Министерство образования и науки Российской Федерации

Государственное образовательное учреждение
высшего профессионального образования
«Санкт–Петербургский государственный технологический институт
(технический университет)»

Кафедра систем автоматизированного проектирования и управления

А.Ю. Рогов

ТЕХНОЛОГИИ ПРОГРАММИРОВАНИЯ

Методические указания к выполнению контрольных работ
для студентов заочной формы обучения
направления подготовки «Информатика и вычислительная техника»

Санкт-Петербург
2010

УДК 681.3.06

Рогов А.Ю. Технологии программирования: Методические указания к выполнению контрольных работ/ СПб.: СПбГТИ(ТУ), 2010.- 60 с.

В методических указаниях приводятся задания по трём контрольным работам, требования и комментарии к их выполнению, а также примеры их решения. Контрольные работы предназначены для индивидуального выполнения и подразумевают, что студент ознакомлен с теоретическим материалом, изложенным в учебном пособии к дисциплине «Технологии программирования». Для выполнения контрольных работ студенту также требуются знания хотя бы одного языка программирования высокого уровня, которые получены им из дисциплины «Алгоритмические языки и программирование».

Материал указаний разделен на три раздела соответственно каждой контрольной работе. Каждый из разделов содержит цель, задание, требования к выполнению, комментарии, список пунктов оформления, варианты заданий, и пример выполнения с комментариями.

Методические указания предназначены для студентов 3 курса заочной формы обучения направления подготовки 230100 «Информатика и вычислительная техника» соответствует рабочей программе учебной дисциплины «Технологии программирования».

Рис. 19, табл. 15, библиогр. 16 назв.

Рецензент:

В.К. Викторов , д-р техн. наук, профессор кафедры информационных систем в химической технологии Санкт-Петербургского государственного технологического института (технического университета)

Утверждены на заседании учебно-методической комиссии факультета Информатики и Управления 24.05.2010 протокол № 8.

Рекомендованы к изданию РИСо СПбГТИ(ТУ)

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Целью дисциплины «Технологии программирования» является изучение основ программирования и технологии разработки программных продуктов, ознакомление с задачами и этапами разработки программных продуктов, освоение современных алгоритмических языков, формирование практических навыков разработки алгоритмов, освоение современных систем программирования.

В результате изучения практической части дисциплины студенты получить знания по основным понятиям и терминологии программирования, задачам и этапам проектирования программных продуктов, технологическим средствам программирования, декомпозиции задач, алгоритмическим конструкциям, структурированию данных, тестированию программных продуктов, подготовке программной документации, оценке качества программных продуктов, планированию работ по созданию программ.

При выполнении контрольных заданий студенты должны приобрести навыки декомпозиции задач, составления алгоритмов, структурирования данных, написания собственных программ, тетирования программ, и подготовки документации.

1 УКАЗАНИЯ К ВЫПОЛНЕНИЮ И ОФОРМЛЕНИЮ КОНТРОЛЬНЫХ РАБОТ

Общие требования к работам:

- Работы выполняются индивидуально.
- Номер варианта выбирается по последней цифре номера зачетной книжки.
- При выполнении работы помимо задания должны быть учтены требования и комментарии.
- Выполненная работа демонстрируется преподавателю. В процессе демонстрации студент должен понимать и уметь объяснить все этапы работы.
- В разделе «Оформление» к каждой работе указано, что должен содержать отчёт по работе.
- Все работы подшиваются в одну общую папку с общим титульным листом и записываются на CD в электронном виде. Папка с диском сдаётся в конце семестра, и является основанием для допуска к зачёту.
- На зачёте студент должен ответить на контрольные и дополнительные вопросы.

Требования к оформлению отчетов:

- Отчет оформляется на листах формата А4 шрифтом Times 14 одиночным интервалом.
- Каждая задача начинается с нового листа.
- Допускается двусторонняя печать.
- Рукописные тексты и рисунки в отчетах не допускаются.
- Скриншоты экранов не допускаются.
- Блок-схемы только в печатном виде с помощью фигур Word или Visio.
- Уравнения выполняются с помощью Office Equation Editor.

Требования к оформлению листингов программ:

- Перед каждой разработанной функцией должен быть комментарий, рассказывающий, что эта функция делает, что возвращает и какие аргументы принимает.
- Листинги программ оформляются шрифтом Courier 11 одиночным интервалом.
- В начале листинга должна быть информация о тех, кто писал программу ФИО и номер группы, и назначении программы.
- В теле функции должны быть поясняющие комментарии, если части алгоритма не являются очевидными.

При выполнении заданий необходимо использовать знания, полученные из курса «Алгоритмические языки и программирование» по языкам C#, C++, VB, Pascal.

2 КОНТРОЛЬНАЯ РАБОТА № 1

2.1 Теоретическая часть

Цель:

Ознакомление со средой программирования, приобретение навыков разработки алгоритмов ветвления, декомпозиции задач и составления тест-планов.

Задание:

Даны несколько фигур согласно вариантам, которые разбивают плоскость на области. Координатные оси не считаются. Вводятся координаты точки (x,y). Необходимо разработать алгоритм, который определяет, в какую из имеющихся на плоскости областей попадает точка с заданными координатами. Программа должна выводить сообщение о номере области.

Требования:

Интерфейс ввода координат и отображения результатов должен быть реализован в главной функции Main(). Проверка условий должна быть реализована отдельной функцией, назовем ее CheckArea(). Функция CheckArea() вызывается из Main() и возвращает номер области, в которую попала точка, не выводя никаких сообщений. Функция Main() вводит координаты точки и выводит сообщение о результате.

Комментарии:

Фигуры заданы координатами точек. Используя координаты необходимо вывести уравнения этих фигур. Далее, используя уравнения, составить условия, проверяющие попадание в каждую область. Затем составить алгоритм, последовательно проверяющий эти условия. Написать программу.

Оформление:

1. Текст задания
2. Рисунок фигур на плоскости с указанием номеров областей
3. Уравнения фигур (с выводом, где надо)
4. Перечень условий проверок попаданий по всем областям
5. Тест-план
6. Декомпозицию задач
7. Блок-схемы и описание алгоритмов
8. Листинг программы с комментариями

2.2 Задание для выполнения контрольной работы №1

1. Даны две трапеции. Первая с вершинами в точках: (0,0), (8,0), (6,6), (3,6). Вторая с вершинами в точках: (3,2), (6,2), (9,8), (0,8). В точке (8,2) находится центр окружности радиусом r=2.

2. Дана парабола с вершиной в точке (4,8) и пересекающая ось Ox в точках (0,0) и (8,0). В точке (4,8) находится центр окружности радиусом r=4. Прямая проходит параллельно оси Oy пересекает ось Ox в точке (4,0).

3. Дана парабола с вершиной в точке (4,8), пересекающая ось Ox в точках (0,0) и (8,0), прямая линия, пересекающая ось Ox в точке (8,0) и ось Oy в точке (0,4), и прямая линия, проходящая параллельно оси Oy, пересекающая ось Ox точке (6,0).

4. Даны две параболы: первая с вершиной в точке (4,8), вторая с вершиной в точке (4,4). Обе параболы пересекают ось Ox в точках (0,0) и (8,0). Прямая пересекает ось Ox в точке (8,0) и ось Oy в точке (0,8).

5. Дан квадрат с вершинами в точках: (0,0), (8,0), (8,8), (0,8). В центре квадрата – точка (4,4) – находится окружность радиусом r=3. Из центра окружности исходят две прямые линии к вершинам квадрата (0,8) и (8,8) соответственно.

6. Дана окружность с центром в точке (3,3) и радиусом r=3. Окружность рассекают две прямые. Первая прямая пересекает ось Ox в точке (8,0) и ось Oy в точке (0,8), вторая прямая пересекает ось Ox в точке (3,0) и ось Oy в точке (0,3).

7. Дан ромб с вершинами в точках: (3,0), (6,3), (3,6), (0,3). В центре ромба располагается окружность радиусом 1. Прямая пересекает ось Ox в точке (6,0) и ось Oy в точке (0,6).

8. Дана область, образованная пересечением ромба с вершинами в точках: (2,0), (4,4), (2,8), (0,4) и квадрата с вершинами в точках: (0,6), (0,2), (4,2), (4,6). В вершине ромба (4,4) находится окружность радиусом r=2.

9. Дана трапеция, опирающаяся на оси координат, а верхнее основание образует прямая y=6. Наклонное ребро проходит через точки (6,6) и (9,0). В точке (9,0) находится окружность радиусом R=3. Внутри трапеции находится вторая окружность с центром в точке (3,3) и радиусом R=2.

10. Дана парабола с вершиной в точке (4,8), пересекающая ось Ox в точках (0,0) и (8,0), ромб с вершинами в точках: (0,4), (4,0), (8,4), (4,8), окружность с центром в точке (4,4) и радиусом R=2, линия, проходящая через точки (0,0) и (8,8).

2.3 Пример выполнения контрольной работы № 1

Задание:

Даны фигуры, которые разбивают плоскость на области: трапеция с вершинами в точках: (0.5 , 0.5), (0.5 , 2.5), (4.5 , 2.5), (2.5 , 0.5), окружность с центром в точке (2,2) и радиусом r=1, и парабола с вершиной в точке (2,2), пересекающая ось Ox в точках (0,0) и (4,0). Составить алгоритм, который определяет, в какую из имеющихся на плоскости областей попадает точка с заданными координатами (x,y) и выводом соответствующего сообщения.

Графическая модель:

Построим фигуры и пронумеруем области. Графики с указанием номеров областей представлены на рисунке 1.1:

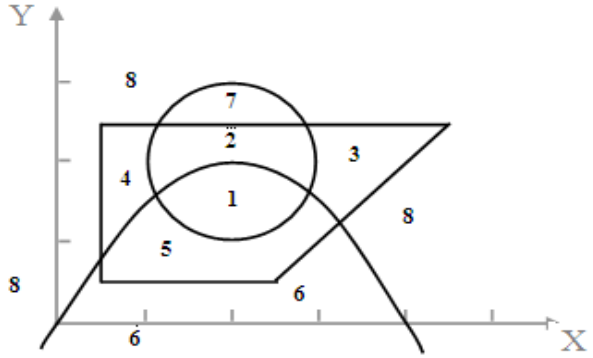


Рисунок 1.1 – Графики с указанием номеров областей

Математическая модель:

Получим математические уравнения кривых и линий, образующих фигуры на плоскости:

Таблица 1.1 - математические уравнения кривых и линий, образующих фигуры на плоскости

1	парабола	$Y = -0.5 * X^2 + 2.0 * X$
2	окружность	$(X - 2)^2 + (Y - 2)^2 = 1$
3	основание трапеции	$Y = 2.5$
4	основание трапеции	$Y = 0.5$
5	прямая сторона трапеции	$X = 0.5$
6	наклонная сторона трапеции	$Y = X - 2.0$

Составим условия проверки попадания точки с координатами (X_z , Y_z) в каждую из фигур, изображенных на рисунке 1.1:

Таблица 1.2 - условия проверки попадания точки с координатами (X_z , Y_z)

1	внутри окружности	$(X_z - 2.0)^2 + (Y_z - 2.0)^2 \le 1.0$
2	снаружи окружности	$(X_z - 2.0)^2 + (Y_z - 2.0)^2 > 1.0$
3	внутри параболы	$Y_z + 0.5 * X_z^2 - 2.0 * X_z \le 0$
4	снаружи параболы	$Y_z + 0.5 * X_z^2 - 2.0 * X_z > 0$
5	внутри трапеции	$Y_z \le 2.5 \wedge Y_z \ge 0.5 \wedge X_z \ge 0.5 \wedge (X_z - Y_z) \le 2.0$
6	снаружи трапеции	$Y_z > 2.5 \vee Y_z < 0.5 \vee X_z < 0.5 \vee (X_z - Y_z) > 2.0$

Составим условия проверки попадания точки в каждую из областей, изображенных на рисунке 1.1:

Таблица 1.3 - условия проверки попадания точки в каждую из областей

№	Окружность	Парабола	Трапеция	Доп.Условия
1	внутри	внутри	внутри	—
2	внутри	снаружи	внутри	—
3	снаружи	снаружи	внутри	$X_z > 2.0$
4	снаружи	снаружи	внутри	$X_z < 2.0$
5	снаружи	внутри	внутри	—
6	снаружи	внутри	снаружи	—
7	внутри	снаружи	снаружи	—
8	снаружи	снаружи	снаружи	—

Тест-план:

Для тестирования программы необходимо выбрать по точке в каждой из областей, используя следующие данные:

Таблица 1.4 – Координаты точек, в каждой из областей

Тест	X_z	Y_z	Результат
1	2.0	1.5	область № 1
2	2.2	2.2	область № 2
3	3.5	1.9	область № 3
4	0.9	2.0	область № 4
5	1.5	0.8	область № 5
6	3.0	0.3	область № 6
7	1.0	–1.0	область № 6
8	2.0	2.7	область № 7
9	4.1	1.1	область № 8
10	–1.1	1.1	область № 8
11	2.0	3.5	область № 8
12	ab	1.0	Incorrect X-value (повтор ввода)
13	1.0	ab	Incorrect Y-value (повтор ввода)

Декомпозиция задач:

Исходя из условий задания, в основной задаче Main можно выделить самостоятельную задачу CheckArea – проверка попадания точки в одную из областей на плоскости. Эту задачу можно разбить на три независимые подзадачи:

IsCircle – проверка попадания точки внутрь окружность,

IsParabola – проверка попадания точки внутрь параболы

IsTrapezium– проверка попадания точки внутрь трапеции.

Дерево декомпозиции задач следующее:

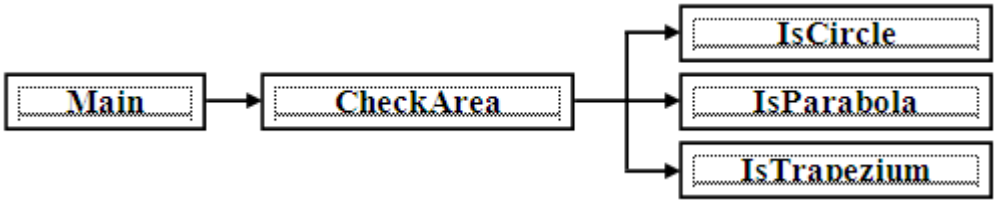


Рисунок 1.2 - Дерево декомпозиции задач

Блок-схемы и описание алгоритмов:

Алгоритм IsCircle получает координаты точки, как исходные данные, проверяет условия 1 и 2, и возвращает True, если точка лежит внутри окружности, или False, если точка лежит снаружи окружности.

Алгоритм IsParabola получает координаты точки, как исходные данные, проверяет условия 3 и 4, и возвращает True, если точка лежит внутри параболы, или False, если точка лежит снаружи параболы.

Алгоритм IsTrapezium получает координаты точки, как исходные данные, проверяет условия 5 и 6, и возвращает True, если точка лежит внутри трапеции, или False, если точка лежит снаружи трапеции.

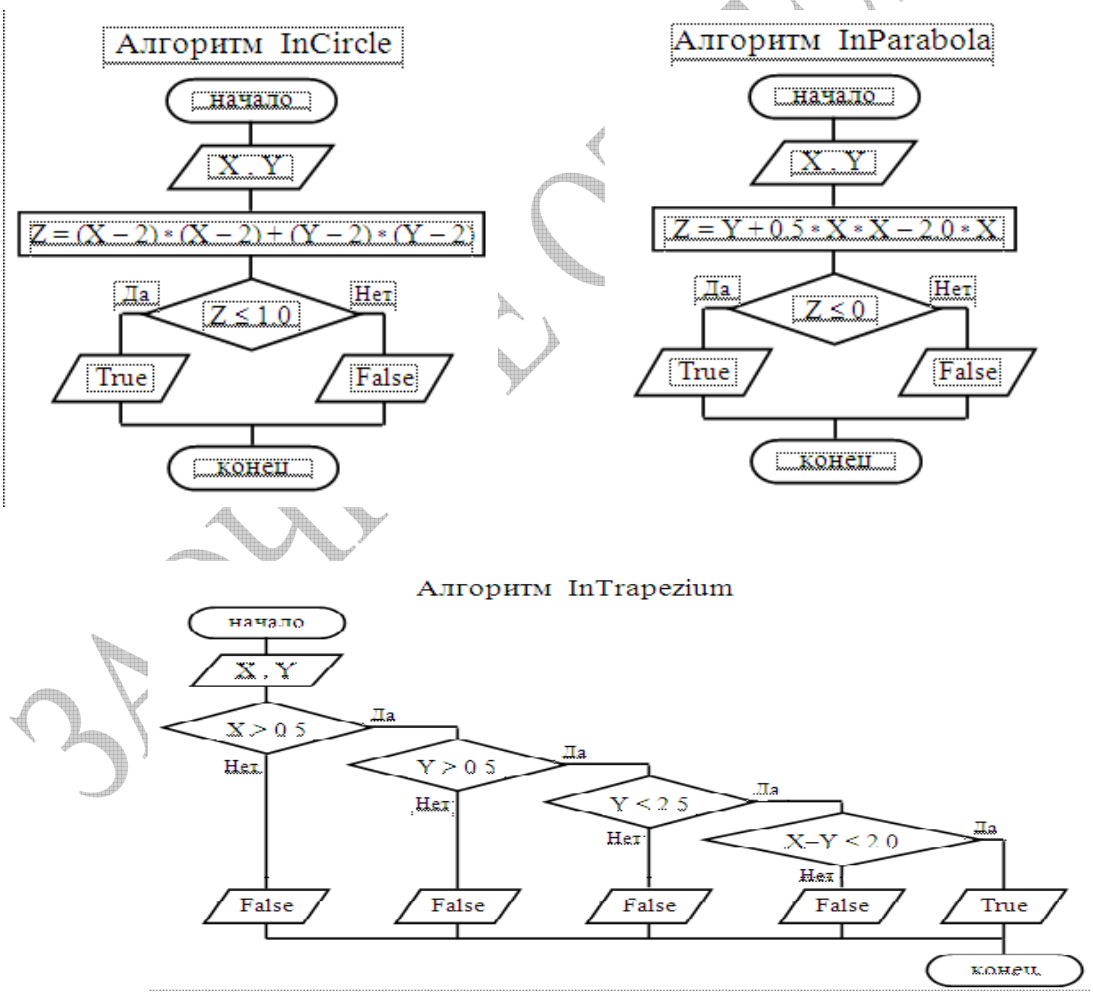


Рисунок 1.3 – Алгоритмы IsCircle, IsParabola, IsTrapezium

Алгоритм CheckArea

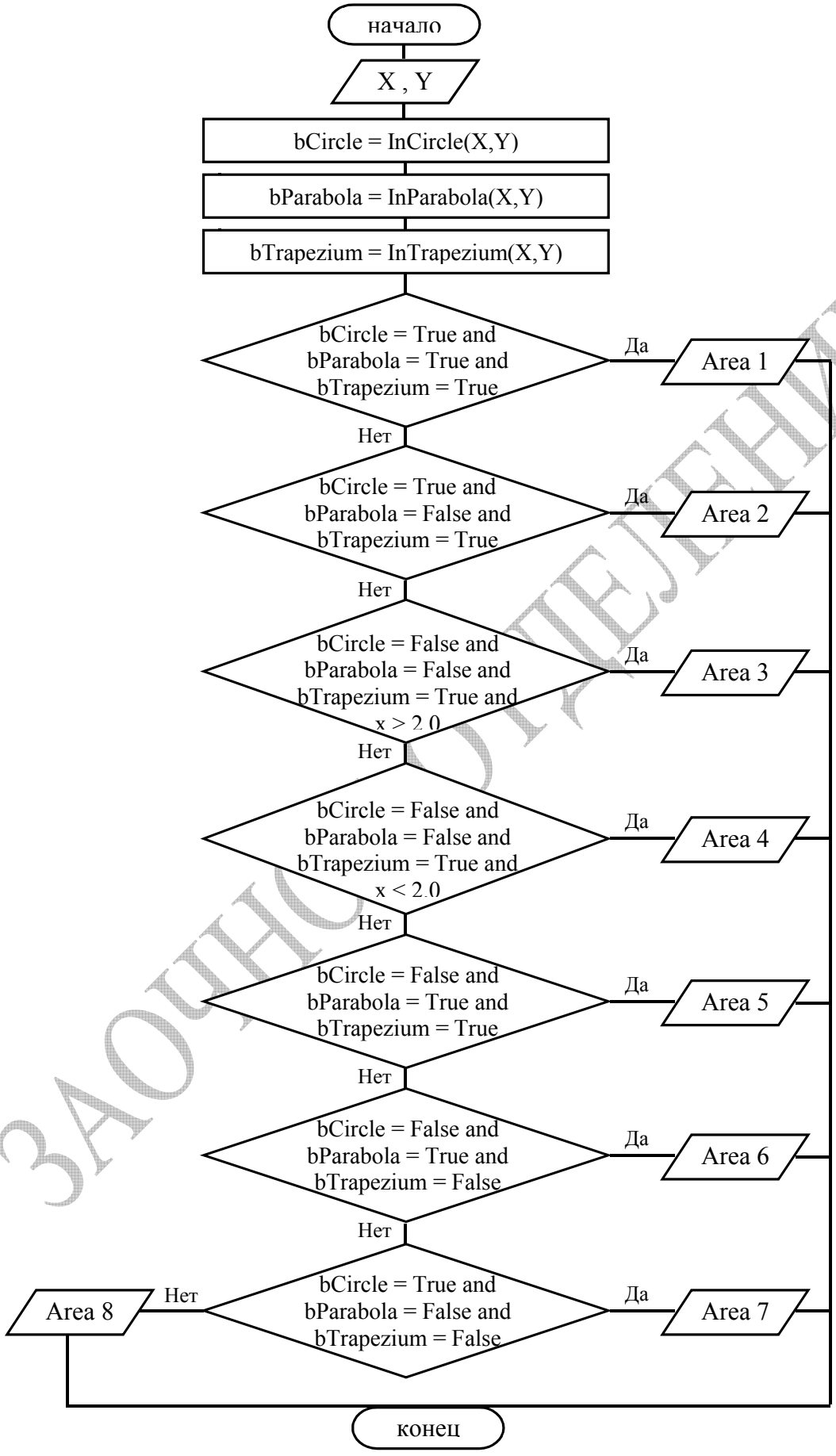


Рисунок 1.4 - Алгоритм CheckArea

Алгоритм CheckArea получает координаты точки, как исходные данные, и использует алгоритмы IsCircle, IsParabola, IsTrapezium для проверки попадания точки в каждую из фигур по отдельности и, затем, осуществляет блок последовательных проверок в соответствии с таблицей условий по областям для проверки попадания точки в каждую из областей. Область 8 определяется как исключение, если ни одно из условий не выполнилось.

Листинг программы:

```
#include <iostream>
#include <locale.h>

using namespace std;

/*
    Функция проверяет, находится ли точка с координатами (x,y)
    внутри параболы (true) или снаружи параболы (false)
*/
bool InParabola(double x, double y)
{
  return ((y + 0.5 * x * x - 2.0 * x) <= 0) ? true : false;
}

/*
    Функция проверяет, находится ли точка с координатами (x,y)
    внутри окружности (true) или снаружи окружности (false)
*/
bool InCircle(double x, double y)
{
  return (((x - 2.0)*(x - 2.0) +
           (y - 2.0)*(y - 2.0)) <= 1.0) ? true : false;
}

/*
    Функция проверяет, находится ли точка с координатами (x,y)
    внутри трапеции (true) или снаружи трапеции (false)
*/
bool InTrapezium(double x, double y)
{
  return (y <= 2.5 && y >= 0.5 &&
          x >= 0.5 && (x - y) <= 2.0) ? true : false;
}

/*
    Функция проверяет, какой области принадлежит точка
    с координатами (x,y) и возвращает номер области от 1 до 8
*/
int CheckArea(double x, double y)
{
  bool bCircle = InCircle(x,y);
  bool bParabola = InParabola(x,y);
  bool bTrapezium = InTrapezium(x,y);

  if(bCircle == true && bParabola == true &&
     bTrapezium == true) return 1;
```

```
  else if(bCircle == true && bParabola == false &&
          bTrapezium == true) return 2;

  else if(bCircle == false && bParabola == false &&
          bTrapezium == true && x > 2.0) return 3;

  else if(bCircle == false && bParabola == false &&
          bTrapezium == true && x < 2.0) return 4;

  else if(bCircle == false && bParabola == true &&
          bTrapezium == true) return 5;

  else if(bCircle == false && bParabola == true &&
          bTrapezium == false) return 6;

  else if(bCircle == true && bParabola == false &&
          bTrapezium == false) return 7;
  return 8;
}


/*
    Главная функция. Реализует интерфейс с пользователем.
*/
void main(void)
{
  char ch;
  double x,y;
  do
  {
    cout << "\nEntry X: ";
    cin >> x;
    if(cin.fail())
    {
       cin.clear();
       cout << "Incorrect X-value";
    }
    else
    {  cout << "\nEntry Y: ";
       cin >> y;
       if(cin.fail())
       {
          cin.clear();
          cout << "Incorrect Y-value";
       }
       else
       {
          cout << "\nPoint is placed in area: " <<
CheckArea(x,y);
       }
    }
    cin.sync();
    cout << "\nDo you want to entry a new point [Y/N]?";
    cin.get(ch);
  }
  while((ch == 'Y') || (ch == 'y'));
}
```

3 КОНТРОЛЬНАЯ РАБОТА № 2

3.1 Теоретическая часть

Цель:

Изучение среды программирования, отладочных средств, циклических конструкций языка, ознакомление со свойствами рядов целых чисел, приобретение навыков разработки циклических алгоритмов, преобразования данных, декомпозиции задач и составления тест-планов.

Задание:

Разработать алгоритм распознающий те числа, которые удовлетворяют заданным условиям: согласно вариантам. Программа должна позволять по выбору пользователя либо печатать числа, обладающих свойством, для заданного пользователем диапазона [*n*,*m*], либо вводить, печатать информацию и подсчитывать числа, обладающих свойством, до тех пор, пока пользователь не введет два нуля подряд (количество чисел неизвестно). Опционально программа должна распечатывать ряд чисел согласно вариантам.

Требования:

Проект должен состоятть из двух модулей кода. Консольный интерфейс с пользователем реализуется в главной функции Main(), которая находится в модуле Interface.cpp. Печать и ввод чисел реализуются отдельными функция-ми, которые тоже находятся в модуле Interface.cpp. Проверка свойств чисел реализуется отдельными функциям, которые находятся в модуле Solve.cpp. Функции проверки свойств чисел НЕ должны использовать операции с плавающей точкой, вещественные типы данных, или готовые библиотечные функции. Прототипы всех функций описываются в модуле Task.h. Интерфейс должен предоставлять опции печати ряда, печати чисел в диапазоне, ввода последовательности чисел с подсчётом.

Комментарии:

Найти описание свойств чисел, используя www.ru.wikipedia.org, согласно заданию, записать пример ряда чисел, разработать алгоритм, вычисляющий следующий член ряда через предыдущий(ие), и разработать алгоритм, разбивающий числа на цифры. Записать условие проверки числа. Разработать алгоритм по вводу последовательности чисел (количество чисел заранее не известно), придумать способ завершения ввода. Разработать алгоритмы вывода проверенных чисел.

Оформление:

1. Текст задания
2. Математическую модел (Описание свойств чисел с примерами)
3. Тестовые планы
4. Декомпозицию задач

5. Блок-схемы и описание алгоритмов
6. Листинг программы с комментариями

3.2 Задание для выполнения контрольной работы №2

1. Числа, сумма цифр которых является числом Фибоначчи и разность между соседними цифрами равна *k*. Например:

14 → 1+4=5 (*k*=3); 579 → 5+7+9=21 (*k*=2), …

2. Числа, сумма цифр которых является треугольным числом и разность между соседними цифрами равна *k*. Например:

357 → 3+5+7=15 (*k*=2); 678 → 6+7+8=21 (*k*=1), …

3. Числа, сумма цифр которых является квадратным числом и разность между соседними цифрами равна *k*. Например:

1357 → $1+3+5+7=16=4^2$, при *k*=2, …

4. Числа, сумма цифр которых является пятиугольным числом и разность между соседними цифрами равна *k*. Например:

23 → 2+3=5 (*k*=1); 25852→ 2+5+8+5+2=22 (*k*=3), …

5. Числа, сумма цифр которых является шестиугольным числом и разность между соседними цифрами равна *k*. Например:

24 → 2+4=6 (*k*=2); 1474741→ 1+4+7+4+7+4+1=28 (*k*=3), …

6. Числа, сумма цифр которых равна одному из делителей самого числа и разность между соседними цифрами равна *k*. Например:

24 → 2+4=6 (24/6=4 *k*=2); 36 → 3+6=9 (36/9=4 *k*=3), …

7. Числа, сумма цифр которых является числом Трибоначчи и разность между соседними цифрами равна *k*.

25 → 2+5=7 (*k*=2); 636306→ 6+3+6+3+0+6=24 (*k*=3), …

8. Числа, сумма цифр которых является простым числом и разность между соседними цифрами равна *k*.

74 → 7+4=11 (*k*=3); 35753 → 3+5+7+5+3=23 (*k*=2), …

9. Числа, сумма цифр которых является числом степени тройки 3^n и разность между соседними цифрами равна *k*.

36 → 3+6=9 ($9=3^2$ *k*=3); 75753 → 7+5+7+5+3=27 ($27=3^3$ *k*=2), …

10. Числа, сумма цифр которых является числом Мерсенна и разность между соседними цифрами равна *k*.

34 → 3+4=7 (*k*=1); 258 → 2+5+8=15 (*k*=3), …

3.3 Пример выполнения контрольной работы № 2

Задание:

Разработать алгоритмы распознающие и распечатывающие при вводе те числа, сумма цифр которых является числом степени двойки 2^n и разность между соседними цифрами равна k. Например:

26 → 2+6=8 ($8=2^3$ *k*=4); 25252 → 2+5+2+5+2=16 ($16=2^4$, *k*=3),…

Математическая модель: $N_{i+1} = N_i * 2$

Числа степени двойки получаются по формуле:
Последовательность чисел для n = 0, 1, 2, … начинается так:
1 , 2 , 4 , 8 , 16 , 32 , 64 , 128 , 256 , …

Тестовые планы:
Тестирование программы предлагается разбить на три части:

1. Тестирование печати чисел в заданном диапазоне

Вызвать функцию печати чисел, обладающих свойством, для заданного диапазона и использовать следующие данные:

Таблица 2.1 – Данные для вызова функции печати чисел

Тест	От	До	K	Результат
1	10	100	2	13, 20, 31, 35, 53, 79, 97
2	10	100	4	26, 40, 62
3	10	100	6	17, 71
4	10	100	8	80
5	100	10000	3	1474, 4741, 7414
6	100	10000	4	404, 484, 2626, 4040, 4048, 4840, 6262, 8404
7	100	10000	5	161, 727
8	100	10000	6	1717, 7171
9	10000	9000000	3	25252, 1414141, 5252585, 5258525, 5852525
10	10000	9000000	6	2828282
11	10000	9000000	8	8080808
12	–500	–10	4	–484, –404, –62, –40, –26
13	100	10	6	17, 71
14	ab	100	2	Incorrect lower bound value
15	10	ab	2	Incorrect upper bound value
16	10	100	A	Incorrect digit difference value
17	10	100	9	Incorrect digit difference value
18	10	100	–1	Incorrect digit difference value

2. Тестирование последовательности ввода чисел

В качестве тестовых последовательностей использовать следующие данные (жирным выделены числа, обладающие свойством):

Таблица 2.2 – Тестовые последовательности

Тест	K	Последовательность чисел											N	M
1	2	11	**31**	17	11	**97**	**242**	56	25	**1357**	66	0	4	6
2	4	11	44	**26**	**40**	23	**484**	95	37	**4048**	**62**	0	5	5
3	3	12	45	65	24	67	123	81	67	1782	60	0	0	10
4	1	**10**	**101**	**565**	**323**	**121**	**–10**	**101**	**1012**	**3212**	**10**	0	10	0
5	1	11	ab	0	**10**	ed	34	0	**1012**	3678	hh	0	2	3
6	8											0	0	0

Тесты 1 и 2 – последовательности, содержащие чисела, обладающие и не обладающие свойством. Тест 3 – последовательность, не содержащая чисел, обладающих свойством. Тест 4 – последовательность, содержащая только числа, обладающие свойством. Тест 5 – последовательность, содержащая ошибочные данные. Тест 6 – пустая последователь-ность (сразу два нуля).

3. Тестирование печати ряда степени двойки

Вызвать функцию печати чисел степени двойки. Результат:
1 , 2 , 4 , 8 , 16 , 32 , 64 , 128 , 256 , 512 , 1024 , 2048 , 4096 , 8192 , 16384, 32768 , 65536 , 131072 , 262144 , 524288 , 1048576 , 2097152 , 4194304 , 8388608 , 16777216 , 33554432 , 67108864 , 134217728 , 268435456

Декомпозиция задач:

Исходя из условий задания, главную задачу Main можно разбить на две основные задачи:

CheckNumber – проверка свойств чисел,

Entry&Print – ввод данных и печать результатов.

Задачу CheckNumber можно разбить на три независимые подзадачи:

1 – проверка числа на степень двойки,

2 – проверка разности между цифрами,

3 – вычисление суммы цифр.

Задачу Entry&Print можно разбить на три независимые подзадачи:

1 – печать ряда степени двойки,

2 – печать в заданном диапазоне чисел, удовлетворяющих свойствам,

3 – определение и подсчёт чисел, удовлетворяющих свойствам, при вводе, а

условием прекращения ввода является ввод двух нулей подряд.

Для CheckNumber предлагается использовать алгоритмы:

IsDegreeTwo – для проверки, является ли число числом степени двойки,

IsDifference– для проверки, равна ли разность между цифрами числа заданному значению,

SumDigits –для вычисления суммы цифр числа.

CheckNumber – для проверки, является сумма цифр числа числом степени двойки и разность между соседними цифрами равна k.

Для решения Entry&Print предлагается использовать алгоритмы:

PrintDegreeTwo – для печати ряда чисел степени двойки,

PrintNumbers – для печати в заданном диапазоне тех чисел, которые обладают заданными свойствами,

InputNumbers – для ввода чисел с консоли, проверки их на заданные свойства, подсчёта чисел, пока не будут введены два нуля подряд.

Дерево декомпозиции задач следующее:

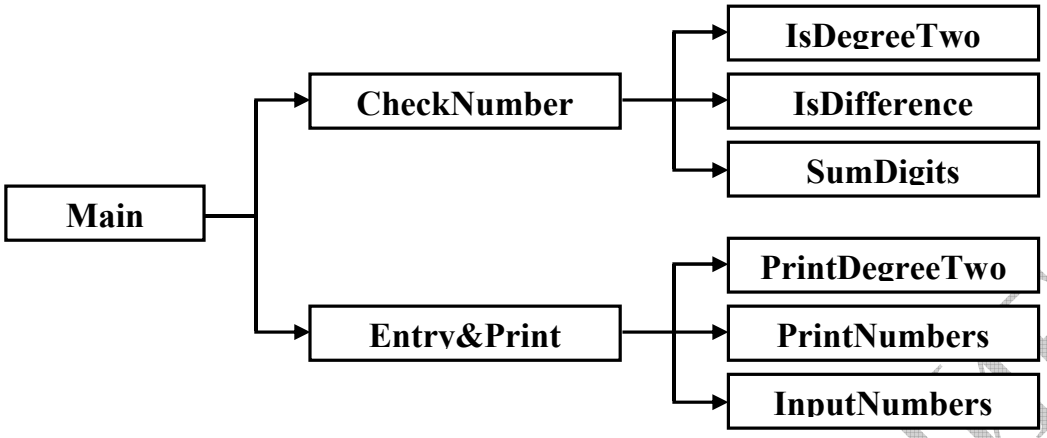


Рисунок 2.1 - Дерево декомпозиции задач

Блок-схемы и описание алгоритмов:

Алгоритм IsDegreeTwo

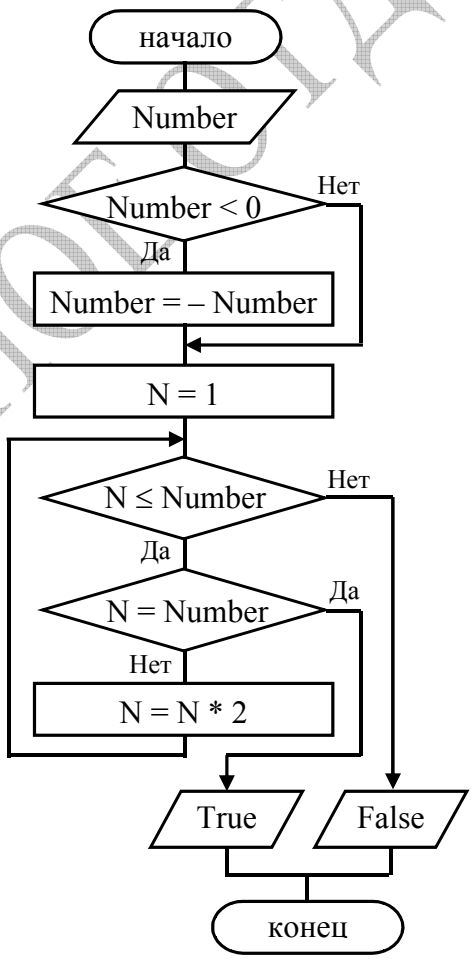


Рисунок 2.2 - Алгоритм IsDegreeTwo

Алгоритм IsDegreeTwo получает число для проверки, как исходные данные. Затем, он последовательно в цикле вычисляет следующее число степени двойки и сравнивает его с числом, переданным как исходные данные. Если текущее число степени двойки равно заданному число, то цикл прерващается, и алгоритм возвращает True, как результат. Если в процеесе вычислений, число степени двойки стало больше заданного числа, то вычисления можно прекратить, и алгоритм возвращает False, как результат.

Алгоритм SumDigits

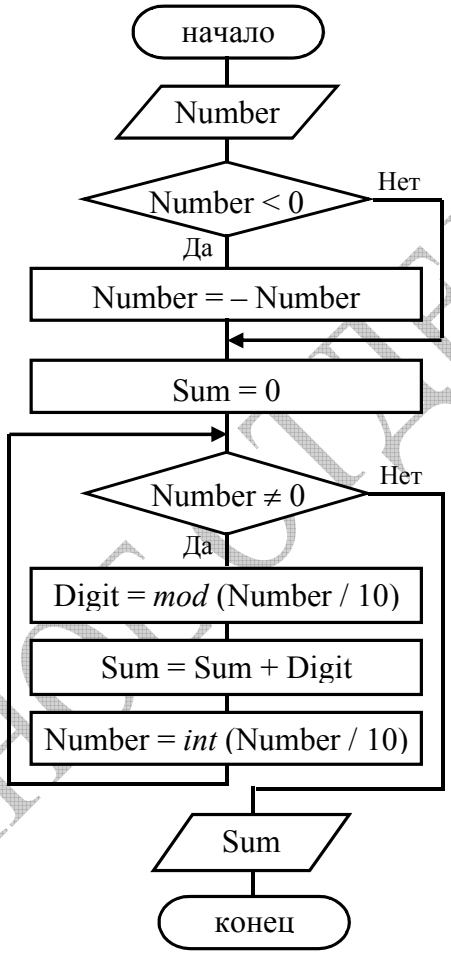


Рисунок 2.3 - Алгоритм SumDigits

Алгоритм SumDigits получает число для проверки, как исходные данные. Затем, он последовательно в цикле разбивает число на цифры и суммирует их. Чтобы отделить очередную цифру, алгоритм использует операцию вычисления остатка от деления на цело. Чтобы получить число без отделённой цифры, алгоритм использует обычную операцию деления и отбрасывает дробную часть. Цикл выполняется до тех пор, пока в числе не останется цифр. Алгоритм возвращает сумму цифр, как результат.

Алгоритм IsDifference получает число для проверки и разность между цифрами, как исходные данные. Затем, он последовательно в цикле разбивает

число на цифры, вычисляет разность между соседними цифрами и сравнивает её с заданной разностью. Чтобы отделить очередную цифру, алгоритм использует операцию вычисления остатка от деления на цело. Чтобы получить число без отделённой цифры, алгоритм использует обычную операцию деления и отбрасывает дробную часть. Цикл выполняется до тех пор, пока в числе не останется цифр. Если на очередном шаге вычислений разность оказывается больше или меньше заданной, то вычисления можно прекратить, и алгоритм возвращает False, как результат. Если в числе больше не осталось цифр, то цикл прерващается, и алгоритм возвращает True, как результат.

Алгоритм CheckNumber

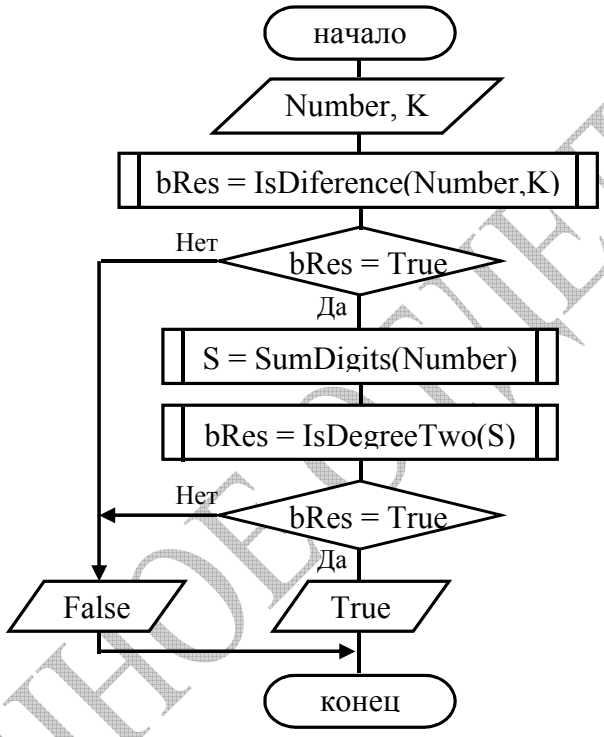


Рисунок 2.4 - Алгоритм CheckNumber

Алгоритм CheckNumber получает число для проверки и разность между цифрами, как исходные данные. Затем, он последовательно использует алгоритмы IsDegreeTwo, IsDifference и SumDigits, описанные выше, для проверки, является ли сумма цифр числом степени двойки при заданной разности между соседними цифрами. Алгоритм возвращает True, если результаты всех проверок оказались True, или False, если хотя бы одна проверка не выполнилась.

Алгоритм IsDifference

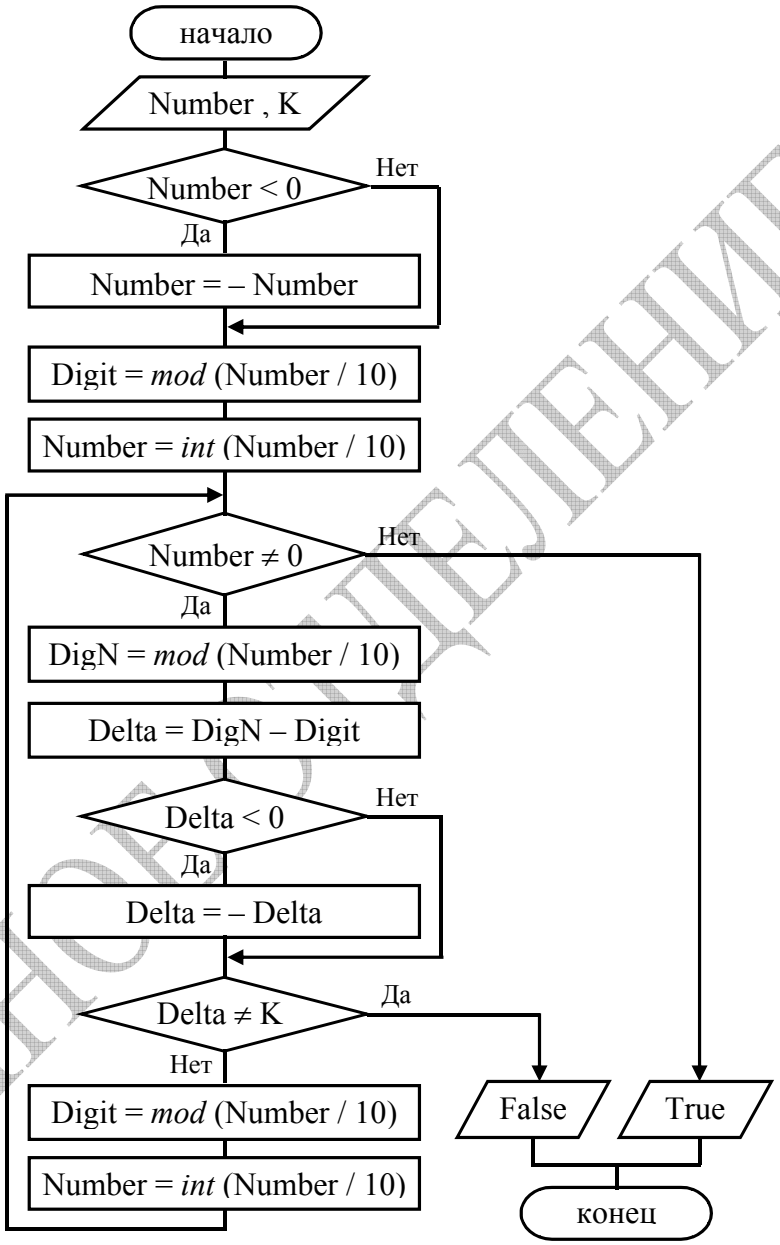


Рисунок 2.5 - Алгоритм IsDifference

Алгоритм InputNumbers запрашивает только разность между цифрами, количество вводимых чисел ему не известно. Алгоритм организует цикл по вводу чисел с проверкой каждого нового числа до пор, пока пользователь не введет два нуля подряд. Для проверки введенного числа он использует алгоритм CheckNumber. Если введенное число удовлетворяет свойствам, то алгоритм увеличивает счётчик проверяемых чисел (N) на единицу и печатет

информацию о числе, если нет, то увеличивает счётчик прочих чисел (M). Чтобы проверить ввод двух нулей подряд, алгоритм использует переменную Zero, которая получает значение True, всякий раз, когда был введен ноль, и значение False, всякий раз, когда было введено ненулевое значение. Если был введен первый ноль, т.е. Zero еще False, то алгоритм присваиват введенному числу ненулевое значение, чтобы не прекращать цикл.

Алгоритм InputNumbers

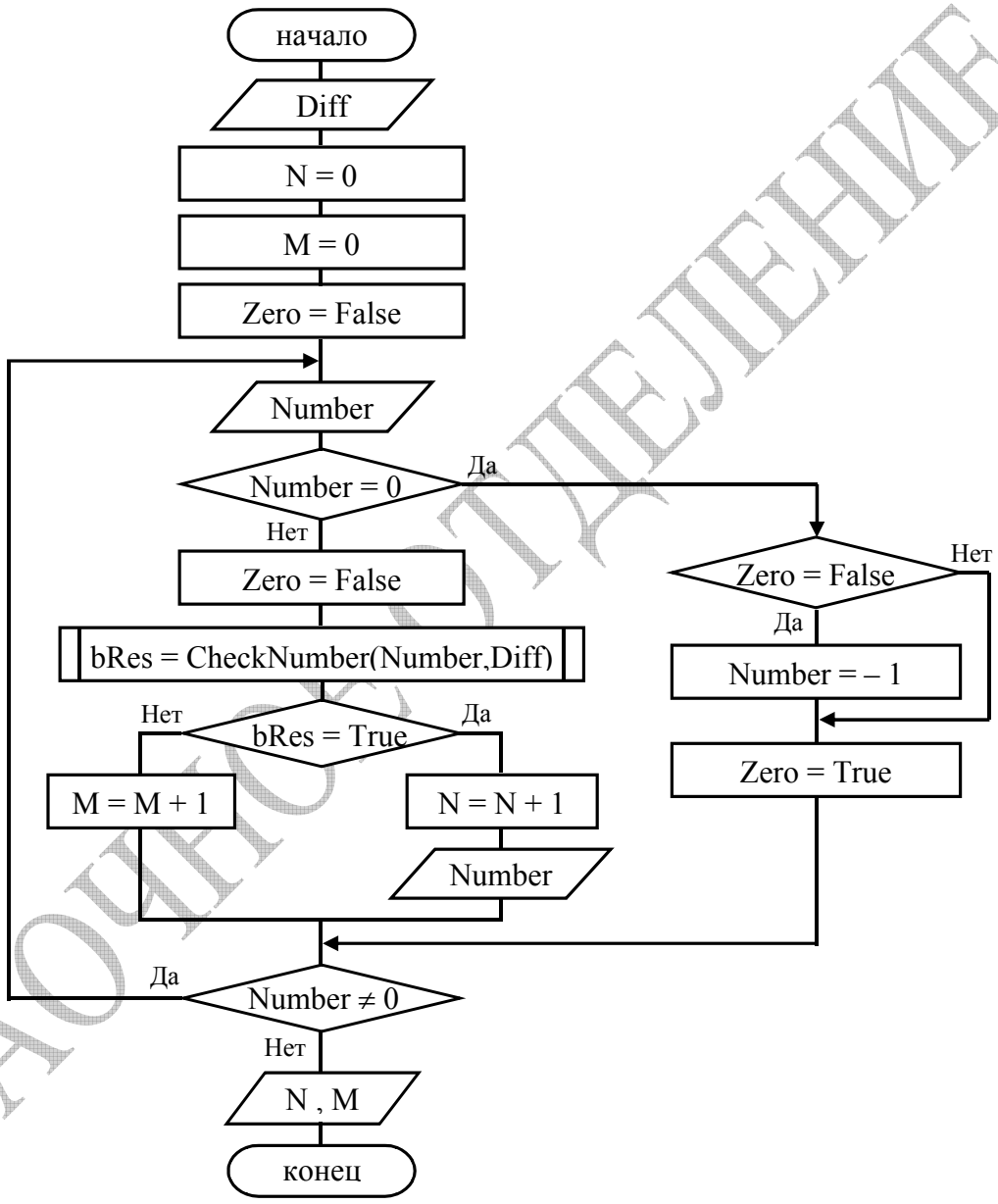
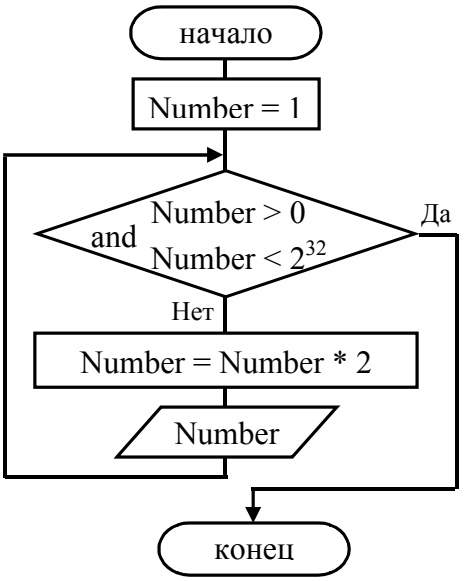


Рисунок 2.6 – Алгоритм InputNumbers

Алгоритм PrintDegreeTwo печатает все числа степени двойки, используя цикл от 1 до предельного значения используемого типа данных (long). В цикле он умножает значение на два. Цикл выполняется, пока текущее вычисленное значение меньше предела и является положительным.

Алгоритм PrintNumbers запрашивает нижнее и верхнее значение диапазона чисел и разность между цифрами. Алгоритм организует цикл от нижнего значения до верхнего значения и использует в цикле алгоритмы IsDegreeTwo, IsDifference и SumDigits, описанные выше, для проверки, является ли сумма цифр очередного числа числом степени двойки при заданной разности между соседними цифрами.

Алгоритм PrintDegreeTwo



Алгоритм PrintNumbers

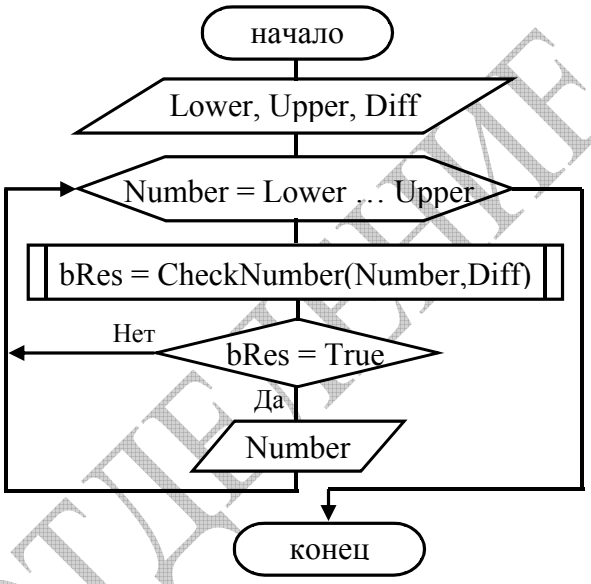


Рисунок 2.7 - Алгоритмы PrintDegreeTwo и PrintNumbers

Листинг программы:

```
/*****************************************************************
        *              Содержание модуля TASK.H                 *
*****************************************************************/
        void InputNumbers();
        void PrintNumbers();
        void PrintDegreeTwo();
        long SumDigits(long number);
        bool IsDegreeTwo(long number);
        bool IsDifference(long number, __int8 k);
        bool CheckNumber(long number, __int8 k);

/*****************************************************************
        *              Содержание модуля SOLVE.CPP              *
*****************************************************************/
        /*  Функция проверяет, обладает ли число заданными свойствами
        *.
        bool CheckNumber(long number, __int8 k)
        {
         return (IsDifference(number,k) == true &&
```

```
        IsDegreeTwo(SumDigits(number)) == true);
}

/*  Функция проверяет, является ли число числом степени двойки
    Возвращает true – если число степени двойки или false – если нет
*/
bool IsDegreeTwo(long number)
{
 long n;
 if(number < 0) number = -number;

 for(n = 1 ; n <= number ; n *= 2)
   if(n == number) return true;

 return false;
}
/*  Функция проверяет разность между цифрами числами
    Возвращает true – если разность между цифрами равна K
    или false – если хотя бы для одной пары цифр разность неравна K
*/
bool IsDifference(long number, __int8 k)
{
 __int8 digit, delta;

 if(number < 0) number = -number;
 digit = (__int8)(number % 10);
 number /= 10;

 while(number != 0)
 {
   delta = (__int8)(number % 10 - digit);
   if(delta < 0) delta = -delta;
        if(delta != k) return false;

        digit = (__int8)(number % 10);
        number /= 10;
 }
 return true;
}

/*   Функция вычисляет сумму цифр числа
*/
long SumDigits(long number)
{
 long SumDigits = 0;

 if(number < 0) number = -number;
```

```
  while(number != 0)
  {
        SumDigits += number % 10;
        number /= 10;
  }
  return SumDigits;
}

/*********************************************************************
 *              Содержание модуля INTERFACE.CPP                      *
*********************************************************************/
#include <iostream>
#include <limits.h>

using namespace std;

/*   Функция печатает числа степени двойки
*/
void PrintDegreeTwo()
{
  long number;

  cout << "\nSeries of degree two:\n";
  for(number=1 ; number < LONG_MAX && number > 0 ; number++)
    cout << number << "\n";
}


/*   Функция печатает в заданном диапазоне те числа,
     сумма цифр которых является числом степени двойки
     и разность между соседними цифрами числа равна k
*/
void PrintNumbers()
{
  int diff;                  // Difference between digits
  long lower, upper;           // Lower and Upper of the bound
  long number;               // Current number

  cout << "Entry the lower bound of numbers: ";
  cin >> lower;

  if(cin.fail())
  {
    cin.clear();
    cout << "Incorrect lower bound value";
        return;
```

```
}

cout << "Entry the upper bound of numbers: ";
cin >> upper;

if(cin.fail())
{
  cin.clear();
  cout << "Incorrect upper bound value";
  return;
}

cout << "Entry the digit difference in range [1...8]: ";
cin >> diff;

if(cin.fail())
{
  cin.clear();
  diff = -1;
}
if(diff < 1 || diff > 8)
{
  cout << "Incorrect digit difference value";
  return;
}

if(lower > upper)
{
  number = lower;
  lower = upper;
  upper = number;
}
cout << "List of numbers:";

for(number=lower ; number <= upper ; number++)

  if(CheckNumber(number,diff) == true)
    cout << "\n" << number;
}

/*  Функция вводит числа до тех пор, пока не будут введены
    Два нуля подряд, и проверяет каждое число, чтобы сумма
    цифр была числом степени двойки и разность между соседними
    цифрами числа равна k, и подсчитывает их.
*/
void InputNumbers()
{
```

```
bool zero = false;         // Flag to be finished
int n = 0, m = 0;          // Counters of numbers
int diff;                  // Difference between digits
long number;               // Entered number

cout << "Entry the digit difference in range [1...8]: ";
cin >> diff;

if(cin.fail())
{
  cin.clear();
  diff = -1;
}
if(diff < 1 || diff > 8)
{
  cout << "Incorrect digit difference value";
  return;
}

do
{
  cin.sync();
  cin >> number;

  if(cin.fail())
  {
    cin.clear();
    cout << “Incorrect number value”;
  }
  else if(number == 0)
  {
    if(zero == false) number = -1;
    zero = true;
  }
  else
  {
    zero = false;
    if(CheckNumber(number,diff) == true)
    {
      n++;
      cout << "(" << number << ")\t is (" << n << ")"
           << "\tsum=" << SumDigits(number) << "\n";
    }
    else m++;
  }
}
while(number != 0);
```

```
  cout << endl << "Numbers having properties: " << n
       << endl << "Other numbers: " << m << endl;
}

/*   Главная функция. Реализует интерфейс с пользователем.
*/
void main(void)
{
  char ch = '0';

  do
  {
    cout << "\nPress [N] to count the numbers";
    cout << "\nPress [P] to print the numbers in the bound";
    cout << "\nPress [D] to print the numbers of degree two";
    cout << "\nPress [Q] to quit the program\n?>";
    cin.get(ch);

    if((ch == 'D') || (ch == 'd'))
    {
      PrintDegreeTwo();
    }
    else if((ch == 'P') || (ch == 'p'))
    {
      PrintNumbers();
    }

    else if((ch == 'N') || (ch == 'n'))
    {
      CheckNumbers();
    }
    cin.sync();
  }
  while((ch != 'Q') && (ch != 'q'));
}
```

4 КОНТРОЛЬНАЯ РАБОТА № 3

4.1 Теоретическая часть

Цель:

Изучение приемов работы с одномерными статическими и динамическими массивами, способов передачи параметров и массивов в функции, конструкций множественного выбора. Изучение работы с файлами. Приобретение навыков декомпозиции задач, модульного проектирования программ, разработки циклических алгоритмов, алгоритмов ветвления, составления тест-планов и написания программной документации.

Задание:

Разработать программу, состоящую из нескольких модулей, которая в массиве чисел из Size элементов находит такую непрерывную последовательность элементов (1<M<Size–1), которая обладает свойством: согласно вариантам, и выводит индекс начального и индекс конечного элемента найденной последовательности. Программа должна предоставлять пользователю опциональный выбор критерия поиска, консольный ввод массива, отображение текущего массива, загрузку массива из файла, сохранение массива в файле, создание тествых массивов, и диагностику ошибок во всех функциях программы.

Требования:

Интерфейс программы в виде консольного меню реализуется в главной функции Main(), которая находится в модуле Lab3.cpp. Консольный ввод и вывод массива реализуется отдельными функциями соответственно InputArray и ShowArray, которые находятся в модуле InOut.cpp. Файловый ввод и вывод массива реализуется отдельными функциями соответственно LoadArray и SaveArray, которые находятся в модуле File.cpp. Поиск последовательности с заданным свойством реализуется соответственно функциями Search и CalcCriteria, которые находятся в модуле Solve.cpp. Найденные индексы возвращаются через аргументы функции. Тестовые примеры реализуется в виде статических массивов и копируются в рабочий массив функцией TestArray, которая также находятся в модуле Solve.cpp. Программа должна идентифицировать ошибки при вводе данных, при загрузке данных из файла, при сохранении данных в файл, и предоставлять диагностику ошибок в виде текстовых сообщений в консоли, например функция Error.

Интерфейс должен предоставлять пользователю возможности задания размера массива, ввода элементов массива, отображения текущего массива, результата поиска последовательности, загрузку из указанного файла, сохранение в указанном файле, выбор тестового примера, выбор опций поиска.

Комментарии:

Воспользоваться сайтами www.google.ru, www.wikipedia.org, www.yandex.ru для поиска информации. Подготовить тестовые массивы, сформулировать критерии поиска, разработать алгоритм для перебора всех последователь-ностей элементов в массиве, разработать алгоритм для расчета критерия по последовательности, ознакомиться с ситуациями, возникающими при работе с файлами, разработать алгоритмы консольного и файлового ввода-вывода массивов чисел. Где не указано явно, подразумеваются вещественные числа с плавающей точкой.

Оформление:

1. Текст задания
2. Математическую модель
3. План работы с расчасовкой
4. Техническое задание
5. Декомпозицию задач
6. Спецификацию требований к задачам
7. Спецификацию архитектуры модулей
8. Спецификацию формата файла
9. Тестовые планы
10. Блок-схемы и описание алгоритмов
11. API Reference реализованных функций
12. Листинги всех модулей программы с комментариями
13. Заметки о версии
14. Руководство пользователя

4.2 Задание для выполнения контрольной работы №3

1. Непрерывная последовательность чисел имеет либо минимальное, либо максимальное (*должно выбираться опционально*) произведение разностей между соседними элементами.

2. Непрерывная последовательность чисел имеет либо минимальную, либо максимальную (*должно выбираться опционально*) сумму разностей между соседними элементами.

3. Непрерывная последовательность чисел является пропорциональной либо на увеличение, либо на уменьшение (*должно выбираться опционально*).

Последовательность пропорциональна, если следующий элемент получается из предыдущего умножением на один и тот же коэффициент, например:

2,1 → 4,2 → 8,4 → 16,8 → 33,6 (*2)

4. Непрерывная последовательность чисел имеет либо минимальное, либо максимальное (*должно выбираться опционально*) среднеарифметическое значение элементов по абсолютному значению.

5. Непрерывная последовательность чисел имеет либо минимальное, либо максимальное (*должно выбираться опционально*) среднеквадратичное отклонение.

6. Непрерывная последовательность чисел либо имеет максимальную сумму и является упорядоченной по возрастанию, либо имеет минимальную сумму и является упорядоченной по убыванию (*должно выбираться опционально*).

7. Непрерывная последовательность чисел состоит из триморфных чисел и является упорядоченной либо по возрастанию, либо по убыванию (*должно выбираться опционально*).

8. Непрерывная последовательность целых чисел является частью последовательности Фибоначчи либо по возрастанию, либо по убыванию (*должно выбираться опционально*).

9. Непрерывная последовательность целых чисел является частью последовательности Софи Жермен либо по возрастанию, либо по убыванию (*должно выбираться опционально*).

10. Непрерывная последовательность чисел имеет либо минимальную, либо максимальную (*должно выбираться опционально*) сумму разностей между чётными и нечётными элементами.

4.3 Пример выполнения контрольной работы № 3

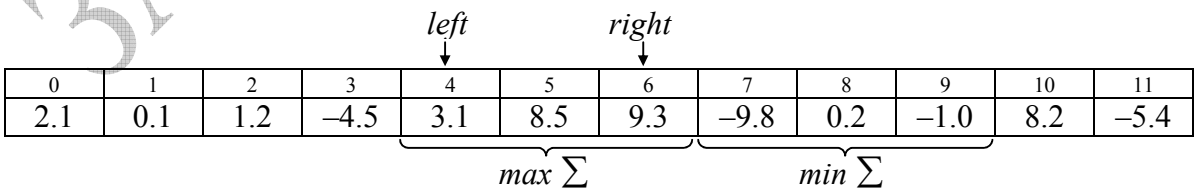
Задание:

Непрерывная последовательность чисел имеет либо минимальную, либо максимальную (*должно выбираться опционально*) сумму элементов.

Математическая модель:

Для решения задачи предлагается использовать метод перебора всех возможных непрерывных последовательностей элементов массива длины 1<M<Size–1 с вычислением суммы каждой последовательности, которое сравнивается с критериальным значением суммы.

Пусть имеется массив:



Для перебора последовательностей необходимо завести два индекса Left и Right. Индекс Left обозначает начало очередной последовательности и сдвигается все время влево к концу массива. Индекс Right обозначает конец

последовательности и сдвигается от индекса (Left+1) влево не более длины последовательности M. Длины последовательностей перебираются, начиная от 2 до (Size – Left). Для каждой последовательности вычисляется критерий как $K = \sum_{j=left}^{j=right} A_j$.

План работы:

Таблица 3.1 - План работы

№	Вид работы	Время, ч
1	Изучение предметной области, анализ интернет-публикаций, составление технического задания	8
2	Определение функциональности, декомпозиция задач и проектирование архитектуры модулей	8
3	Разработка и написание спецификаций (SRS, AMS, FFS)	6
4	Разработка алгоритмов решения задач, составление блок-схем	20
5	Анализ ситуаций, подбор входных данных, разработка тестовых планов	16
6	Кодирование алгоритмов, написание C++ кода	24
7	Тестирование программы в соответствии с тестовыми планами, отладка модулей, исправление ошибок	10
8	Документирование: API reference, руководство пользователя, комментарии к коду, оформление отчёта	10

Техническое задание:

Разработать программу поиска в массиве вещественных чисел непрерывной последовательности, удовлетворяющей критерию: максимум или минимум суммы элементов.

Требования к функциональности:

– консольный ввод-вывод массива чисел,
– файловый ввод-вывод массива чисел,
– обработка массива чисел для поиска последовательности,
– диагностика ошибок.

Требования к интерфейсу:

– диалоговое консольное меню ко всем функциям,
– наличие подсказок при при вводе данных и выборе команд,
– запрос подтверждений от пользователя в неоднозначных ситуациях,
– вывод понятной информации об ошибках.

Требования к тестированию:

– все функции программы должны иметь тестовые планы,
– программа должна хранить готовые тестовые массивы,
– отработать тестовые случаи при работе с файлами.

Требования к форматам хранения данных:

– разработать формат хранения массива в файле,
– составить спецификацию хранения данных.

Требования к архитектуре:

– программа должна строиться из четырёх модулей, каждый из которых отвечает за опередеенную функциональность программы,

– составить спецификацию на содержание и назначение модулей.

Требования к комплектности и поставке:

– программа должна поставляться в виде zip-архива,

– в поставку должны входить тестовые файлы,

– в поставку должно входить руководство пользователя.

Требования к техническим и программным средствам:

– PC Pentium, не менее 128M ОЗУ,

– ОС Microsoft Windows XP SP1/SP2/SP3/

Требования к средствам разработки:

– Microsoft Visual Studio 2005/2008/2010.

Декомпозиция задач:

Исходя из условий задания, главную задачу Main можно разбить на четыре основные задачи:

Console I/O – Организация консольного ввода-вывода массива,

File I/O – Организация файлового ввода-вывода массива,

Solve – Обработка массива в соответствии с заданием,

Interface – Организация интерфейса с пользователем.

Задачу Console I/O можно разбить на три независимые подзадачи:

ShowArray – вывод массива на консоль,

InputArray – вывод массива с консоли,

Error – вывод информации об ошибках.

Задача InputArray дополнительно требует решения задачи диагностики ошибок при вводе размера и элементов массива (InputChecking)

Задачу File I/O можно разбить на две независимые подзадачи:

SaveArray – сохранение массива в файле,

LoadArray – загрузка массива из файла.

Задача LoadArray дополнительно требует решения задачи диагностики ошибок при чтении размера и элементов массива из файла (FileChecking). Обе задачи дополнительно требуют решения задачи проверки корректности имени файла (CheckName), задачи проверки существования файла (CheckExist) и доступности файла для чтения-записи (CheckAccess).

Задачу Solve можно разбить на две независимые подзадачи:

Search – поиск непрерывной последовательности элементов массива, которая имеет минимальную или максимальную сумму элементов,

TestArray – создание тестового массива.

Задача Search дополнительно требует решения задачи вычисления суммы последовательности элементов массива (Criteria).

Задача Interface можно разбить на подзадачу разработки консольного меню (MainMenu) и подзадачу обращений к пользователю по ситуации. В рамках данной работы выделяется одна ситуация – необходимость перезаписи файла при сохранении массива (IsRewrite).

Дерево декомпозиции задач следующее:

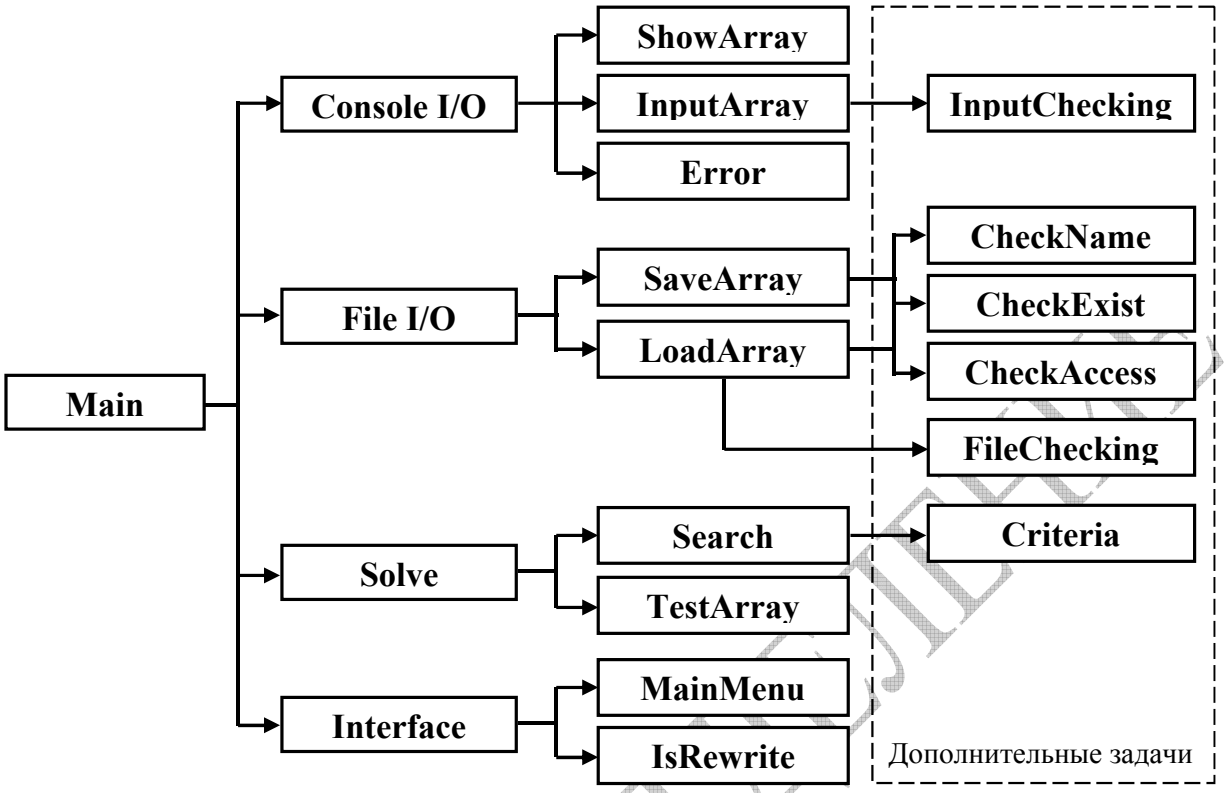


Рисунок 3.1 - Дерево декомпозиции задач

Таблица 3.2 - Спецификация требований к задачам (System Requirement Specification)

Требования		Задачи	
1	Консольный ввод-вывод	1	Вывод массива чисел на экран
		2	Ввод чисел с консоли и сохранение их в массиве
		3	Диагностика ошибок при вводе размера массива и чисел
		4	Вывод информации об ошибках пользователя
2	Файловый ввод-вывод	1	Сохранение массива в файле
		2	Загрузка массива из файла
		3	Диагностика ошибок работы файлом при чтении-записи данных
		4	Проверка корректности имени файла
		5	Обработка ситуаций существования файла
3	Обработка массива	1	Вычисление критерия
		2	Поиск последовательности, удовлетво-ряющей критерию
		3	Загрузка тестового массива
4	Интерфейс пользователя	1	Разработка интерактивного консольного меню для выбора всех команд

Таблица 3.2 - Спецификация архитектуры модулей (Architectural Module Specification)

Модуль	Тип	Функция	Назначение	SRS
INOUT	CPP	ShowArray	Вывод массива на консоль. Массив передается через аргументы.	1.1
		InputArray	Ввод размера массива, резервирование масива, ввод чисел и сохранение их в массиве с проверкой ошибок при вводе. Новый массив передается через аргументы.	1.2 1.3
		Error	Вывод информации об ошибке на консоль.	1.4
FILE	CPP	SaveArray	Сохранение массива в файле с диагностикой возникающих ошибок. Массив передается через аргументы.	2.1 2.3
		LoadArray	Загрузка массива из файла с диагностикой возникающих ошибок. Корректно загруженный массив передается через аргументы.	2.2 2.3
		CheckFileName	Проверка корректности имени файла.	2.4
SOLVE	CPP	Search	Поиск последовательности чисел, удовлетворяющей критерию.	3.2
		Criteria	Рассчет суммы чисел от одного индекса до другого.	3.1
		TestArray	Копирование тестового массива на место текущего массива	3.3
LAB3	CPP	MainMenu	Организация диалога с пользователем в виде консольного меню.	4.1
		IsRewrite	Запрос подтверждения на перезапись файла.	2.5
		Main	Главная функция	
TASK	H	enum errCode	Коды ошибок	
		Прототипы всех функций		

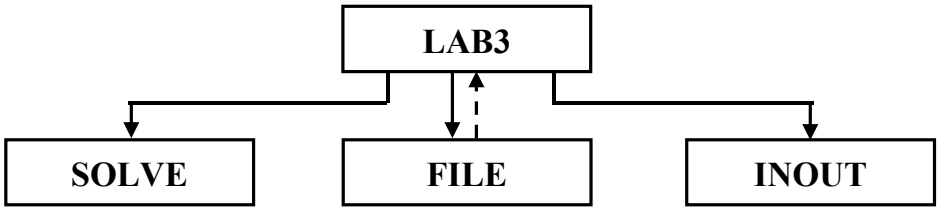


Рисунок 3.3 - Диаграмма взаимосвязи модулей

Спецификация формата файла (File Format Specification)

Назначение:

Разработанный формат файла предназначен для хранения массива вещественных чисел, обрабатываемых программой LAB3.

Основные требования:

Программа использует текстовые файлы формата ASCII. Файл содержит последовательность чисел, разделенных символами пробела, табуляции или переноса строки. При записи чисел допускаются цифры 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 и знаки +, –, точка. Точка разделяет целую и дробную части числа. В файле не допускаются иные символы, кроме указанных.

Внутренняя структура:

Первое число в файле хранит количество чисел в массиве. Количество последущих чисел должно быть не меньше указанного количества. Каждое число отделяется хотя бы одним пробельным символом. Первое число присутствует обязательно и не может иметь точку или знак минус.

$$N \quad Element_1 \quad Element_2 \quad \ldots \quad Element_i \quad \ldots \quad Element_N$$

где N – количество чисел – размер массива

$Element_i$ – числа – элементы массива

Пример файла:

4

1.25 3.45 –0.56 10

Таблица 3.3 – Диагностика ошибок формата файла

Код ошибки	Описание ошибки
errArraySize	Если первое число – размер массива – указано не правильно. Например, если использован алфавитный символ, число отрицательно, или выходит за допустимый предел.
errArrayElement	Если число – элемент массива – указано не правильно. Например, если использован алфавитный символ.
errAbsentElements	Если количество чисел в последовательности меньше, чем размер массива, указанный первым числом.
errCanNotSaveValue	Если значение не может быть сохранено в файле из-за системных ошибок работы с файлом.

Таблица 3.4 – Диагностика ошибок работы с файлом

Код ошибки	Описание ошибки
errFileName	Если имя файла записано некорректно.
errFileExists	Если файл существует с указанным именем, а должен отсутствовать.
errFileNotFound	Если файл не существует с указанным именем, а должен присутствовать.
errCanNotOpenFile	Если файл не может быть открыт.
errFileIsReadOnly	Если файл предназначен только для чтения.
errFileIsWriteOnly	Если файл предназначен только для записи.

Блок-схемы и описание алгоритмов:

Алгоритм CalcCriteria рассчитывает сумму последовательности и получает ссылку на массив, в котором находится последовательность, индекс начала и индекс конца последовательности как исходные данные. Алгоритм обходит все элементы последовательности и суммирует их в переменную sum. Алгоритм обнуляет сумму перед вычислениями. Алгоритм возвращает вычисленное значение суммы.

Алгоритм CalcCriteria

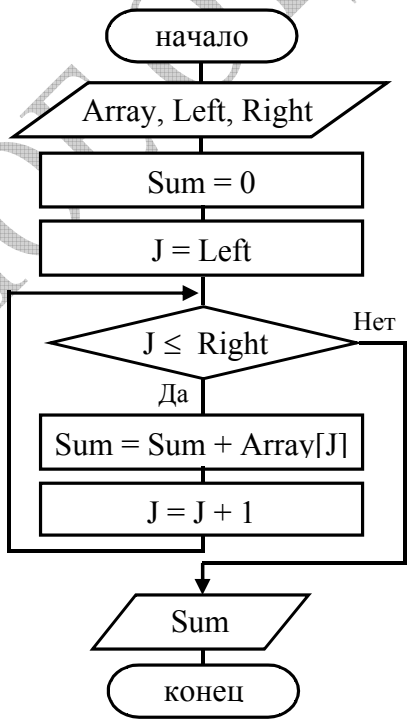


Рисунок 3.4 - Алгоритм CalcCriteria

Алгоритм Search осуществляет поиск массиве чисел такой непрерывной последовательности элементов, которая имеет максимальную

или минималь-ную сумму. Алгоритм получает ссылку на массив, размер массива и опцию поиска (True – максимум, False – минимум) как исходные данные. Алгоритм организует перебор последовательностей длины от 2 до (Size – 1). Для этого он использует счётчик I, который проходит элементы от первого до предпослед-него. Для каждого значения счётчика I алгоритм перебирает для него возможные длины последовательностей с помощью счётчика M. Для I-ого положения можно перебрать длины от 2 до (Size – I), так как последова-тельности больших длин уже будут перебраны на предыдущих шагах. Для каждой последовательности длины M, начинающейся от элемента I, алгоритм вычисляет конечный элемент как (I + M – 1) и использует алгоритм вычисления критерия CalcCriteria, описанный выше, чтобы рассчитать сумму.

Полученное значение суммы алгоритм Search сравнивает со значением критерия. В зависимости от значения опции поиска, чтобы запомнить границы последовательности, полученное значение суммы либо должно быть больше значения критерия при поиске максимума, либо должно быть меньше значения критерия при поиске максимума. Если условие критерия выполняется, то алгоритм запоминает начало последовательности как I, конец последовательности как (I + M – 1), и значение суммы как новое значение критерия.

Перед началом вычислений алгоритм Search инициализирует критериаль-ное значение суммы Criteria либо минимально возможным значением для используемого типа данных (double) при поиске максимума суммы, либо максимально возможным значением для используемого типа данных (double) при поиске минимума суммы в зависимости от опции поиска. Так как алгоритм использует предельное значение типа данных в качестве стартового значения критерия, то при первом же вычислении критерия в циклах алгоритма, он получит значение соответственно либо большее предельно минимального, либо меньшее предельно максимального, и первое сравнение полученного значения со стартовым значением критерия приведёт к корректировке значения критерия. Это позволяет избежать контроля номеров итераций для установки значения критерия.

Алгоритм возвращает начало последовательности и конец последователь-ности. Поскольку алгоритм возвращает два значения, то при практической реализации рекомендуется использовать передачу аргументов по ссылке.

Алгоритм ShowArray получает ссылку на массив и размер массива как исходные данные. Алгоритм последовательно выводит все элементы массива на консоль.

Алгоритм InputArray получает ссылку на массив, который должен быть перезаписан, и ссылку на переменную, хранящую размер массива как исходные данные. Алгоритм запрашивает размер массива, резервирует память под массив, и последовательно вводит элементы массива с консоли. алгоритм проверяет вводимые данные. Так, размер массива не может быть

отрицательным числом и должен быть в диапазоне от 1 до 1000 элементов. Чтобы не испортить текущий массив, находящийся в главной функции, алгоритм InputArray сначала запоминает ссылку и размер нового массива во временных переменных. Если в процессе ввода произойдут ошибки, то алгоритм освободит зарезервированную память, не изменяя массив в главной программе. Если ввод нового массива будет успешным, то алгоритм освободит память текущего массива, используя ссылку на него, и сделает новый массив текущим.

Алгоритм Search

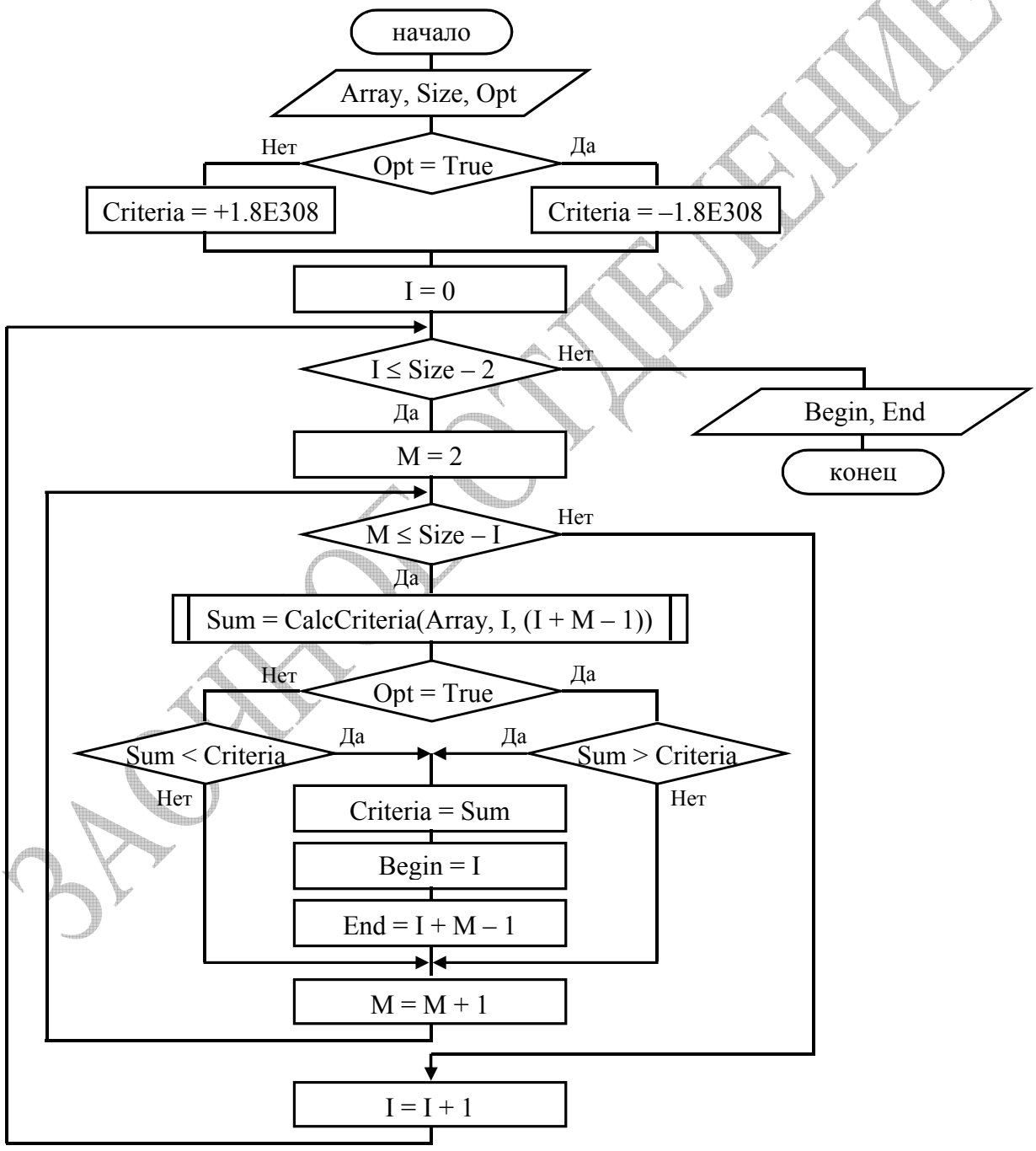


Рисунок 3.5 - Алгоритм Search

Алгоритм SaveArray получает ссылку на массив, размер массива, имя файла для сохранения, и признак перезаписи как исходные данные. Сначала алгоритм осуществляет проверку ситуаций работы с файлами. Если имя файла задано неправильно, то алгоритм возвращает ошибку errFileName. Если файл существует, но имеет атрибут Read-Only, то алгоритм возвращает ошибку errFileIsReadOnly. Если файл существует, то алгоритм использует признак перезаписи для выбора действий. Если признак перезаписи False, т.е. перезапись запрещена, то алгоритм возвращает ошибку errFileExists. Если признак перезаписи True, т.е. перезапись разрешена, о алгоритм запрашивает подтверждение на выполнение перезаписи, используя функцию IsRewrite, определенную в интерфейсе. Если файл существует, но не может быть открыт для записи, например, файл используется другим приложением, то алгоритм возвращает ошибку errCanNotOpenFile. Если ошибок работы с файлом не произошло, то алгоритм последовательно записывает все элементы массива в файл.

Алгоритм LoadArray получает ссылку на массив, который должен быть перезаписан, и ссылку на переменную, хранящую размер массива, и имя файла для загрузки как исходные данные. Алгоритм работает также как InputArray, но дополни-тельно осуществляет проверку ситуаций работы с файлами. Если имя файла задано неправильно, то алгоритм возвращает ошибку errFileName. Если файл не существует или путь к файлу неправильный, то алгоритм возвращает ошибку errFileNotFound. Если файл существует, но не может быть открыт для чтения, то алгоритм возвращает ошибку errCanNotOpenFile. При чтении данных алгоритм проверяет корректность размера массива, соответствие размера массива и количества элементов в файле, правильность задания чисел и соответствие спецификации формата.

Алгоритм TestArray использует статические массивы для хранения тестовых массивов. Алгоритм освобождает память из под текущего массива, резервирует память под тестовый массив, копирует элементы из статического массива, и делает тестовый массив текущим. Чтобы определить размер статического массива, он использует операцию sizeof.

Алгоритм CheckFileName получает имя файла как исходные данные и проверяет его на наличие недопустимых символов <, >, |, *, ?, “, символ двоеточия может быть только на вторым символом в строке, имя не должно содержать двух и более обратных подряд, и т.д.

Тестовые планы:

Тестирование программы предлагается разбить на четыре части:

1. Тестирование поиска последовательности чисел в массиве

В качестве тестового массива использовать следующие данные:

2.1	0.1	1.2	–4.5	3.1	8.5	9.3	–9.8	0.2	–1.0	8.2	–5.4

Результат поиска последовательности с максимальной суммой:
Left Index = 4
Right Index = 6
Результат поиска последовательности с минимальной суммой:
Left Index = 7
Right Index = 9

2. Тестирование ввода массива с консоли

Для тестирования ввода размера массива необходимо использовать следующие данные:

Таблица 3.5 – Данные для тестирования ввода размера массива

№	Размер	Ожидаемый результат
1	A1	Вывод ошибки: Incorrect array size
2	0	Вывод ошибки: Incorrect array size
3	–10	Вывод ошибки: Incorrect array size
4	пробелы	Повтор ввода
5	5	Правильный ввод. Резервирование трёх элементов.

Для тестирования ввода элемента массива необходимо использовать следующие данные:

Таблица 3.6 - Данные для тестирования ввода элемента массива

№	Элемент	Ожидаемый результат
1	A1	Вывод ошибки: Incorrect value
2	пробелы	Повтор ввода
3	3.2	Ввод числа 3.2
4	–4.89	Ввод числа –4.89
5	1.2.3	Ввод числа 1.2
6	00091	Ввод числа 91
7	1.20000	Ввод числа 1.2

Для тестирования функций ввода, отображения и создания тестового массива выполнить следующие действия:

1) Вызвать функцию ввода массива и ввести пять элементов:

00004	–.1	9.4	3.1.5	1.2000

2) Вызвать функцию отображения массива. Результат на экране:
4 –0.1 9.4 3.1 1.2

3) Вызвать функцию ввода массива и ввести три элемента:

14	35	–1.4

4) Вызвать функцию отображения массива. Результат на экране:
14 35 –1.4

5) Вызвать функцию ввода массива и ввести некорректные данные.

6) Вызвать функцию отображения массива. Результат на экране:

14 35 –1.4

7) Вызвать функцию создания тестового массива.

6) Вызвать функцию отображения массива. Результат на экране:

2.1 0.1 1.2 –4.5 3.1 8.5 9.3 –9.8 0.2 –1.0 8.2 –5.4

3. Тестирование загрузки массива из файла

Создать тестовые файлов со следующим содержанием:

Файл	Содержание	Ожидаемый результат
1.txt	u3 1.1 –2.3 4.0	Вывод ошибки: Incorrect array size
2.txt	0	Вывод ошибки: Incorrect array size
3.txt	–10 1.1 4.0	Вывод ошибки: Incorrect array size
4.txt	3 1.A 1/0 e4	Вывод ошибки: Incorrect element value
5.txt	3 1.1 –2.3	Вывод ошибки: There are absent elements
6.txt	3 2.1 –1.4 6.0	Правильный ввод: Array is loaded successfully

Для тестирования загрузки вместе с сохранением выполнить следующие действия:

1) Вызвать функцию создания тестового массива.

2) Вызвать функцию сохранения массива в файле. Ввести имя 0.txt.

Вывод сообщения: Array is saved successfully

3) Открыть файл 0.txt в программе «Блокнот». Результат в файле:

12 2.1 0.1 1.2 –4.5 3.1 8.5 9.3 –9.8 0.2 –1.0 8.2 –5.4

4) Вызвать функцию загрузки массива из файла. Ввести имя 6.txt.

5) Вызвать функцию отображения массива. Результат на экране:

2.1 –1.4 6.0

6) Вызвать функцию загрузки массива из файла. Ввести имя 0.txt.

5) Вызвать функцию отображения массива. Результат на экране:

2.1 0.1 1.2 –4.5 3.1 8.5 9.3 –9.8 0.2 –1.0 8.2 –5.4

4. Тестирование работы с файлами

Для тестирования функции проверки имени файла использовать следующие данные:

№	Имя файла	Ожидаемый результат
1	1?.txt	Вывод ошибки: Incorrect filename
2	0*1.txt	Вывод ошибки: Incorrect filename
3	0.t:x	Вывод ошибки: Incorrect filename
4	C:\\1.txt\\	Вывод ошибки: Incorrect filename
5	11\2.txt\\	Вывод ошибки: Incorrect filename
4	C:\0.txt	Корректное сохранение или загрузка файла

Для тестирования ситуаций при сохранении данных в файл выполнить следующие действия:

1) Скопировать файл 6.txt в файл 7.txt и в файл 8.txt.

2) В проводнике установить атрибут Read-Only для файла 7.txt.

3) Вызвать функцию создания тестового массива.
4) Вызвать функцию сохранения массива в файле. Ввести имя 7.txt.
Вывод ошибки: File is accessed for read-only
5) Вызвать функцию сохранения массива в файле. Ввести имя 8.txt.
Вывод сообщения: File already exists. Do you want to overwrite it [Y/N]?
6) Ввести [Y].
Вывод сообщения: Array is saved successfully
3) Открыть файл 8.txt в программе «Блокнот». Результат в файле:
12 2.1 0.1 1.2 –4.5 3.1 8.5 9.3 –9.8 0.2 –1.0 8.2 –5.4
5) Вызвать функцию сохранения массива в файле. Ввести имя 6.txt.
Вывод сообщения: File already exists. Do you want to overwrite it [Y/N]?
6) Ввести [N].
7) Открыть файл 6.txt в программе «Блокнот». Результат в файле:
3 2.1 –1.4 6.0
8) Открыть файл 6.txt в программе Word.
9) Вызвать функцию сохранения массива в файле. Ввести имя 6.txt.
Вывод сообщения: File can not be opened

Для тестирования ситуаций при загрузке данных из файла выполнить следующие действия:

1) Вызвать функцию загрузки массива из файла. Ввести имя 100.txt.
Вывод сообщения: File not found

API reference основных функций:

Прототип:
errCode **LoadArray**(double* &pArray, int &Size, char *Name)
Описание:
Функция загружает массив вещественных чисел из файла.
Аргументы:
pArray – ссылка к указателю на массив вещественных чисел,
Size – ссылка к переменной, которая хранит размер массива,
Filename – указатель на строку имени файла с путем.
Возвращаемое значение: Код:
errOK – загрузка была успешна, ошибок нет,
errFileName – некорректное имя файла,
errFileNotFound – файл не найден или неверный путь,
errCanNotOpenFile – невозможно открыть,
errArraySize – файл хранит некорректный размер массива,
errNotMemory – недостаточно памяти для резервирования массива,
errArrayElement – файл хранит некорректные элементы массива,
errAbsentElements – количество элементов массива меньше его размера

Прототип:
errCode **SaveArray**(double* pArray, int Size, char *Name, bool rw)

Описание:
Функция записывает массив вещественных чисел в файл.
Аргументы:
pArray – указатель на массив вещественных чисел,
Size – размер массива,
Name – указатель на строку имени файла с путем,
rw – режим перезаписи в случае существования файла:
true – запросить подтверждение на перезапись,
false – отменить сохранение с выдачей ошибки.
Возвращаемое значение: Код:
errOK – сохранение было успешено, ошибок нет,
errFileName – некорректное имя файла,
errFileExists – файл существует и перезапись отменена,
errFileIsReadOnly – файл существует и доступен только для чтения,
errCanNotOpenFile – невозможно открыть или создать файл.

Прототип:
void **ShowArray**(double * pArray, int Size)
Описание:
Функция выводит массив вещественных чисел на консоль.
Аргументы:
pArray – указатель на массив вещественных чисел,
Size – размер массива.
Возвращаемое значение: нет.

Прототип:
errCode **InputArray**(double* &pArray, int &Size)
Описание:
Функция вводит массив вещественных чисел с консоли.
Аргументы:
pArray – ссылка к указателю на массив вещественных чисел,
Size – ссылка к переменной, которая хранит размер массива.
Возвращаемое значение: Код:
errOK – ввод был успешен, ошибок нет,
errArraySize – был введен некорректный размер массива,
errNotMemory – недостаточно памяти для резервирования массива,
errArrayElement – элементы были введены некорректно.

Прототип:
Double **CalcCriteria**(double * pArray, int Left, int Right)
Описание:
Функция вычисляет сумму последовательности элементов массива.
Аргументы:
pArray – указатель на массив вещественных чисел,

Left – индекс начала последовательности,
Right – индекс конца последовательности.
Возвращаемое значение:
Сумма элементов последовательности.

Прототип:
void **Search**(double *Arr, int Size, bool Opt, int& Begin, int& End)
Описание:
Функция ищет в массиве вещественных чисел такую непрерывную последовательность элементов, которая имеет либо максимальную, либо минимальную сумму элементов.
Аргументы:
Arr – указатель на массив вещественных чисел,
Size – размер массива,
Opt – опция поиска:
true – максимум суммы,
false – минимум суммы.
Begin – ссылка на переменную, в которую помещается индекс начала последовательности,
End – ссылка на переменную, в которую помещается индекс конца последовательности.
Возвращаемое значение:
Функция возвращает индексы начала и конца последовательности через свои аргументы.

Прототип:
bool **CheckFileName**(char *name)
Описание:
Функция проверяет корректность имени файла.
Аргументы:
Filename – указатель на строку имени файла с путем.
Возвращаемое значение:
true – если имя файла корректно,
false – если имя файла некорректно.

Прототип:
errCode **TestArray**(double* &pArray, int &Size)
Описание:
Функция копирует тестовый массив.
Аргументы:
pArray – ссылка к указателю на массив вещественных чисел,
Size – ссылка к переменной, которая хранит размер массива.
Возвращаемое значение: всегда errOK.

Листинг программы:

```
/*****************************************************************
 *                Содержание модуля TASK.H
*****************************************************************/
/*  Список кодов ошибок  */
typedef enum {
        errOK = 0,
        errNotMemory = 1,
        errArraySize = 2,
        errArrayElement = 3,
        errAbsentElements = 4,
        errCanNotOpenFile = 5,
        errCanNotSaveValue = 6,
        errFileName = 7,
        errFileExists = 8,
        errFileNotFound = 9,
        errFileIsReadOnly = 10,
        errFileIsWriteOnly = 11
} errCode;
bool IsRewrite();
bool CheckFileName(char *name);
errCode Error(char *operate, errCode err);
errCode InputArray(double* &pArray, int &Size);
errCode TestArray(double* &pArray, int &Size);
errCode SaveArray(double* pArray, int Size, char *filename, bool rw);
errCode LoadArray(double* &pArray, int &Size, char *filename);
void ShowArray(double *arr, int size);
double CalcCriteria(double *arr, int left, int right);
void Search(double *arr, int size, bool opt, int& begin, int& end);

/*****************************************************************
 *                Содержание модуля SOLVE.CPP
*****************************************************************/
#include <float.h>
#include <stddef.h>
#include "task.h"
/*  Функция загружает тестовый массив в качестве текущего массива
*/
errCode TestArray(double* &pArray, int &Size)
{
 static double test[] =
 { 2.1,0.1,1.2,-4.5,3.1,8.5,9.3,-9.8,0.2,-1.0,8.2,-5.4 };
 if(pArray != NULL) delete[] pArray;
```

```
  Size = sizeof(test) / sizeof(double);
  pArray = new double[Size];
  for(int i=0 ; i < Size ; i++)
  pArray[i] = test[i];
  return errOK;
 }
 /*  Функция вычисляет сумму последовательности элементов
    в заданном массиве arr от индекса left до индекса right
 */
 double CalcCriteria(double *arr, int left, int right)
 {
  double sum = 0;
  for(int j=left ; j <= right ; j++)
    sum += arr[j];
  return sum;
 }
 /*  Функция ищет в массиве arr размерностью size такую непрерывную
    последовательность элементов, начинающуюся с индекса begin и
    заканчивающуюся индексом end, которая имеет максимальную при
    opt = true или минимальную при opt = false сумму элементов.
 */
 void Search(double *arr, int size, bool opt, int& begin, int& end)
 {
  int i,m;
  double sum;
  double criteria = ((opt) ? -DBL_MAX : DBL_MAX);
  for(i=0 ; i <= size-2 ; i++)
  {
    for(m=2 ; m <= size-i ; m++)
    {
      sum = CalcCriteria(arr, i, (i + m - 1));
      if((opt) ? sum > criteria : sum < criteria)
      {
        criteria = sum;
        begin = i;
        end = i + m - 1;
      }
    }
  }
 }
/*****************************************************************
       *              Содержание модуля INOUT.CPP              *
*****************************************************************/
 #include <iostream>
```

```
#include "task.h"
using namespace std;
/*  Функция выводит на консоль массив pArray размером Size
*/
void ShowArray(double *pArray, int Size)
{
 if(pArray != NULL || Size < 1)
 {
    cout << "Current array:" << endl;
    for(int i=0 ; i < Size ; i++)
       cout << pArray[i] << " ";
 }
 else cout << "Array is not entered";
}
/*  Функция вводит массив с консоли и диагностирует ошибки
   pArray – ссылка на текущий массив, Size – ссылка на его размер
*/
errCode InputArray(double* &pArray, int &Size)
{
 char ch = '\0';
 double* newArray = NULL;
 int idx, newSize = 0;
 do
 {
  cout << "Entry the array size: ";
  cin >> newSize;
  cin.sync();
  if(cin.fail())
  {
    cin.clear();
    newSize = 0;
  }
  if(newSize < 1 || newSize > 1000)
  {
    cout << "Invalid array size. [A]-Abort. [Enter]-Repeat.";
    cin.get(ch);
    if((ch == 'A') || (ch == 'a')) return errArraySize;
  }
  else ch = '\0';
 }
 while(ch != '\0');
 newArray = new double[newSize];
 if(newArray == NULL) return errNotMemory;
```

```
  cout << "Entry " << newSize << " elements of the array:" << endl;
  for(idx=0 ; idx < newSize ; idx++)
  {
   do
   {
    cout << "arr[" << idx << "] ?>";
    cin >> newArray[idx];
    cin.sync();

    if(cin.fail())
    {
      cin.clear();
      cout << "Invalid value. [A]-Abort. [Enter]-Repeat.";
      cin.get(ch);
      if((ch == 'A') || (ch == 'a'))
      {
        delete[] newArray;
        return errArrayElement;
      }
    }
    else ch = '\0';
   }
   while(ch != '\0');
  }

  if(pArray != NULL) delete[] pArray;

  pArray = newArray;
  Size = newSize;
  return errOK;
}

/*  Список описаний ошибок
*/
char *errDescription[] = {
        "",
        "Not enough of memory",
        "Incorrect array size",
        "Incorrect element value",
        "There are absent elements",
        "File can not be opened",
        "Value can not be saved",
        "Incorrect filename",
        "File already exists",
```

```
        "File not found",
        "File is accessed for read-only",
        "File is accessed for write-only"
    };

    /*  Функция выводит на консоль описание ошибки
    */
    errCode Error(char *Operate, errCode err)
    {
     if(err) cout << endl << Operate << " status: "
              << errDescription[err] << endl;
     return err;
    }
/*****************************************************************
 *              Содержание модуля FILE.CPP                       *
*****************************************************************/
    #include <fstream>
    #include <io.h>
    #include "task.h"

    using namespace std;

    /*  Функция проверяет корректность имени файла filename, и
       Возвращает true – если имя корректно, или false – если нет.
    */
    bool CheckFileName(char *name)
    {
     char ch;
     int i;

     for(i=0,ch=name[0] ; ch != '\0' ; i++,ch=name[i])
     {
      if(ch == ':' && i != 1) return false;
       if(ch == '\\' && (name[i+1] == '\0' || name[i+1] == '\\'))
         return false;
       if(ch == '<' || ch == '>' || ch == '*' || ch == '?' ||
          ch == '|' || ch == '\"') return false;
     }
     return true;
    }
    /*  Функция загружает массив из файла с именем filename и
       диагностирует ошибки. pArray – ссылка на текущий массив,
       который должен быть перезаписан, Size – ссылка на его размер
    */
```

```
errCode LoadArray(double* &pArray, int &Size, char *filename)
{
 double* tmpArray = NULL;
 int idx, tmpSize = 0;

 if(CheckFileName(filename) == false) return errFileName;

 if(_access(filename,0) != 0) return errFileNotFound;

 ifstream in(filename);
 if(!in) return errCanNotOpenFile;

 in >> tmpSize;
 if(!in) tmpSize = 0;

 if(tmpSize < 1 || tmpSize > 1000) return errArraySize;

 tmpArray = new double[tmpSize];

 if(tmpArray == NULL) return errNotMemory;

 for(idx=0 ; idx < tmpSize ; idx++)
 {
   in >> tmpArray[idx];
   if(!in)
   {
     delete[] tmpArray;
      return ((in.eof()) ? errAbsentElements : errArrayElement);
   }
 }

 if(pArray != NULL) delete[] pArray;

 pArray = tmpArray;
 Size = tmpSize;
 return errOK;
}

/*  Функция сохраняет массив pArray размеров Size в файле с
    именем filename в режиме перезаписи rw = true или отмены
    записи в случае его существования rw = false.
*/
errCode SaveArray(double *pArray, int Size, char *filename, bool rw)
{
```

```
  if(CheckFileName(filename) == false) return errFileName;

  if(_access(filename,0) == 0)
  {
    if(_access(filename,6) != 0) return errFileIsReadOnly;

    if(rw == false) return errFileExists;
    else if((rw = IsRewrite()) == false) return errFileExists;
  }

  ofstream out(filename);
  if(!out) return errCanNotOpenFile;

  out << Size << endl;
  if(!out) return errCanNotSaveValue;

  for(int idx=0 ; idx < Size ; idx++)
  {
    out << pArray[idx] << endl;
    if(!out) return errCanNotSaveValue;
  }
  return errOK;
}

/*****************************************************************
 *                Содержание модуля LAB3.CPP                     *
*****************************************************************/
  #include <iostream>
  #include "task.h"

  using namespace std;

  /*  Функция делает запрос на перезапись файла.
  */
  bool IsRewrite()
  {
   char ch;

   cout << endl
        << "File already exists. Do you want to overwrite it [Y/N]?";
   cin >> ch;

   return (ch == 'Y' || ch == 'y');
  }
```

```
/*   Функция реализует консольное меню и интерфейс с пользователем.
*/
void MainMenu(double* &wrkArray, int& wrkSize)
{
 char ch = ' ';
 char filename[66];

 bool bCriteria;
 int idxLeft, idxRight;

 do
 {
  cout << endl << "[I] Input a new array";
  cout << endl << "[T] Create the test array";
  cout << endl << "[S] Save the array in a file";
  cout << endl << "[L] Load the array from a file";
  cout << endl << "[D] Display the current array";
  cout << endl << "[R] Search the sequence for criteria";
  cout << endl << "[Q] Quit the program" << endl << "?>";
  cin >> ch;

  switch(ch)
  {
   case 'I':
   case 'i':
       InputArray(wrkArray, wrkSize);
       break;
   case 'T':
   case 't':
       TestArray(wrkArray, wrkSize);
   case 'D':
   case 'd':
       ShowArray(wrkArray, wrkSize);
       break;
   case 'S':
   case 's':
       cout << "Entry the filename:>";
       cin >> filename;
       if(Error("Saving", SaveArray(wrkArray, wrkSize,
                                    filename, true)) == errOK)
         cout << endl << "Array is saved successfully" << endl;
       break;
   case 'L':
```

```
    case 'l':
        cout << "Entry the filename to be loaded:>";
        cin >> filename;
        if(Error("Loading", LoadArray(wrkArray, wrkSize,
                                filename)) == errOK)
          cout << "Array is loaded successfully" << endl;
        break;
    case 'R':
    case 'r':
        if(wrkArray == NULL)
        {
          cout << "Array is not entered";
          break;
        }
        cout << "Select criteria:" << endl
            << "[I]-max(sum) , [D]-min(sum) ?>";
        cin >> ch;

        if(ch == 'I' || ch == 'i') bCriteria = true;
        else if(ch == 'D' || ch == 'd') bCriteria = false;
        else {  cout << "Incorrect option: " << ch;
                ch = ' ';
                break;
             }

        Search(wrkArray, wrkSize, bCriteria, idxLeft, idxRight);

        ShowArray(wrkArray, wrkSize);
        cout << endl << "Left index = " << idxLeft
             << endl << "Right index = " << idxRight;
        break;
    default:
        cout << "Incorrect menu item";
    }
    cin.sync();
  }
  while(ch != 'Q' && ch != 'q');
}

void main(void)                 //  Главная функция
{
  double* myArray = NULL;       //  Текущий рабочий массив
  int arrSize = 0;              //  Размер текущего массива
  MainMenu(myArray, arrSize);
```

}

Заметки о версии:

Программа Lab3 версии 1.0.0 предназначена для поиска в массиве вещест-венных чисел такой последовательности, которая имеет либо максимальную, либо минимальную сумму в зависимости от параметров поиска.

Программа имеет следующую функциональность:

– консольный ввод и отображение массива;

– сохранение массивов в файлах и загрузка массивов из файлов, имеющих формат, описанный в FFS;

– поиск в массиве последовательности чисел в соответствии с параметрами поиска;

– диагностику ошибок формата файла и действий пользователя.

В комплект поставки входят файлы:

Lab3.exe – программа для запуска,
Test.dat – файл с тестовым массивом в формате, описанном в FFS,
FFS.txt – описание формата файла данных,
Manual.htm – руководство пользователя,
Release.txt – заметки о версии,
Licence.txt – информация о производителе и соглашение об использовании.

Комплект поставляется в виде ZIP-архива. Специальной маркировки не требуется.

Руководство пользователя:

Руководство пользователя пишется студентом в свободной форме и оформляется в электронном виде как набор html-страничек с гиперссылками между ними и скриншотами программы в формате bmp, gif, или jpg.

Информация о производителе (Licence.txt) должна содержать сведения о студенте, выполнявшем работу (ФИО, группа, факультет, e-mail).

5 КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Какие виды работ по разработке программ вы знаете? В чём заключаются эти работы? В каком порядке они проводятся?
2. Какие стадии разработки программного обеспечения вы знаете? В чём они заключаются? Как составляется техническое задание?
3. Что такое функциональность? Что такое задача? Что такое декомпозиция задач? Как строится дерево задач? Как составляются SRS?
4. Что такое модуль? Какие свойства присущи модулю? Какие виды связности? В чём заключается технология модульного проектирования? Как составляются AMS?
5. Что такое алгоритм? Какими свойствами обладает алгоритм? Какие виды алгоритмов вы знаете? Какие способы записи алгоритмов вам известны?
6. Какой язык программирования вы использовали в выполнения контроль-ных работ и почему? Какие синтаксические конструкции языка програм-мирования вы применяли для реализации ваших алгоритмов?
7. Что такое блок-схема? Какие графические элементы вы знаете и что они обозначают? По каким правилам оформляются блок-схемы?
8. Что такое данные? Какие типы данных вы знаете? Какими свойствами и ограничениями они обладают? Какие типы данных вы использовали в ходе выполнения работ?
9. Что такое файл? Какие виды файлов вы знаете? Что такое формат файла и какие форматы вы знаете? Какие атрибуты могут иметь файлы? Какие ошибки могут возникать при работе с файлами? Как составляются FFS?
10. В чём заключается тестирование программы? Какие виды тестирования вы знаете? Какие методы тестирования вы применяли в ходе выполнения работ? Как составляются тестовые планы?
11. Из каких частей состоит интегрированная среда разработки программного обеспечения? Какие основные части в неё входят и для чего они предназ-начены? С какой средой вы работали? Расскажите о среде.
12. Что такое отладка программы? Какие виды программных ошибок вы знаете? Какие способы предупреждения ошибок вы знаете? Как вы произ-водили отладку программ в ходе выполнения работ?
13. Какие виды программной документации вы знаете? Какие вы знаете документы, какую информацию они содержат, и из каких разделов они состоят? Какие документы вы составляли в ходе выполнения работ?

ВИДЫ УЧЕБНЫХ ЗАНЯТИЙ И ИХ ОБЪЁМ

В результате изучения дисциплины студенты должны:

Знать:
- основные понятия и терминологию программиста,
- задачи и этапы проектирования программных продуктов,
- технологические средства программирования,
- типовые алгоритмические конструкции,
- типы данных и методы структурирования данных,
- способы представления программных структур.

Уметь:
- записывать алгоритмы стандартными способами,
- программировать на современном алгоритмическом языке,
- готовить техническую документацию на программные продукты,
- тестировать программные продукты.

Иметь опыт:
- составления алгоритмов,
- структурирования данных в предметной области,
- написания собственных программ,
- тестирования программ.

Иметь представление:
- о содержании работ по созданию программной продукции,
- о критериях оценки качества программных продуктов,
- об интерфейсах и диалогах.

Таблица 4.1 - Объем дисциплины и виды учебной работы

Вид учебной работы	Всего часов
Общая трудоемкость дисциплины	80
Аудиторные занятия	51
Лекции	34
Лабораторные работы (ЛР)	17
Самостоятельная работа	29
Вид итогового контроля (зачет, экзамен)	Зачет

Рабочей программой дисциплины предусмотрена самостоятельная работа студентов в объеме 29 часов. Самостоятельная работа проводится с целью углубления знаний по дисциплине и предусматривает:
- чтение студентами рекомендованной литературы;
- подготовку к сдаче зачёта.

Таблица 4.2 - Тематический план дисциплины включает:

№ п/п	Наименование темы	Всего часов	Аудиторные занятия			Самостоятельная работа
			Лекции	Лабора-тор-ные работы	Курсовое проекти-ро-вание	
1	Введение и терминология	1	1	-	-	-
2	Процесс проектирования программных продуктов	3	2	1	-	-
3	Алгоритмы, представле-ние алгоритма	6	2	2	-	2
4	Структуры данных	12	6	2	-	4
5	Основы программирования	18	6	4	-	8
6	Объектно-ориентирован-ное программирование	13	6	2	-	5
7	Средства программирования	12	6	2	-	4
8	Файлы, работа с ними	6	2	2	-	2
9	Тестирование программных продуктов	9	3	2	-	4
	ИТОГО	**80**	**34**	**17**	**-**	**29**

Итоговый контроль осуществляется в виде зачёта в конце семестра. Для допуска к зачёта необходимо выполнение трёх контрольных работ. При подготовке к зачёту рекомендуется сначала прочитать теоретический материал и дополнить его сведениями из литературы. При этом студент получает представление о предмете изучаемой дисциплины в целом. Подобная подготовка позволяет студенту продемонстрировать на зачете свою эрудицию.

ЛИТЕРАТУРА

1. Давыдов В.Г. Технологии программирования С++: Учебное пособие, рекомендовано УМО.- СПб: bhv-питер, 2005.- 672 с.
2. Иванова Г.С. Технологии программирования: Учебник для вузов. – М.: Издательство МГТУ им. Н.Э. Баумана, 2003.
3. Брукс Ф. Мифический человеко-месяц или как создаются программные системы. – СПб.: Символ-Плюс, 1999.
4. Липаев В.В. Управление разработкой программных комплексов. М.: Финансы и статистика, 1993.
5. Липаев В.В. Проектирование программных средств: Учебное пособие.- М.: Высшая школа, 1990.- 303 с.
6. Боэм Б. Инженерное проектирование программного обеспечения в действии – М.: Радио и связь, 1985.
7. Зиглер К. Методы проектирования программных систем. – М.: Мир, 1985.
8. Фокс Д. Программное обеспечение и его разработка. – М.: Мир, 1985.
9. Вирт Н. «Алгоритмы + структуры данных = программы».- М.: Мир, 1985.- 406 с.
10. Страуструп Б. Язык программирования С++: Специальное издание. / пер. с англ. - М.: «Бином-Пресс», 2006.- 1104 с.
11. Мэтью Уилсон Практический подход к решению проблем программирования С++. / пер. с англ.- М.: «Кудиц-Образ», 2006.- 736 с.
12. Стивен Прата Язык программирования Си. Лекции и упражнения. 5-е изд. / пер. с англ.- М.: изд. дом «Вильямс», 2006.- 960 с.
13. Дейтел Х.М., Дейтел П.Дж. Как программировать на Си. 4-е изд. / пер. с англ.- М.: «Бином-Пресс», 2006.- 912 с.
14. Романов Е.Л. Практикум по программированию на С++: Учебное пособие.- СПб: bhv-питер, 2004.- 432 с.
15. Трой Д. Программирование на языке СИ для персонального компьютера / пер. с англ. Кузьмина Б.А.- М.: Радио и Связь, 1991.- 432 с.
16. Керниган Б., Ритчи Д. Язык программирования СИ / пер. с англ. Штаркмана В.С.- М.: Финансы и Статистика, 1992.- 272 с.

Кафедра систем автоматизированного проектирования и управления

Методические указания к выполнению контрольных работ
для студентов заочной формы обучения
направления подготовки «Информатика и вычислительная техника»

Технологии программирования

Александр Юрьевич Рогов

Отпечатано с оригинал макета. Формат 60x90 $^{1}/_{16}$
Печ. л. 7.5. Тираж 100 экз.

Государственное образовательное учреждение
высшего профессионального образования
Санкт-Петербургский государственный технологический институт
(технический университет)

190013, Типография издательства СПбГТИ(ТУ), тел. 49-49-365
Санкт-Петербург, Московский пр., 26