Министерство образования Республики Беларусь

Учреждение образования

«Белорусский государственный университет

информатики и радиоэлектроники»

Факультет компьютерных систем и сетей

Кафедра электронных вычислительных машин

Дисциплина: Основы алгоритмизации и программирования

Отчёт

по учебной практике(ознакомительной)

на тему

Программа для решения задачи о Ханойской башне

Студент В.И.Кипятков

Руководитель А.М.Ковальчук

МИНСК 2022

# **Содержание**

1. Введение ………………………………………………………………..
2. Краткие теоретические сведения………………………………………
3. Функция main()………………………………………………………….
4. Описание 2 основных функций………………………………………..
5. Код программы…………………………………………………………
6. Пример работы программы……………………………………………
7. Заключение……………………………………………………………..

Приложение А Блок-схема Функции main()

# **ВВЕДЕНИЕ**

Язык С (С++) часто называют языком среднего уровня. Это означает, что С (С++) объединяет элементы языков высокого уровня с функциональностью Ассемблера.

Языки высокого уровня поддерживают концепцию типов данных. Тип данных определяет набор значений, которые переменная может хранить, и набор операций, которые могут выполняться над переменными. Наряду с тем, что в языке С (С++) представлены все основные типы данных, он не так жестко типизирован, как языки Паскаль или Ада. Язык С (С++) позволяет осуществлять большинство преобразований типов. Контроль за выполнением этих преобразований, а также проверка некоторых ошибок (например, выход за границы массива) возлагается на программиста. Реализованная в С (С++) возможность напрямую манипулировать битами, байтами, словами и указателями необходима для программирования на системном уровне. Язык С (С++) считается структурированным языком. Отличительной чертой структурированного языка является разделение кода и данных. Одним из способов решения этой проблемы является использование подпрограмм (функций), широко использующих локальные переменные. Необходимо отметить, что излишнее использование глобальных переменных может приводить к фатальным ошибкам. Как и ряд других структурированных языков, С (С++) поддерживает ряд операторов цикла, условных операторов и операторов ветвления.

Язык С (С++) содержит стандартные библиотеки, предоставляющие функции, выполняющие наиболее типичные задачи. Эти библиотеки легко могут быть подключены, а также дополнены. Язык С (С++) позволяет разбивать программу на части и выполнять их раздельную компиляцию. Откомпилированные таким образом файлы объединяются для создания полного объектного кода. Преимущество раздельной компиляции в том, что при изменении одного файла не требуется перекомпиляции всей программы.

**Тема работы:** реализация алгоритма решения задачи о Ханойской башне на языке программирования С, используя рекурсивное решение.

**Ханойская башня** является одной из популярных [головоломок](https://ru.wikipedia.org/wiki/%D0%93%D0%BE%D0%BB%D0%BE%D0%B2%D0%BE%D0%BB%D0%BE%D0%BC%D0%BA%D0%B0) [XIX века](https://ru.wikipedia.org/wiki/XIX_%D0%B2%D0%B5%D0%BA). Даны три стержня, на один из которых нанизаны восемь колец, причём кольца отличаются размером и лежат меньшее на большем. Задача состоит в том, чтобы перенести пирамиду из восьми колец за наименьшее число ходов на другой стержень. За один раз разрешается переносить только одно кольцо, причём нельзя класть большее кольцо на меньшее.

В информатике **рекурсия**-это метод **решения** задачи, **решение** которой зависит от **решений** более мелких экземпляров одной и той же задачи. Такие проблемы обычно могут быть решены итерацией, но для этого необходимо идентифицировать и индексировать меньшие экземпляры во время программирования.

### **Рекурсивное решение**

Рекурсивно решаем задачу «перенести башню из *n*−1 диска на 2-й штырь». Затем переносим самый большой диск на 3-й штырь, и рекурсивно решаем задачу «перенеси башню из *n*−1 диска на 3-й штырь».

Отсюда методом [математической индукции](https://ru.wikipedia.org/wiki/%D0%9C%D0%B0%D1%82%D0%B5%D0%BC%D0%B0%D1%82%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B0%D1%8F_%D0%B8%D0%BD%D0%B4%D1%83%D0%BA%D1%86%D0%B8%D1%8F) заключаем, что минимальное число ходов, необходимое для решения головоломки, равно 2*n* − 1, где *n* — число дисков.

**СУТЬ АЛГОРИТМА И РЕАЛИЗАЦИЯ ПРОГРАММЫ**

**Рекурсивный** метод. Для того, чтобы переложить всю пирамиду, надо сначала переложить все, что выше самого большого диска, с первого на вспомогательный стержень, потом переложить это самое большой диск с первого на третий стержень, а потом переложить оставшуюся пирамиду со второго на третий стержень, пользуясь первым стержнем, как вспомогательным.

Рассмотрим Ханойскую башню с **n** дисками, пронумерованными от **1** до **N**. Минимальное время выполнения алгоритма **O(2*n*)**. Это достигается путём [математической индукции](https://ru.wikipedia.org/wiki/%D0%9C%D0%B0%D1%82%D0%B5%D0%BC%D0%B0%D1%82%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B0%D1%8F_%D0%B8%D0%BD%D0%B4%D1%83%D0%BA%D1%86%D0%B8%D1%8F).

Формат ввода:

1. Выбор функции из меню(вывод условия, продолжение решения, начать решение, выйти)
2. Ввод количества дисков на Ханойской башне(от 3 до 15)
3. Выбор способа решения(ручной, машинный)

3.1) Ввод переноса диска(при ручном решении)

Формат вывода:

1. Вывод результата в виде расположения дисков на определённом этапе решения

Структура диска:

struct disk

{

char\* print; //изображение диска

int position; //позиция диска

int size; //размер диска

disk\* down\_disk; //указатель на диск под нынешним

disk\* uper\_disk; //указатель на диск над нынешним

};

Структура стержня:

struct tower

{

disk\* first\_disk; //указатель на верхний диск стержня

disk\* last\_disk; //указатель на нижний стержень диска

int number; //номер стержня

};

**Алгоритм функции** hanoi\_towers

(функция рекурсивного решения задачи)

1. Начало
2. Входные данные

tower\*\* pegs – указатель на массив стержней

int n – количество дисков на вспомогательной башне

tower\* from – с какого стержня перемещать диск

tower\* to – на какой стержень переместить диск

tower\* buf\_peg – промежуточный стержень

int m – общее количество дисков

1. Глобальные переменные

int kol = 0 – номер хода

int MOVE\_TIME – время задержки между перемещениями

1. Если n не равно 0 то переход на шаг 5,

иначе переход на шаг 13

1. Вызов функции hanoi\_towers(pegs, n - 1, from, buf\_peg, to, m)
2. Вызов функции очистки экрана консоли
3. Вывод “[” kol “ / ” 2*m* − 1 “]”
4. kol = kol + 1
5. Вызов функции перемещения верхнего диска стержня

from(from->first\_disk) на верх стержня to

(move\_disk(from->first\_disk, to, from))

1. Вызов функции задержки на время MOVE\_TIME
2. Вызов функции вывода расположения дисков на экран

(jpg(\*pegs, m))

1. Вызов функции hanoi\_towers(pegs, n - 1, buf\_peg, to, from, m)
2. Конец

**Алгоритм функции** jpg

**(вывода расположения дисков)**

1. Начало
2. Входные данные

tower\* pegs – указатель на массив стержней

int n – общее количество дисков

1. Выделение памяти

int peg – счётчик для прохода по стержням

char\*\* picture – массив строк для вывода расположения дисков

disk\* disks – указатель на диск для обхода по стержням

1. peg = 0
2. Если peg < 3 то переход на шаг 5,

Иначе переход на шаг 18

1. Устанавливаем указатель disks на нижний диск стержня pegs[peg]
2. Цикл по стержню pegs[peg] от i = n – 1 до i >= 0 с шагом -1
3. Если disks то переход на шаг 9,

Иначе переход на шаг 13

1. Копируем изображение диска disks в строку picture[i] с позиции peg\*39
2. Перемещаем указатель disks на диск, который находится над нынешним диском
3. i = i - 1
4. Переход на шаг 8
5. Если i >= 0 то переход на шаг 14,

Иначе переход на шаг 15

1. Копируем содержимое строки empty в строку picture[i] с позиции peg\*39
2. Конец цикла по стержню pegs[peg]
3. peg = peg + 1
4. Переход на шаг 5
5. Вывод текста picture

Вывод строки "^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^"

1. Конец

Блок-схема функции main представлена в приложении А.

Далее представлен код выполненной программы с комментариями по осуществлению алгоритма решения задачи Ханойской башни на языке программирования С.

Функция main():

#include "Header.h"

int main()

{

setlocale(LC\_ALL, "rus");

{

int m = 3; //количество стержней

int n = 0; //количество дисков

disk\* disks = nullptr; //указатель на диски

tower\* pegs = (tower\*)calloc(m, sizeof(tower)); //массив стержней

do

{

options(); //вывод функционала

int menu\_traker; //переменная выбора в меню

menu\_traker = isCorrect(); //функция ввода переменной menu\_traker

switch (menu\_traker)

{

case 1: //вывод условия и правил пользования

{

system("CLS");

rules(); //функция вывода условия и правил пользования

rewind(stdin);

while (!getchar()); //ввод ENTER для выхода

system("CLS");

break;

}

case 2: //продолжение последнего сохранения

{

if (!continue\_play(&pegs, &n)) //функция открытия сохранения

break;

start(n, m, disks, pegs); //функция ручного решения

system("CLS");

jpg(pegs, n); //функция вывода расположения дисков

break;

}

case 3: //начало решения

{

pegs = solve(n, m, disks, pegs); //функция начала решения

break;

}

case 0: //выход

{

end(pegs); //функция полной очистки стержней и дисков

return 0;

break;

}

default: //иные ситуации

puts("Такой команды нет"); //сообщение о вводе не существующего

Sleep(ERROR\_TIME); //параметра меню

break;

}

} while (true); //зацикливание решения

return 0;

}

return 0;

}

Функция вывода расположения дисков:

void jpg(tower\* pegs, int n)

{

int peg = 0; //счётчик для прохода по стержням

char\*\* picture = (char\*\*)calloc(n, sizeof(char\*)); //массив строк для вывода расположения дисков

for (int i = 0; i < n; i++)

picture[i] = (char\*)calloc(120, sizeof(char));

while (peg < 3) //массив по стержням

{

disk\* disks = pegs[peg].last\_disk; //указатель на диск для обхода по стержням

for (int i = n - 1; i >= 0; i--) //цикл по стержню pegs[peg]

{

while (disks) //цикл по дискам столбцаpegs[peg]

{

for (int j = 0; j < 39; j++) //копируем содержимое disks->print

picture[i][j + peg \* 39] = disks->print[j]; //в picture[i] с позиции peg \* 39

disks = disks->uper\_disk; //перемещаем указатель на верхний диск

i--;

}

if (i >= 0) //если диски закончелись то заполняем

for (int j = 0; j < 39; j++) //отсутствием диска

picture[i][j + peg \* 39] = empty[j];

}

peg++;

}

for (int i = 0; i < n; i++) //вывод "изображения" дисков

puts(picture[i]);

puts("^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^");

}

Функция символьного представления диска:

void view\_disk(disk\* disks, int n)

{

for (int i = 0; i < n; i++)

{

for (int j = 0; j < 18 - i; j++) //пробелы слева от диска

disks->print[j] = ' ';

disks->print[18 - i] = '['; //левый край диска

for (int j = 19 - i; j < 20 + i; j++) //центр диска

disks->print[j] = '-';

disks->print[20 + i] = ']'; //правый край диска

for (int j = 21 + i; j < 39; j++) //пробела справа от диска

disks->print[j] = ' ';

disks = disks->down\_disk;

}

}

Функция проверки перестановки диска:

int cheak(tower\* pegs, int from, int to)

{

if (from < 0 || from > 2 || to < 0 || to > 2) //проверка на ввод не

{ //существующего столбца

puts("Такого столбика нет(есть только 1, 2 и 3)");

Sleep(ERROR\_TIME);

return 0;

}

if (!pegs[from].first\_disk) //проверка на наличее диска

{

puts("На этой башне нет дисков");

Sleep(ERROR\_TIME);

return 0;

}

if (pegs[to].first\_disk) //проверка на перемещение большего

if (pegs[from].first\_disk->size > pegs[to].first\_disk->size)//на меньший диск

{

puts("Этот диск меньше");

Sleep(ERROR\_TIME);

return 0;

}

return 1;

}

Функция проверки завершённости решения:

int isCompleet(tower peg, int n)

{

int kol\_disks = 0;

disk\* under\_disk = peg.first\_disk;

while (under\_disk) //наличие всех дисков на правом стержне

{

kol\_disks++;

under\_disk = under\_disk->down\_disk;

}

if (kol\_disks == n)

return 1;

else

return 0;

}

Функция выделения памяти для диска:

disk\* give\_memorry\_disk(int n)

{

disk\* pointer;

if (!(pointer = (disk\*)calloc(n, sizeof(disk))))

{

puts("Нет свободной памяти");

return nullptr;

}

pointer->print = (char\*)calloc(40, 1);

return pointer;

}

Функция удаления диска:

void pop\_disk(disk\* disk\_u, tower\* peg\_from)

{

if (peg\_from)

{

(\*peg\_from).first\_disk = (\*disk\_u).down\_disk;

if ((\*disk\_u).down\_disk)

(\*disk\_u).down\_disk->uper\_disk = nullptr;

else

{

(\*peg\_from).first\_disk = nullptr;

(\*peg\_from).last\_disk = nullptr;

}

(\*disk\_u).down\_disk = nullptr;

}

}

Функция добавления диска:

void push\_disk(disk\* disk\_u, tower\* peg\_to)

{

if (disk\_u)

{

if (!(\*peg\_to).last\_disk)

(\*peg\_to).last\_disk = disk\_u;

if ((\*peg\_to).first\_disk)

(\*peg\_to).first\_disk->uper\_disk = disk\_u;

(\*disk\_u).position = (\*peg\_to).number;

(\*disk\_u).down\_disk = (\*peg\_to).first\_disk;

(\*peg\_to).first\_disk = disk\_u;

}

}

Функция перемещения диска с одного стержня на другой:

void move\_disk(disk\* disk\_u, tower\* peg\_to, tower\* peg\_from)

{

if (disk\_u)

{

pop\_disk(disk\_u, peg\_from);

push\_disk(disk\_u, peg\_to);

}

}

Функция открытия файла:

FILE\* bopenf(const char\* namef)

{

FILE\* f;

if (fopen\_s(&f, namef, "rb+")) //Открытие файлов

{

puts("Сохранений нет");

}

return f;

}

Функция чтения сохранения из файла:

tower\* bfreed\_text(FILE\* f, tower\* pegs, int\* n)

{

fseek(f, 0, 0);

fread(n, 4, 1, f);

int i = 0;

for (i = 0; i < 3; i++) //читаем номера столбцов

{

fread(&(pegs[i].number), sizeof(int), 1, f);

}

for (int i = 0; i < \*n; i++)

{

disk\* saved\_disk = give\_memorry\_disk(1);

fread(saved\_disk->print, 40, 1, f); //читаем символьное представление диска

fread(&(saved\_disk)->size, 4, 1, f); //читаем размер диска

fread(&(saved\_disk)->position, 4, 1, f); //читаем позицию диска

push\_disk(saved\_disk, &pegs[saved\_disk->position]);//устанавливаем диск на нужное место

}

return pegs;

}

Функция записи сохранения в файл:

void bfprint\_result(FILE\* f, tower\* pegs, int n)

{

fseek(f, 0, 0);

int i = 0;

fwrite(&n, 4, 1, f);

for (; i < 3; i++) //записываем номера столбцов

{

fwrite(&(pegs[i].number), sizeof(int), 1, f);

}

for (i = 0; i < 3; i++)

{

disk\* saved\_disk = pegs[i].last\_disk;

while (saved\_disk)

{

fwrite(saved\_disk->print, 40, 1, f); //записываем символьное представление диска

fwrite(&(saved\_disk)->size, 4, 1, f); //записываем размер диска

fwrite(&(saved\_disk)->position, 4, 1, f); //записываем позицию диска

saved\_disk = saved\_disk->uper\_disk;//перемещаем указатель на следующий диск

}

}

}

**ПРИМЕР ВЫПОЛНЕНИЯ ЗАДАЧИ ПРОГРАММОЙ**

Дана Ханойская башня с 3 дисками, нужно решить за минимальное количество шагов.

На рисунках 1-7 представлены пример работы программы по исходным данным. В примере переменная MOVE\_TIME равна 1000, таким образом на экране будут появляться расположения с периодом 1000 мс или 1с.

Рис. 1 – пример работы программы(1 шаг – перенос с 1 на 3)

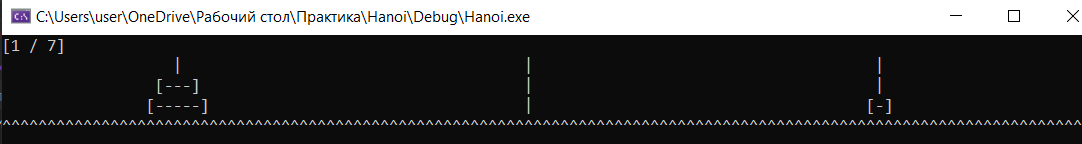


Рис. 2 – пример работы программы(2 шаг – перенос с 1 на 2)

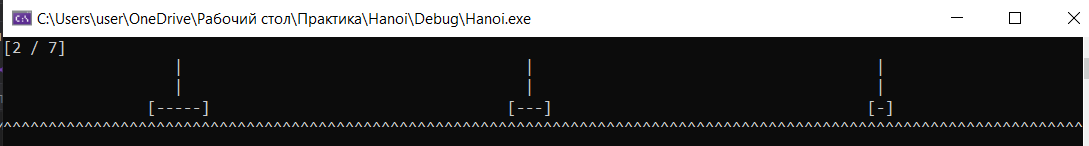


Рис. 3 – пример работы программы(3 шаг – перенос с 3 на 2)

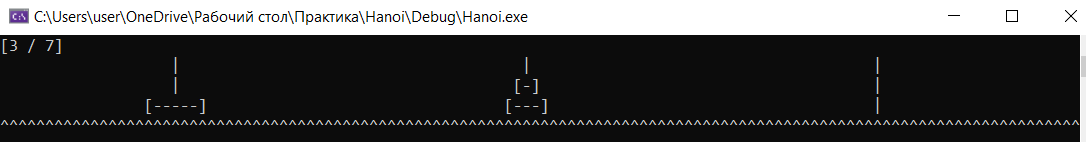


Рис. 4 – пример работы программы(4 шаг – перенос с 1 на 3)

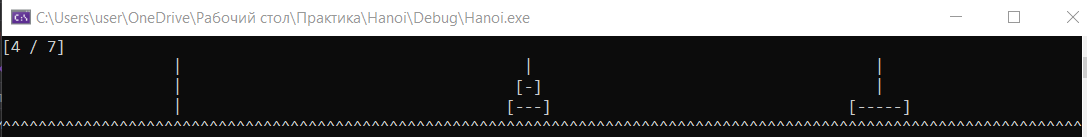


Рис. 5 – пример работы программы(5 шаг – перенос с 2 на 1)

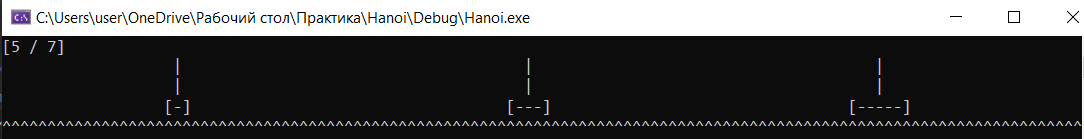


Рис. 6 – пример работы программы(6 шаг – перенос с 2 на 3)

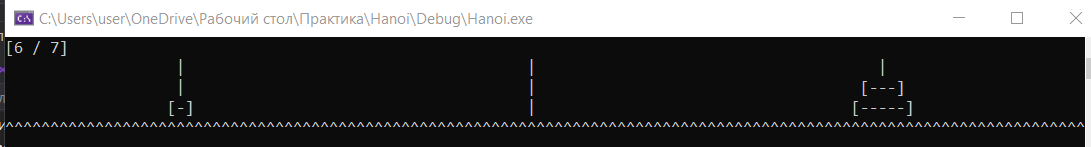
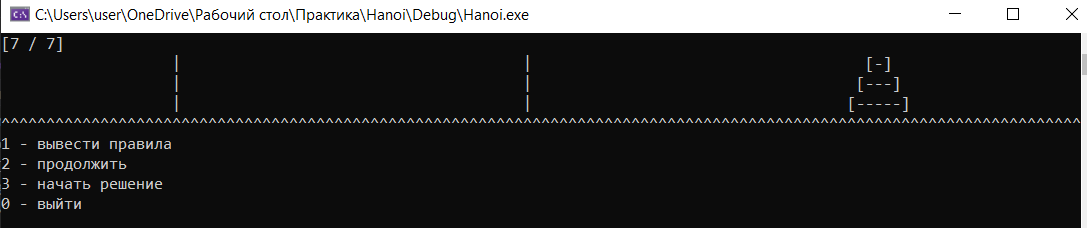


Рис. 7 – пример работы программы(7 шаг – перенос с 1 на 3)



Также при ручном решении можно сохранить расположение дисков в бинарные файлы.

Рис. 8 – пример сохраняемого расположения дисков

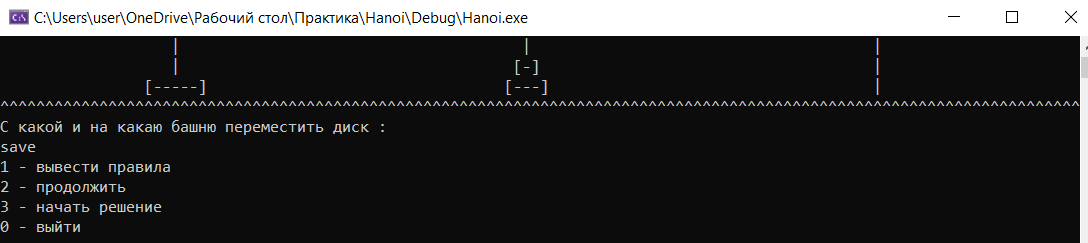
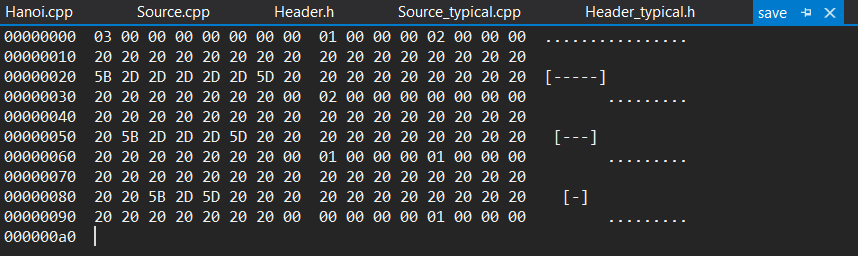


Рис. 9 – пример того, как сохранение выглядит в бинарном файле



**ЗАКЛЮЧЕНИЕ**

В процессе выполнения учебной практики были выполнены первоначально заданные цели, а именно реализован алгоритм решения задачи о Ханойской башне. Время работы составляет **O(2*n*)**, где n – количество дисков.

Задача о Ханойской башне является каноном в освоении рекурсии и часто применяется во многих курсах для познания основ алгоритмизации.

Моя программа помогает решить эту известную задачку за меньшее количество времени нежили человек. Если бы человек решал эту головоломку при 15 дисках и на один ход ему бы потребовалось 1 с, то он решал бы 32 767с, 546 мин или 9 ч.

ОС – Windows 10

Среда разработки – Microsoft Visual Studio 2019

Минимальные системные требования(при 15 дисках):

Занимаемая память процессора – 86 мб

Время выполнения – 47 мин 25 с 623 мс

# **СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

Луцик Ю.А., Ковальчук А.М., Сасин Е.А. – Учебное пособие по курсу «Основы алгоритмизации и программирования: язык С». – Минск: БГУИР, 2015 г.

Луцик Ю.А., Ковальчук А.М., Лукьянова И.В., Бушкевич А.В. – Лабораторный практикум по курсу «Основы алгоритмизации и программирования» (в 2-х частях). – Минск: БГУИР, 2008 г.

**ПРИЛОЖЕНИЕ А**

*(обязательное)*

Блок-схема алгоритма функции main()