МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образования "Витебский государственный технологический университет"

ИНФОРМАТИКА

Методические указания и задания к типовому расчету по теме «Программирование на алгоритмическом языке» для студентов технологических специальностей

ВИТЕБСК
2014

УДК 004

Информатика: методические указания и задания к типовому расчету по теме «Программирование на алгоритмическом языке» для студентов технологических специальностей

Витебск: Министерство образования Республики Беларусь, УО "ВГТУ", 2014.

Составитель: доц. Казаков В. Е.

В настоящих методических указаниях приведена методика разработки программ, рассмотрены вопросы построения алгоритмов, а также представлены требования к оформлению и защите типового расчёта.

Методические указания предназначены для использования при выполнении типовых расчетов по курсу «Информатика» и подготовке к их защите.

Одобрено кафедрой информатики УО “ВГТУ”

3 сентября 2014 г., протокол № 1.

Рецензент к.т.н., доц. Шарстнёв В.Л.
Редактор к.т.н., доц. Дягилев А.С.

Рекомендовано к опубликованию редакционно-издательским советом УО "ВГТУ"
“27” ноября 2014 г., протокол № 8.

Ответственный за выпуск Шарпалёв М.В.

Учреждение образования «Витебский государственный технологический университет»

Подписано к печати 19.01.15. Формат 60x90 1/16. Уч.-изд. листов. 1.7.
Печать ризографическая. Тираж 40 экз. Заказ № 15.

Отпечатано на ризографе учреждения образования «Витебский государственный технологический университет».
Свидетельство о государственной регистрации издателя, изготовителя, распространителя печатных изданий № 1/172 от 12.02.2014 г.
210035, Витебск, Московский пр-т, 72

Содержание

Введение 4
Методика решения задач 4
Постановка задачи. 5
Математическое описание задачи. 5
Выбор и обоснование метода решения. 6
Разработка алгоритма 7
Разработка программы на алгоритмическом языке высокого уровня 14
Использование программного продукта. 17
Содержание отчёта о решении задачи 18
Защита работы 18
Задания к типовому расчету 22
Задание №1 22
Задание №2 23
Задание №3 25
Указания к решению заданий: 25
Список рекомендуемой литературы 27

Введение

Целью данных методических указаний является помощь студентам первого курса КТФ в выполнении типового расчёта по курсу «Информатика: программирование на алгоритмическом языке».

В методических указаниях изложена методика, включающая: математическое исследование задачи, разработку вычислительного алгоритма, его реализации посредством современного языка, тестирование программы.

Цель выполнения типового расчёта – демонстрация студентом навыков самостоятельного решения задач с помощью алгоритмического языка высокого уровня, приобретённых при выполнении лабораторных работ под контролем преподавателя.

Методика решения задач

Решение задачи можно разделить на несколько этапов:

1) исследование задачи;
2) математическое описание задачи;
3) выбор и обоснование метода решения;
4) алгоритмизация вычислительного процесса;
5) реализация алгоритма на языке программирования;
6) тестирование программы;
7) решение задачи на ЭВМ и анализ результатов;
8) оформление отчёта о решении задачи;
9) подготовка к защите.

Перечисленные этапы связаны друг с другом. Например, анализ результатов может показать необходимость внесения изменений в программу, алгоритм или даже в постановку задачи. Для уменьшения числа подобных изменений необходимо на каждом этапе по возможности учитывать требования, предъявляемые последующими этапами. В некоторых случаях связь между различными этапами, например, между постановкой задачи и выбором метода решения, между составлением алгоритма и программированием может быть настолько тесной, что разделение их становится затруднительным.

Рассмотрим последовательно каждый из вышеперечисленных этапов применительно к решению следующей задачи.

Имеется заданный пользователем массив из n чисел (n задаётся пользователем). Необходимо отсортировать элементы массива по убыванию.

Постановка задачи

На данном этапе формулируется цель решения задачи и подробно описывается ее содержание. Анализируются характер и сущность всех величин, используемых в задаче, и определяются условия, при которых она решается. Корректность постановки задачи является важным моментом, так как от нее в значительной степени зависят другие этапы.

Для рассматриваемой задачи входными данными являются: n (целочисленное значение, больше 0), и последовательность чисел, которую можно сохранить в одномерный массив. Поскольку в условии задачи отсутствуют сведения о типе чисел в массиве (целые, вещественные и т. д.), то выберем наиболее общий случай – вещественные числа. Выходные данные: массив с отсортированными по возрастанию значениями из исходного массива. Поскольку в задаче требуется именно отсортировать исходный массив, а не записать его элементы в другой массив, то новый массив в программе создаваться не будет.

Математическое описание задачи

Этап характеризуется математической формализацией задачи, при которой существующие соотношения между величинами, определяющими результат, выражаются посредством математических формул. Так формируется математическая модель явления с определенной точностью, допущениями и ограничениями. При этом в зависимости от специфики решаемой задачи могут быть использованы различные разделы математики и других дисциплин.

Математическая модель должна удовлетворять, по крайней мере, двум требованиям: реалистичности и реализуемости. Под реалистичностью понимается правильное отражение моделью наиболее существенных черт исследуемого явления.

Реализуемость достигается разумной абстракцией, отвлечением от второстепенных деталей, чтобы свести задачу к проблеме с известным решением. Условием реализуемости является возможность практического выполнения необходимых вычислений за отведенное время при доступных затратах требуемых ресурсов.

Для рассматриваемой задачи математическая постановка будет выглядеть следующим образом.

Имеется последовательность

$$A = \{a_i\}, i \in 1..n .$$

Преобразовать последовательность так, чтобы

$$a_1 >= a_2 >= \ldots >= a_n$$

Выбор и обоснование метода решения

Модель решения задачи с учетом ее особенностей должна быть доведена до решения при помощи конкретных методов решения. Само по себе математическое описание задачи в большинстве случаев трудно перевести на язык машины. Выбор и использование метода решения задачи позволяет привести решение задачи к конкретным машинным операциям. При обосновании выбора метода необходимо учитывать различные факторы и условия, в том числе точность вычислений, время решения задачи на ЭВМ, требуемый объем памяти и другие.

Одну и ту же задачу можно решить различными методами, при этом в рамках каждого метода можно составить различные алгоритмы.

Задача сортировки одномерного массива имеет множество методов решения отличающихся сложностью реализации, временем сортировки, оптимальностью и объёмом дополнительно используемой памяти.

Вот некоторые из них:

Сортировка выбором – выполняется поиск наименьшего или наибольшего элемента (в зависимости от порядка сортировки) и помещение его в начало или конец упорядоченного массива.

Сортировка пузырьком – для каждой пары смежных элементов массива производится обмен, если элементы расположены не в соответствии порядку сортировки.

Сортировка вставками – определяем, где текущий элемент должен находиться в упорядоченном массиве, и вставляем его туда.

Сортировка Шелла – метод основан на попытке улучшить сортировку вставками.

Среди алгоритмов сортировки имеются и так называемые непрактичные алгоритмы:

Bogosort – произвольно перемешать массив, проверить, не является ли полученный массив упорядоченным.

Сортировка перестановкой – для каждой пары элементов массива осуществляется проверка верного порядка и генерируются всевозможные перестановки элементов исходного массива.

Решающими факторами для выбора метода сортировки в нашем случае станут:

небольшой размер сортируемого массива;

отсутствие необходимости использовать дополнительную память;

отсутствие в условии задачи требований к скорости выполнения сортировки;

относительная простота и наглядность алгоритма.

Для решения задачи выберем метод сортировки выбором. Дополнительно заметим, что скорость работы метода зависит от размера массива и не зависит от степени его первоначальной упорядоченности, т. е. даже если исходный

массив уже является отсортированным, метод будет выполняться столько же времени, сколько потребуется на обработку неупорядоченного массива.

Разработка алгоритма

Алгоритм – точное указание, которое определяет процесс, обработки исходных данных и получение требуемых результатов. Алгоритмами, например, являются правила сложения, умножения, решения алгебраических уравнений, умножения матриц и т. п.

Слово алгоритм происходит от algoritmi, являющегося латинской транслитерацией арабского имени хорезмийского математика IX века аль-Хорезми. Благодаря латинскому переводу трактата аль-Хорезми европейцы в XII веке познакомились с позиционной системой счисления, и в средневековой Европе алгоритмом называлась десятичная позиционная система счисления и правила счета в ней.

Применительно к ЭВМ алгоритм определяет вычислительный процесс обработки некоторой совокупности исходных данных, направленный на получение определенных этими исходными данными результатов. Термин «вычислительный процесс» распространяется и на обработку других видов информации, например, символьной, графической, звуковой и т. д.

Для успешного применения любой алгоритм должен обладать следующими основными свойствами.

Результативность означает возможность получения результата после выполнения алгоритма, например, в виде выводимых на экран полученных значений.

Конечность требует, чтобы алгоритм обязательно заканчивался, т. е. не приводил к зацикливаниям.

Определенность состоит в совпадении получаемых результатов независимо от пользователя и применяемых технических средств.

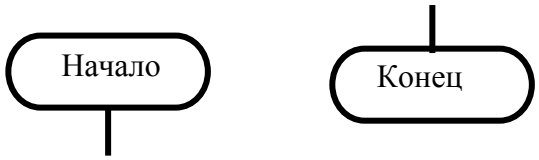
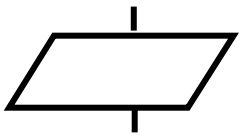
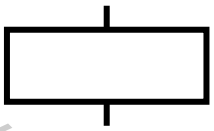
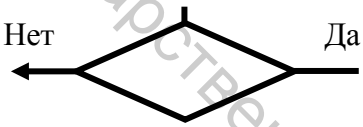
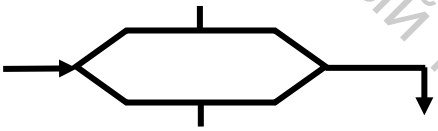
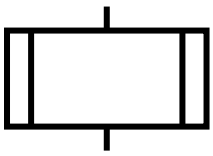
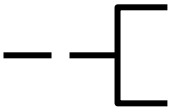
Массовость заключается в возможности применения алгоритма к целому классу однотипных задач, различающихся конкретными значениями исходных данных.

На данном этапе составляется алгоритм решения задачи согласно выбранному методу её решения. Процесс обработки данных разбивается на отдельные относительно самостоятельные операции (алгоритмические действия), и устанавливается последовательность их выполнения. Описать алгоритм можно различными способами, однако стандартизированным методом является блок-схема.

Для начертания схем алгоритмов используется набор символов, определяемых ГОСТ 19.701 — 90 (ИСО 5807 — 85) «Единая система программной документации». Схема, составленная из этих символов, называется блок-схемой.

Блоки основных алгоритмических операций представлены в таблице 1.

Таблица 1 – Стандартные обозначения алгоритмических действий

№	Обозначение	Алгоритмическое действие
1		Начало и конец алгоритма
2		Ввод или вывод информации
3		Вычисление и изменение значения переменной
4		Проверка условия и разветвление
5		Заголовок цикла с параметром
6		Вызов подпрограммы
7		Внутристраничные соединительные блоки
8		Межстраничные соединительные блоки
9		Комментарий к блоку

При разработке схем алгоритмов необходимо соблюдать некоторые требования.

- В схеме алгоритма все линии от блока «начало» до блока «конец» не должны иметь разрывов, не помеченных соединителями. Все линии, указывающие последовательность выполнения действий, должны быть замкнутыми.
- В схеме должны четко прослеживаться потоки информации. Блоки следует размещать таким образом, чтобы избегать пересечения линий. При передаче управления в схеме «снизу-вверх» или «справа—налево» линии обязательно помечают стрелками.
- Блоки следует размещать таким образом, чтобы избегать пересечения линий. При передаче управления в схеме «снизу-вверх» или «справа-налево» линии обязательно помечают стрелками.
- Не допускается передача управления «в никуда», т. е. «источник» передачи управления и «получатель» должны быть четко обозначены.

Рассмотрим основные виды алгоритмов и их представление с помощью блок-схем.

Процесс обработки данных при решении задачи, как уже упоминалось ранее, разбивается на отдельные относительно самостоятельные блоки. Рассмотрим некоторые стандартные алгоритмические конструкции и их описание с использованием блок-схем.

Линейные вычисления

В линейной части алгоритма все операции выполняются последовательно одна за другой.

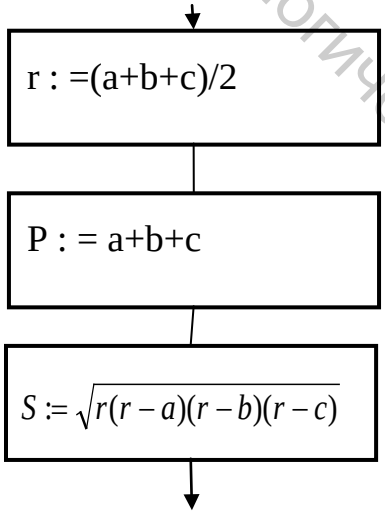


Рисунок 1 – Линейный алгоритм

Блок-схема вычислительной части алгоритма, вычисления периметра (r) и площади (S) треугольника, у которого заданы длины сторон (a, b, c), представлена на рисунке 1.

Ветвление

Разветвленные структуры применяются, когда в зависимости от некоторого условия необходимо выполнить либо одно, либо другое действие.

Примером может служить алгоритм выбора большего из двух заданных значений a и b (рисунок 2), который используется как составная часть многих алгоритмов, таких как сортировка или поиск элементов в массиве данных.

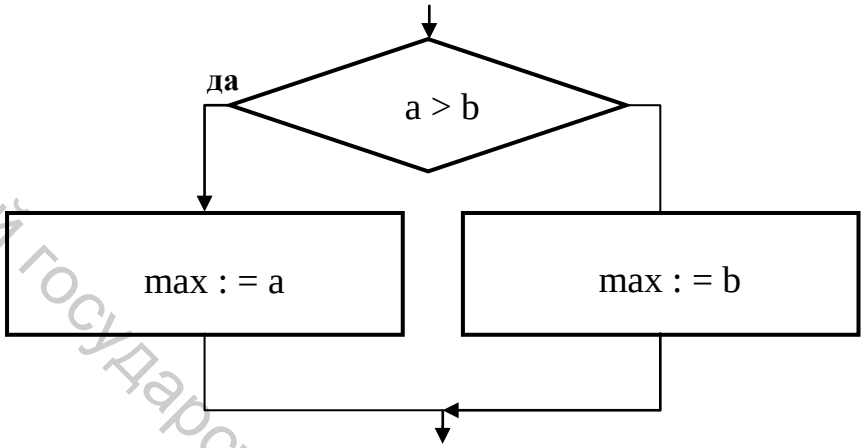


Рисунок 2 – Разветвляющийся алгоритм

Цикл со счётчиком

Циклы со счётчиком используются в том случае, когда количество повторений определённой части алгоритма (тела цикла) может быть определено в момент запуска цикла.

Наиболее часто циклы со счётчиком используются для обработки массивов и строк (которые, по сути, являются одномерными массивами, состоящими из символьных элементов).

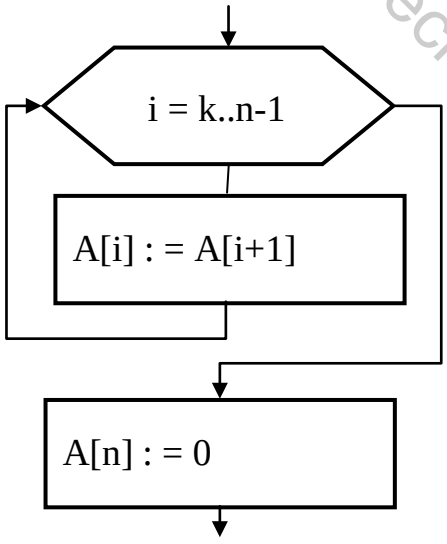


Рисунок 3 – Цикл с параметром

На рисунке 3 приведена блок-схема алгоритма удаления элемента с заданным индексом (k) из массива (A) со сдвигом остальных элементов влево (рисунок 3).

Цикл с условием

Имеется целый класс задач, заранее определить количество повторений определённых действий при решении которых затруднительно либо является нерациональным например ведущим к неоправданному усложнению алгоритма или к дополнительным вычислениям, увеличивающим время его работы. Такие задачи решаются при помощи так называемых циклов с условием.

Пример использования такого цикла, при решении задачи вывода цифр, из которых состоит целое число N представлен на рисунке 4.

Суть метода основана на последовательном целочисленном делении на 10 начального значения (операция div) и нахождении остатка от деления на 10 на каждом шаге цикла при помощи операции mod. Например, число 234, записанное в переменной N будет обработано за три прохода цикла таким образом:

1) 234 mod 10 = 4 (цифра 4 выводится на экран), 234 div 10 = 23 (полученное значение 23 записывается в переменную N);

2) 23 mod 10 = 3 (цифра 3 выводится на экран), 23 div 10 = 2 (полученное значение 2 записывается в переменную N);

3) 2 mod 10 = 2 (цифра 2 выводиться на экран), 2 div 10 = 0 (полученное значение 0 записывается в переменную N и вызывает окончание цикла по условию N>=1);

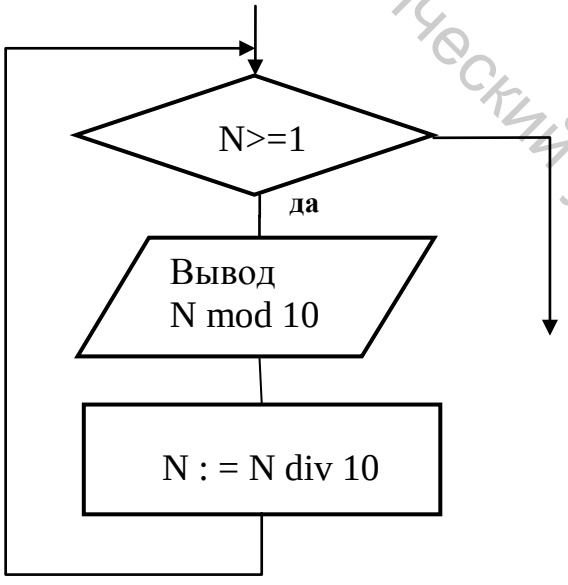


Рисунок 4 – Цикл с предусловием

Следует обратить внимание на то, что внутри цикла находиться линейная часть алгоритма: операция вывода и присваивания.

Ту же самую задачу можно решить и с помощью цикла с постусловием (рисунок 5). Однако в этом случае нужно учитывать, что тело цикла обязательно будет выполнено хотя бы один раз.

Нужно отметить, что представленный алгоритм требует, чтобы значение N до начала работы цикла было больше 1.

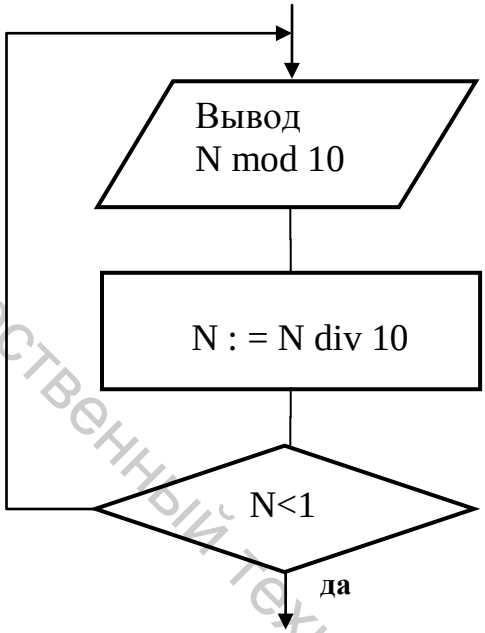


Рисунок 5 – Цикл с постусловием

Рассмотрим законченный алгоритм рассматриваемой в качестве примера задачи сортировки одномерного массива (рисунок 6).

Весь алгоритм можно разделить на несколько частей, выполняющихся последовательно: ввод размера массива (n), ввод элементов массива (A), обработка массива A, вывод элементов массива A.

Ввод элементов массива выполняется с помощью конечного цикла. Блок 3 является заголовком, а тело цикла составляет блок ввода (4).

Обработка массива выполняется также с помощью конечного цикла 5, счётчик цикла i изменяется от 1 до значения n-1. Таким образом, счётчик цикла принимает значения индексов элементов массива с первого до предпоследнего. Таким образом, тело цикла, состоящее из блоков 6, 7, 8, 9 и 10, будет выполнено для всех элементов массива, кроме предпоследнего.

Блоки 6, 7, 8, 9 и 10 составляют алгоритм с одним входом $A_{5>6}$ и одним выходом $A_{6>5}$ (рис. 6). От внешнего цикла алгоритм получает значение i. Затем в конечном цикле (блок 6) при помощи счётчика j перебираются индексы элементов массива A, начиная с i+1-го до последнего.

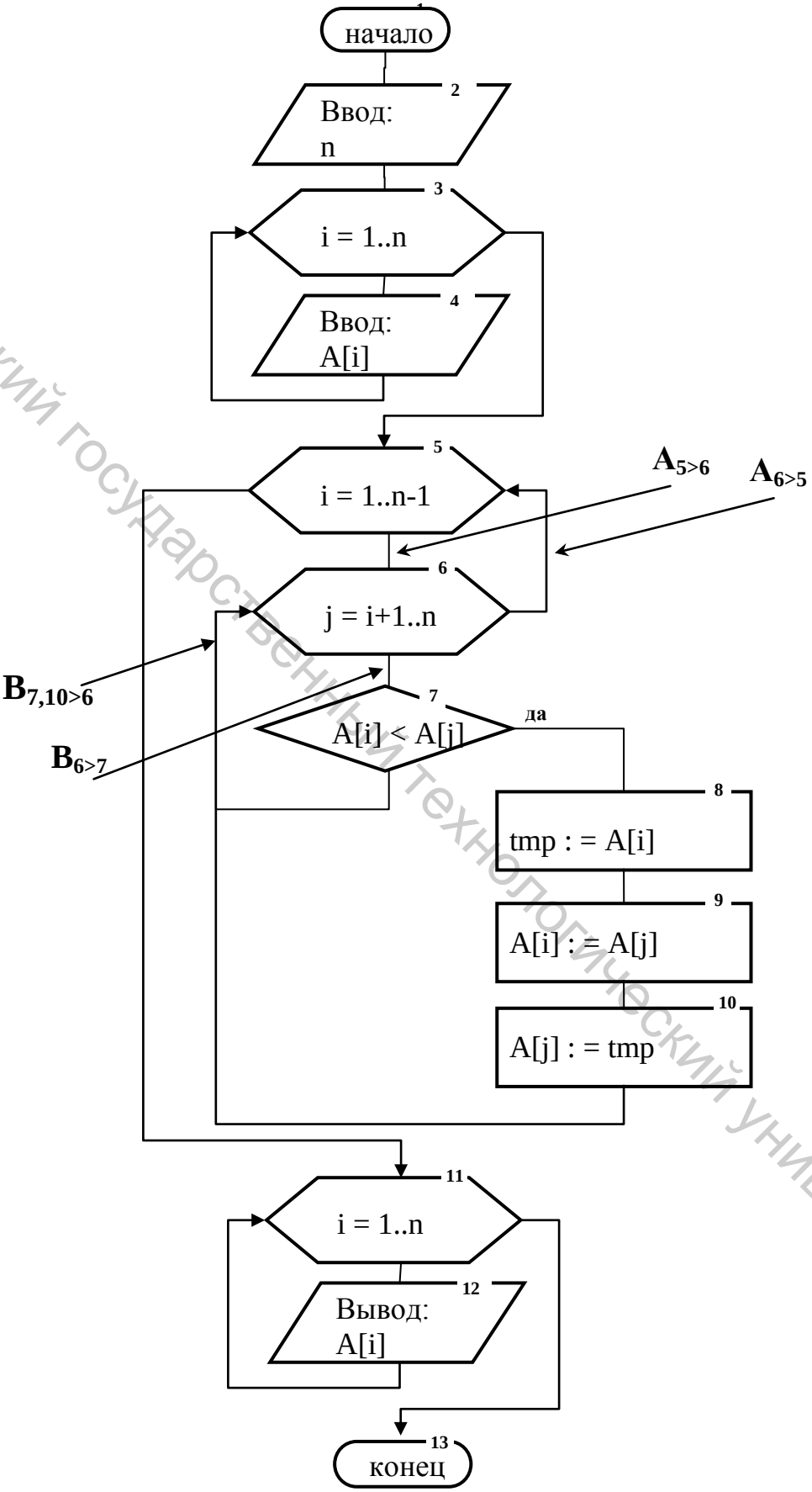


Рисунок 6 – Алгоритм сортировки одномерного массива

В теле цикла 6 при помощи блока разветвления 7 сравнивается элемент с индексом i и элемент с индексом j. Если массив должен быть отсортирован в порядке убывания значений элементов, то элемент с индексом i должен быть больше элемента с индексом j, в противном случае запускается линейный алгоритм перестановки двух элементов: значение элемента A[i] сохраняется во временной переменной tmp (блок 8), значение элемента A[j] копируется в ячейку A[i] (блок 9), значение из переменной tmp записывается в ячейку A[j] (блок 10).

Завершающая часть алгоритма – вывод элементов массива – является конечным циклом. Как видно из рисунка 6, эта часть алгоритма имеет такую же структуру, как и часть, отвечающая за ввод элементов массива. Разница состоит лишь в использовании блока вывода вместо блока ввода.

Можно отметить, что сложный алгоритм состоит из некоторого количества более простых алгоритмов, а те, в свою очередь, состоят из элементарных алгоритмических конструкций.

Разработка программы на алгоритмическом языке высокого уровня

Разработанный на предыдущем этапе алгоритм должен быть реализован в исходном тексте программы на одном из алгоритмических языков высокого уровня. При этом каждый блок алгоритма заменяется соответствующим оператором или конструкцией языка.

Необходимо также помнить о необходимости описания всех структур хранения данных (переменных) в специальном разделе текста программы.

На данном этапе программа проходит проверку на соответствие алгоритму, разработанному на предыдущем этапе.

```
program sort;
var
   a: array[1..100] of real;
   i, j, n: byte;
   tmp: real;

begin
   write('n>');
   readln(n);
   for i := 1 to n do
   begin
     write('A[',i,']>');
     readln(a[i]);
   end;
  for i:=1 to n-1 do
```

```
    for j:=i+1 to n do
      if a[i]<a[j] then
        begin
          tmp:=a[i];
          a[i]:=a[j];
          a[j]:=tmp;
        end;
    for i := 1 to n do
        write(a[i]:4:1,'   ');
    readln;
    end.
```

Компиляция программы

После составления исходного текста программы необходимо её откомпилировать, т. е. перевести исходный текст на языке программирования в исполняемый файл с расширением exe. В ходе компиляции выявляются синтаксические ошибки, т. е. такие ошибки, из-за которых невозможно преобразовать исходный текст в исполняемый файл.

Сообщения об ошибках компилятора во всех современных средах разработки программ содержат текстовое описание ошибки и указывают на строку в исходном тексте, которая её вызвала.

Компилятор находит синтаксические ошибки сразу во всём тексте программы. Исправлять ошибки рекомендуется в порядке появления их тексте программы, поскольку в большинстве случаев ранее по тексту допущенная ошибка влечёт за собой следующие и после исправления первой, последующие устраняются автоматически.

Наиболее частые ошибки, допускаемые при решении задач типового расчёта, приведены в таблице 2.

Таблица 2 – Ошибки синтаксиса.

Сообщение об ошибке	**Перевод**	**Наиболее частая причина**
E2003 Undeclared identifier: 'Идентификатор'	Необъявленный идентификатор:	Опечатка в написании стандартного идентификатора либо использование необъявленной в разделе var переменной
E2066 Missing operator or semicolon	Отсутствует оператор или точка с запятой	Оператор не завершается «;»

Окончание таблицы 2

Сообщение об ошибке	**Перевод**	**Наиболее частая причина**
E2010 Incompatible types: 'Integer' and 'Extended'	Несовместимые типы: 'Integer' и 'Extended'	Попытка использовать вещественное значение ('Extended') там, где требуется целочисленное ('Integer'), например, в операторе присваивания
E2029 ')' expected but ';' found	')' Ожидается, но ';' найдено	Не хватает закрывающей скобки «)» в выражении
E2029 'END' expected but end of file found	'END' ожидается, но найден конец файла	Не хватает закрывающей операторной скобки «end» в тексте программы

Тестирование

Отладка программы. Отладка заключается в поиске и устранении логических ошибок в программе.

После устранения синтаксических ошибок проверяется логика работы программы в процессе ее выполнения с конкретными исходными данными. Для этого составляется несколько наборов входных значений. Такие наборы должны обеспечить проверку работы программы при различных условиях. Затем для каждого теста вручную рассчитываются результаты, которые должна получить программа. Эти результаты сверяются со значениями, получаемыми при выполнении отлаживаемой программы.

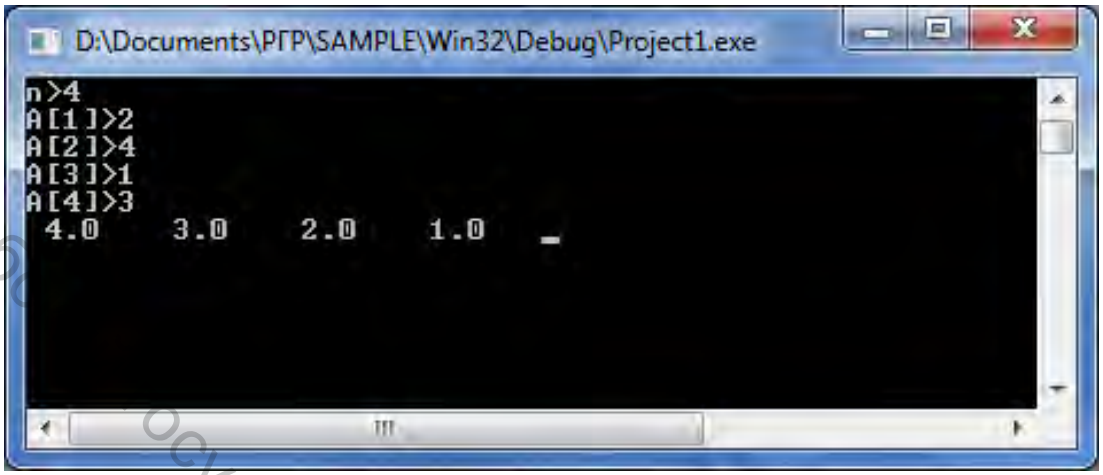
Для поиска логических ошибок используются отладчики, исполняющие программу в пошаговом режиме с возможностью просмотра значений переменных программы.

Для рассматриваемого примера составим два теста. Первый будет содержать натуральные неповторяющиеся числа (рисунок 7 а). Второй тест покажет, как программа отреагирует на ввод нулевых и отрицательных значений, а также в случае, если будут введены повторяющиеся значения (рисунок 7 б).

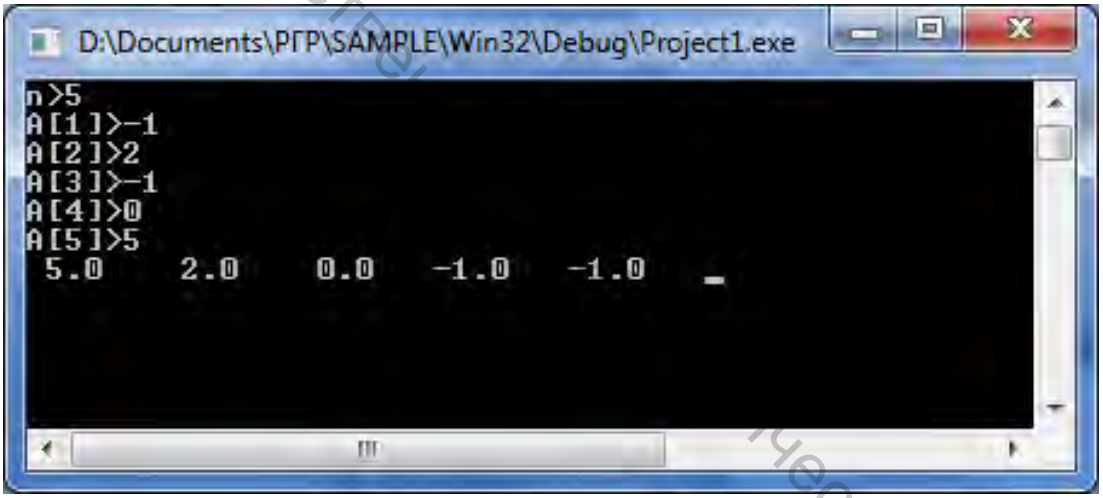
Конечно, данные тесты далеко не полностью представляют работу программы (например, можно было провести тест на обработку уже упорядоченного массива), однако они демонстрируют работу программы при вводе корректных значений, а также в некоторых критических условиях.

Для представления результатов тестирования в работе необходимо привести экран работы программы. Для копирования экрана программы с тестовым примером необходимо нажать комбинацию клавиш Ctrl+PrtSc, после чего изображение текущего окна в растровом формате будет скопировано в

буфер обмена. После этого можно при помощи Ctrl+V вставить это изображение в файл отчёта.



а



б

Рисунок 7 – Экраны работы программы при разных входных данных

Использование программного продукта

После отладки программы ее можно использовать для решения прикладных задач. При этом обычно выполняется многократное использование программы для решения конкретных прикладных задач пользователя. Получаемые результаты интерпретируются и анализируются специалистом или пользователем, поставившим задачу.

Разработанная программа длительного использования устанавливается на ЭВМ, как правило, в виде готовой к выполнению машинной программы. К программе прилагается документация, включая инструкцию для пользователя.

Чаще всего при установке программы на диск для ее последующего использования помимо файлов с исполняемым кодом устанавливаются различные вспомогательные программы (утилиты, справочники, настройщики и т. д.), а также необходимые для работы программ разного рода файлы с текстовой, графической, звуковой и другой информацией.

Содержание отчёта о решении задачи

Отчёт о решении задачи должен содержать:

1) полный текст задания;
2) описание метода решения задачи (если требуется);
3) блок-схему вычислительного алгоритма, выполненную в соответствии с требованиями ГОСТ 19.701 — 90 (ИСО 5807 — 85);
4) исходный текст программы на языке Pascal;
5) результаты тестовых запусков.

Защита работы

Цель защиты – доказательство того, что разработанный студентом алгоритм решает поставленную задачу и обладает всеми требуемыми свойствами, а программа на алгоритмическом языке соответствует схеме разработанного алгоритма и не приводит к возникновению ошибок во время исполнения при вводе различных входных значений.

На защите студент должен быть готов:

- обосновать выбор метода решения задачи;
- описать назначение каждой из переменных;
- описать назначение каждой части алгоритма;
- объяснить, как будет изменяться работа программы при вводе различных входных данных;
- описать альтернативный метод решения задачи.

Рассмотрим, каким образом можно подготовиться к защите решения задачи сортировки одномерного массива.

В качестве обоснования выбора метода решения задачи можно воспользоваться тезисами, представленными в подразделе **«Выбор и обоснование метода решения»** данных методических указаний.

Для описания назначений переменных и частей программы можно использовать примечания, размещённые в исходном коде:

```
program sort;
var
{массив для хранения исходных значений, в данном массиве будут
```

осуществляться перестановки элементов, по окончании работы алгоритма в нём

будет размещаться элементы отсортированные в порядке возрастания их значений}

```
    a: array[1..100] of real;
{служебные переменные для перебора элементов}
    i, j : byte;
{переменная для хранения количества элементов в массиве}
n: byte;
{переменная, используемая для временного хранения значений при перестановке элементов массива }
   tmp: real;
begin
    {ввод количества элементов массива}
    write('n>');
    readln(n);
    {ввод элементов массива}
    for i := 1 to n do
    begin
     write('A[',i,']>');
     readln(a[i]);
    end;
{--------------сортировка массива---------------}
  {цикл для последовательного выбора элемента, расположенного на вершине сортировки}
  for i:=1 to n-1 do
    {цикл для перебора элементов, расположенных «ниже» вершины сортировки для поиска наибольшего из них}
     for j:=i+1 to n do
     {если среди элементов, расположенных «ниже» вершины сортировки, находится элемент с большим значением, чем значение элемента, расположенного на вершине сортировки…}
      if a[i]<a[j] then
       begin
       {… то меняем местами найденный элемент с элементом, расположенном на вершине сортировки}
        tmp:=a[i];
        a[i]:=a[j];
        a[j]:=tmp;
       end;
{--------окончание сортировки-----------}
    {вывод отсортированного массива}
```

```
    for i := 1 to n do
      write(a[i]:4:1,'   ');

    {обеспечение остановки работы программы до нажатия на кнопку
<Enter> для просмотра результатов, выведенных на экран}
    readln;
    end.
```

Для демонстрации работы программы при заданных входных данных может быть использована так называемая трассировка (выполнение программы по шагам и контроль значений в основных переменных программы). Для трассировки используем блок-схему программы, представленную на рисунке 6. Каждый блок на блок-схеме соответствует определённому оператору в исходном тексте, следовательно, проследить за ходом работы программы по исходному тексту также не составит труда.

Будем рассматривать часть программы, отвечающую непосредственно за сортировку массива, считая, что значения в массив уже введены и не рассматривая вывод результатов. В таблице 3 представлена трассировка фрагмента программы сортировки. Жирным выделены значения переменных, которые изменил указанный в строке оператор, серой заливкой выделен элемент, расположенный на вершине сортировки, стрелками в массиве А показаны перестановки элементов.

Таблица 3 – Трассировка программы

№ блока	i	A[i]	j	A[j]	массив A			
					1	2	3	4
…								
5	**1**	**22**	-	-	22	33	11	55
6	1	22	**2**	**33**	22	33	11	55
7	1	22	2	33	22	33	11	55
8,9,10	1	**33**	2	**22**	**33**	**22**	11	55
6	1	33	**3**	**11**	33	22	11	55
7	1	33	3	11	33	22	11	55
6	1	33	**4**	**55**	33	22	11	55
7	1	33	4	55	33	22	11	55
8,9,10	**1**	**55**	**4**	**33**	**55**	22	11	**33**
6	**1**	**55**	4	33	55	22	11	33

Окончание таблицы 3

№ блока	i	A[i]	j	A[j]	массив A			
					1	2	3	4
5	**2**	**22**	**-**	**-**	55	22	11	33
6	2	22	**3**	**11**	55	22	11	33
7	2	22	3	11	55	22	11	33
6	2	22	**4**	**33**	55	22	11	33
7	2	22	4	33	55	22	11	33
8,9,10	**2**	**33**	**4**	**22**	55	**33**	11	**22**
6	2	33	4	22	55	33	11	22
5	**3**	**11**	-	-	55	33	11	22
6	3	11	**4**	**22**	55	33	11	22
7	3	11	4	22	55	33	11	22
8,9,10	**3**	**22**	**4**	**11**	55	33	**22**	**11**
6	3	22	4	11	55	33	22	11
5	3	22	4	11	55	33	22	11
…								

В качестве альтернативного метода решения задачи можно упомянуть так называемый «пузырьковый» метод сортировки, который также подходит для сортировки небольших массивов, без использования дополнительной памяти.

Также можно упомянуть не оптимальность представленного алгоритма: если внимательно проследить за преобразованием массива по таблице 3, то можно заметить лишнюю перестановку элементов при первом проход метода (i = 1). Максимальный элемент массива – 55, и его нужно переместить на вершину сортировки, однако вначале на вершину сортировки перемещается элемент 33.

Это происходит из-за того, что при нахождении очередного элемента, большего, чем элемент на вершине сортировки, они тут же меняются местами.

Устранить такую избыточность можно так: воспользовавшись стандартным алгоритмом поиска максимального элемента массива найти позицию максимального элемента из всех, находящихся «под» вершиной сортировки, элементов, а затем один раз осуществить перестановку.

Задания к типовому расчету

Задание № 1

0. Локатор ориентирован на одну из сторон света (1 — север, 2 — запад, 3 — юг, 4 — восток) и может принимать три цифровые команды поворота: 1 — поворот налево, «–1» — поворот направо, 2 — поворот на 180°. Дана исходная ориентация локатора и целые числа N1 и N2 — две посланные команды. Вывести ориентацию локатора после выполнения этих команд.

1. Элементы окружности пронумерованы следующим образом: 1 — радиус R, 2 — диаметр D = 2·R, 3 — длина L = 2·π·R. Дан номер одного из этих элементов и его значение. Вывести значения остальных элементов данной окружности (в том же порядке).

2. Дан номер года (положительное целое число). Определить количество дней в этом году, учитывая, что обычный год насчитывает 365 дней, а високосный — 366 дней. Високосным считается год, делящийся на 4, за исключением тех годов, которые делятся на 100 и не делятся на 400 (например, годы 300, 1300 и 1900 не являются високосными, а 1200 и 2000 — являются).

3. Даны целочисленные координаты точки на плоскости. Если точка совпадает с началом координат, то вывести 0. Если точка не совпадает с началом координат, но лежит на оси OX или OY, то вывести соответственно 1 или 2. Если точка не лежит на координатных осях, то вывести 3.

4. Даны четыре вещественных значения. Вывести наибольшее из их.

5. Даны координаты точки, не лежащей на координатных осях OX и OY. Определить номер координатной четверти, в которой находится данная точка.

6. Дано целое число. Вывести его строку-описание вида «отрицательное четное число», «нулевое число», «положительное нечетное число» и т. д.

7. Арифметические действия над числами пронумерованы следующим образом: 1 — сложение, 2 — вычитание, 3 — умножение, 4 — деление. Дан номер действия N (целое число в диапазоне 1–4) и вещественные числа A и B (B не равно 0). Выполнить над числами указанное действие и вывести результат.

8. Элементы равнобедренного прямоугольного треугольника пронумерованы следующим образом: 1 — катет a, 2 — гипотенуза c = a·(2)$^{1/2}$, 3 — площадь S = c · h / 2. Дан номер одного из этих элементов и его значение. Вывести значения остальных элементов данного треугольника (в том же порядке).

9. Единицы длины пронумерованы следующим образом: 1 — дециметр, 2 — километр, 3 — метр, 4 — миллиметр, 5 — сантиметр. Дан номер единицы длины (целое число в диапазоне 1–5) и длина отрезка в этих единицах (вещественное число). Найти длину отрезка в метрах.

Задание №2

1. Вычислить сумму элементов бесконечного ряда с заданной пользователем точностью:

$$S = \sum_{n=1}^{\infty} (-1)^n \frac{x^n \cdot \sin(nx)}{(2n+1)!}$$

2. Вычислить сумму элементов бесконечного ряда с заданной пользователем точностью:

$$S = \sum_{n=3}^{\infty} (-1)^n \frac{\sin^{n+3}(2n+x)}{(2n-1)!}$$

3. Вычислить сумму элементов бесконечного ряда с заданной пользователем точностью:

$$S = \sum_{n=1}^{\infty} (-1)^{2n+1} \frac{\sin(2n+x)}{(2n+1)!}$$

4. Вычислить сумму элементов бесконечного ряда с заданной пользователем точностью:

$$S = \sum_{n=1}^{\infty} (-1)^{n+2} \frac{\sin(2nx)+2n}{(2n+3)!}$$

5. Вычислить сумму элементов бесконечного ряда с заданной пользователем точностью:

$$S = \sum_{n=1}^{\infty} (-1)^n \frac{x^{n+3}}{(2n+3)!+(\mathrm{n}+3)}$$

6. Вычислить сумму элементов бесконечного ряда с заданной пользователем точностью:

$$S = \sum_{n=1}^{\infty} \left(\frac{x^{n+2}}{(3n)!} + \sin(nx) \right)$$

7. Вычислить сумму элементов бесконечного ряда с заданной пользователем точностью:

$$S = \sum_{n=1}^{\infty} \frac{(x+5)^{2n-1}}{(2n+3)!}$$

8. Вычислить сумму элементов бесконечного ряда с заданной пользователем точностью:

$$S = \sum_{n=1}^{\infty} (-1)^{2n} \frac{2n \cdot x^{n+3}}{(3n)!}$$

9. Вычислить сумму элементов бесконечного ряда с заданной пользователем точностью:

$$S = \sum_{n=1}^{\infty} \left(\frac{(x+1)^{2n}}{(2n+3)!} + \cos x \right)$$

10. Вычислить сумму элементов бесконечного ряда с заданной пользователем точностью:

$$S = \sum_{n=1}^{\infty} (-1)^{n+1} \frac{(x-1)^{n+3}}{(3n)!+\mathrm{x}}$$

11. Вычислить сумму элементов бесконечного ряда с заданной пользователем точностью:

$$S = \sum_{n=1}^{\infty} (-1)^{2n-1} \frac{2n + x^n}{(3n-2)!}$$

12. Вычислить сумму элементов бесконечного ряда с заданной пользователем точностью:

$$S = \sum_{n=1}^{\infty} (-1)^n \frac{x^{n-2}}{(2n-1)! + \sin(4n+3)}$$

13. Вычислить сумму элементов бесконечного ряда с заданной пользователем точностью:

$$S = \sum_{n=1}^{\infty} (-1)^{2n} \frac{n^x}{(2n+1)!}$$

14. Вычислить сумму элементов бесконечного ряда с заданной пользователем точностью:

$$S = \sum_{n=1}^{\infty} \left(\frac{x^{2-n}}{(n+3)!} + \sin(2n) \right)$$

15. Вычислить сумму элементов бесконечного ряда с заданной пользователем точностью:

$$S = \sum_{n=1}^{\infty} (-1)^{2n+1} \frac{(x-1)^n}{(2n)!}$$

Задание № 3

Указания к решению заданий.

Для множества порядок следования элементов, в отличие от последовательности, не важен, т. е. элементы множества можно переставлять (например, при помощи сортировки), а последовательности – нет.

1. Дано множество из N значений. Выяснить, можно ли составить из этих значений арифметическую прогрессию с шагом 1 (например:2,3,4,5 или 4,5,6,7,8).

2. Дана последовательность из *N* целых чисел, содержащая, по крайней мере, два нуля. Вывести сумму чисел из данного набора, расположенных между последними двумя нулями.

3. Дана последовательность из N чисел. Найти сумму элементов с индексами, большими, чем индекс максимального значения (т. е. расположенных “за” максимальным значением).

4. Дано множество из N значений. Найти элемент с наиболее близким значением к среднеарифметическому элементов множества.

5. Даны две упорядоченные по возрастанию последовательности из N и M значений соответственно. Объединить их в новую упорядоченную по возрастанию последовательность.

6. Даны два множества из N значений. Верно ли, что множества содержат одинаковые значения (например: 1, 3, 2 и 2, 3, 1 – одинаковы; 1, 3, 5 и 1, 2, 5 – различны).

7. Даны две последовательности из N значений. Верно ли, что отрицательный элемент в первой последовательности встречается раньше, чем во второй? Предполагается, что каждая из последовательностей содержит хотя бы один отрицательный элемент.

8. Дана последовательность из *N* чисел. Найти минимальное значение последовательности и увеличить каждый её элемент на это значение.

9. Дана последовательность из *N* чисел, содержащая по крайней мере два нуля. Вывести сумму чисел из данного набора, расположенных между последними двумя нулями.

10. Дано множество из N значений. Найти количество уникальных значений (значение не должно повторяться).

11. Даны два множества из N значений. Верно ли, что множества содержат различные значения (например: 1, 3, 2 и 2, 3, 1 – одинаковы; 1, 3, 5 и 1, 2, 5 – различны).

12. Дано множество из N значений. Найти значения двух ближайших элементов из этого множества (то есть элементов с наименьшим модулем разности).

13. Дано множество из N значений. Составить новое множество, в которое войдут только положительные значения из заданного множества.

14. Дано множество из N значений. Найти два элемента, сумма которых максимальна.

15. Дано множество из N значений. Подсчитать количество значений, меньших среднеарифметического всех значений множества.

Список рекомендуемой литературы

1. Казаков, В. Е. Информатика. Программирование на алгоритмическом языке: конспект лекций / В. Е. Казаков. – Витебск : УО «ВГТУ», 2010. – 93 с.

2. Информатика, численные методы и компьютерная графика : методические указания по выполнению лабораторных работ для студентов дневного отделения спец. конструкторско-технологических факультетов. Ч. 1 / УО "ВГТУ" ; сост. Е. А. Калиновская [и др.]. – Витебск : УО "ВГТУ", 2004.

3. Информатика, численные методы и компьютерная графика: методические указания к лабораторным работам для студентов технологических спец. дневной и заочной формы обучения. Ч. 2 / УО "ВГТУ" ; сост. В. П. Терентьев [и др.]. - Витебск: УО "ВГТУ", 2005.

4. Фаронов, В. В. Delphi. Программирование на языке высокого уровня: учебник для студентов вузов, обучающихся по направлению подготовки дипломированных специалистов "Информатика и вычислительная техника" / В. В. Фаронов. – Санкт-Петербург : Питер, 2010, 2007. – 640 с

5. Бородич, Ю. С. Паскаль для персональных компьютеров : справочное пособие / Ю. С. Бородич, А. Н. Вальвачев, А. И. Кузьмич. – Минск: Вышэйшая школа, 1991. - 365 с.

6. Руководство по языку B.Pascal 7 & Objects/LR [Электронный ресурс] – 2000. – Режим доступа: http://citforum.ru/programming/bp70_lr/index.shtml – Дата доступа: 9.09.2014.

7. Курс «Программирование на языке высокого уровня Паскаль» (Лекции №№1-4) [Электронный ресурс] – 2010. – Режим доступа: http://www.intuit.ru/studies/courses/628/484/info – Дата доступа: 9.09.2014.

8. Основы алгоритмизации и программирования [Электронный ресурс] – 2007. – Режим доступа: http://256bit.ru/informat/index11.htm. – Дата доступа: 9.09.2014.