/
         fprintf(fout, "Искомый процент точек = %5.1f\n", p);       //p по обр9
         fprintf(fout, "Среднее удаление = %8.5f\n", sr); //sr по обр10
   fclose(fin); fclose(fout); // закрываем файлы, входной можно раньше – сразу после ввода всех исх.дан.
   printf("The End");  // если нужно: getch(); - задержать экран и #include <stdio.h>  или system("pause");
   return 0;
}
```