

Балтийский государственный технический университет
«ВОЕНМЕХ» им. Д. Ф. Устинова

Кафедра О7 «Информационные системы и программная инженерия»

Практическая работа №3

по дисциплине «Информатика: Основы программирования»
на тему «Вспомогательные алгоритмы. Функции.
Создание статических библиотек»

Выполнил:

Студент *Фамилия И.О.*

Группа *Номер_группы*

Преподаватель:

Фамилия И.О.

Санкт-Петербург

2023 г.

Задание 1. Восходящее программирование

Перепишите программу 7 из ПР_2, разделив ее текст на функции. Каждая функция должна выполнять только одно законченное действие (ввод массива, вывод массива, поиск значения в массиве, удаление элемента из массива и т.п.).

Текст вариативной части задания 7 ПР_2.

Исходные данные:

описываем входные данные, их обозначение в программе и тип.

Результирующие данные:

описываем выходные данные, их обозначение в программе и тип.

Схема программы без разделения на функции: *(из ПР_2)*

Здесь должна быть схема программы в соответствии с ГОСТ 19.701-90.

Текст программы без разделения на функции: *(из ПР_2)*

Сюда добавляем текст программы 7 из ПР_2. Шрифт Courier New или FreeMono 10 pt, междустрочный интервал одинарный.

Декомпозиция

Основной алгоритм:

Схема основного алгоритма в соответствии с ГОСТ 19.701-90. Схема может быть построена любым способом, в том числе начерчена вручную на листе бумаги с помощью карандаша и линейки и сфотографирована или отсканирована.

Вспомогательные алгоритмы:

1) Алгоритм ввода массива

Входные данные: имя массива, размер.

Результирующие данные: заполненный массив

Схема алгоритма:

Схема алгоритма в соответствии с ГОСТ 19.701-90.

Этот алгоритм описывается в программе функцией

прототип функции (Шрифт Courier New или FreeMono 10 pt)

Параметры:

первый параметр – адрес первого элемента массива

второй параметр – количество элементов массива

третий параметр - ...

и т.д.

Возвращаемое значение – количество считанных элементов (если функция ничего не возвращает (тип результата void), то пишем «отсутствует»)

Побочный эффект – изменение значений элементов массива (если у функции нет побочного

эффекта, то пишем «отсутствует»)

Вспомогательные переменные:

описываем, для чего нужны, их обозначение в программе и тип.

2) Алгоритм вывода массива

Входные данные: имя массива, размер.

Результирующие данные: нет

Схема алгоритма:

Схема алгоритма в соответствии с ГОСТ 19.701-90.

Этот алгоритм описывается в программе функцией

прототип функции (Шрифт Courier New или FreeMono 10 nm)

Параметры:

первый параметр – адрес первого элемента массива

второй параметр – количество элементов массива

третий параметр - ...

и т.д.

Возвращаемое значение отсутствует.

Побочный эффект отсутствует.

Вспомогательные переменные:

описываем, для чего нужны, их обозначение в программе и тип.

3) Алгоритм поиска элемента по ключу

Входные данные: имя массива, размер, искомый ключ.

Результирующие данные: номер элемента в массиве или 0 в случае отсутствия искомого значения

Схема алгоритма:

Схема алгоритма в соответствии с ГОСТ 19.701-90.

Этот алгоритм описывается в программе функцией

прототип функции (Шрифт Courier New или FreeMono 10 nm)

Параметры:

первый параметр – адрес первого элемента массива

второй параметр – количество элементов массива

третий параметр – искомый ключ...

и т.д.

Возвращаемое значение – адрес найденного элемента или NULL в случае отсутствия искомого значения.

Побочный эффект *отсутствует*.

Вспомогательные переменные:

описываем, для чего нужны, их обозначение в программе и тип.

4) Алгоритм *удаления элемента по ключу*.

Входные данные: имя массива, размер, *искомый ключ*.

Результирующие данные: *размер массива, измененный массив*

Схема алгоритма:

Схема алгоритма в соответствии с ГОСТ 19.701-90.

Этот алгоритм описывается в программе функцией

прототип функции (Шрифт Courier New или FreeMono 10 pt)

Параметры:

первый параметр – *адрес первого элемента массива*

второй параметр – *количество элементов массива*

третий параметр – искомый ключ...

и т.д.

Возвращаемое значение – *количество элементов массива.*

Побочный эффект – *изменение значений элементов массива.*

Вспомогательные переменные:

описываем, для чего нужны, их обозначение в программе и тип.

5) Алгоритм *такой-то*

...

и так далее для всех вспомогательных алгоритмов

Текст программы:

Сюда добавляем текст программы с комментариями. Шрифт Courier New или FreeMono 10 pt, междустрочный интервал одинарный.

Результаты работы программ:

без разделения на функции

скриншот

после разделения на функции

скриншот

Задание 2. Распределение памяти

1. *Операционная система Kubuntu 18.04.2, IDE Code::Blocks*

Скриншот окна работы программы

```
/* адреса функций */
printf("Adress of main : %p\n", main);
printf("Adress of f1 : %p\n", f1);
printf("Adress of f2 : %p\n", f2);
printf("Adress of f3 : %p\n", f3);
printf("Adress of f4 : %p\n", f4);
printf("Adress of f5 : %p\n\n", f5);
```

Стартовые адреса функций в памяти (схемы могут быть построены с использованием приведенной таблицы, выполнены в любом удобном для Вас редакторе, а также начерчены на листе бумаги и отсканированы или сфотографированы (текст, набранный красным курсивом, нужно удалить)):

[illegible]

Делаем выводы о размещении кода функций

```

/* адрес глобальной переменной */
printf("var_global : adress : %p\tvalue : %d\n\n", &var_global, var_global);

/* адреса локальных переменных */
int var_main = 100;
printf("var_main      : adress : %p\tvalue : %d\n", &var_main, var_main);

void * pointer1_main = malloc (1024);
printf("pointer1_main : adress : %p\t", &pointer1_main);
printf("value : %p\tmemory block size : %d bytes\n", pointer1_main, 1024); /* адрес динамически выделенного блока памяти */
free(pointer1_main); /* освобождение памяти */
printf("pointer1_main memory free\n\n");

/* вызовы функций. Внутри функции f1 тоже динамически выделяется и освобождается блок памяти */
f1(1);
f2(2, 3.0);
pointer1_main = malloc (1024); /* повторное выделение памяти того же объема */

```

```
printf("pointer1_main : new value : %p\tmemory block size : %d bytes\n\n", pointer1_main, 1024);

/* выделение и освобождение следующего блока памяти */
void * pointer2_main = malloc (1024);
printf("pointer2_main : adress : %p\tvalue : %p\tmemory block size : %d bytes\n\n", &pointer2_main, pointer2_main, 1024);
free (pointer2_main);
printf("pointer2_main memory free\n\n");

f1(4);
f3(5);
f4(6);
f3(7);
```

Распределение памяти под переменные: (схемы могут быть построены с использованием приведенной таблицы, выполнены в любом удобном для Вас редакторе, а также начерчены на листе бумаги и отсканированы или сфотографированы. Каждый столбец – один байт, адреса записывать последовательно, начиная с младшего адреса. Размещение переменных показать объединением ячеек, значения переменных записать внутри объединенной ячейки (как в заданиях 3 и 4 второй практической работы). Текст, набранный красным курсивом, нужно удалить)

[illegible]

вызов функции f1																															
param_f1																															
var_f1																															
pointer_f1																															
вызов функции f3																															
param_f3																															
var_f3																															
вызов функции f4, вызов из нее f5 и рекурсивные вызовы f5																															
param_f4																															
param_f5																															
param_f5																															
param_f5																															
param_f5																															
param_f5																															
param_f5																															
param_f5																															
param_f5																															
вызов функции f3																															
param_f3																															
var_f3																															

Делаем выводы о том,

- где выделяется память под глобальные переменные,*
- где выделяется память под локальные переменные класса auto и класса static,*
- как выделяется память при последовательном вызове функций и в случае, когда одна функция вызывается из другой (в том числе рекурсивно из самой себя),*
- где выделяется память при динамическом ее выделении.*

2. *Онлайн-IDE Replit <https://replit.com/languages/c>*

Скриншот окна работы программы

Стартовые адреса функций в памяти:

Задание 3. Создание библиотеки функций ввода-вывода массивов

1) Алгоритм ввода массива

Входные данные: имя массива, размер.

Результирующие данные: количество считанных элементов, значения элементов массива

Схема алгоритма:

Схема алгоритма в соответствии с ГОСТ 19.701-90.

Этот алгоритм описывается в программе функциями *(если одной, то пишем один прототип)*

прототип функции (Шрифт Courier New или FreeMono 10 nm)

прототип функции (Шрифт Courier New или FreeMono 10 nm)

первый параметр – *адрес первого элемента массива*

второй параметр – *количество элементов массива*

третий параметр - ...

и т.д.

Возвращаемое значение – количество считанных элементов (если функция ничего не возвращает (тип результата void), то пишем «отсутствует»)

Побочный эффект – изменение значений элементов массива (если у функции нет побочного эффекта, то пишем «отсутствует»)

Вспомогательные переменные:

описываем, для чего нужны, их обозначение в программе и тип.

2) Алгоритм вывода массива в виде непрерывной последовательности чисел, разделенных пробелами.

Входные данные: имя массива, размер.

Результирующие данные: *нет*

Схема алгоритма:

Схема алгоритма в соответствии с ГОСТ 19.701-90.

Этот алгоритм описывается в программе функциями

прототип функции (Шрифт Courier New или FreeMono 10 nm)

прототип функции (Шрифт Courier New или FreeMono 10 nm)

Параметры:

первый параметр – *адрес первого элемента массива*

второй параметр – *количество элементов массива*

третий параметр - ...

и т.д.

Возвращаемое значение отсутствует.

Побочный эффект отсутствует.

Вспомогательные переменные:

описываем, для чего нужны, их обозначение в программе и тип.

3) Алгоритм вывода массива в виде таблицы.

Входные данные: имя массива, размер.

Результирующие данные: нет

Схема алгоритма:

Схема алгоритма в соответствии с ГОСТ 19.701-90.

Этот алгоритм описывается в программе функциями

прототип функции (Шрифт Courier New или FreeMono 10 nm)

прототип функции (Шрифт Courier New или FreeMono 10 nm)

Параметры:

первый параметр – адрес первого элемента массива

второй параметр – количество элементов массива

третий параметр - ...

и т.д.

Возвращаемое значение отсутствует.

Побочный эффект отсутствует.

Вспомогательные переменные:

описываем, для чего нужны, их обозначение в программе и тип.

Заголовочный файл *имя_файла.h*:

Сюда добавляем текст заголовочного файла. Шрифт Courier New или FreeMono 10 nm, междустрочный интервал одинарный.

Файл реализации *имя_файла.c*

Сюда добавляем текст файла, содержащего определения функций. Шрифт Courier New или FreeMono 10 nm, междустрочный интервал одинарный.

Полученный файл библиотеки *имя_файла.a* имеет размер 579 байт.

Задание 4. Создание библиотеки для работы с динамической матрицей

1) Алгоритм выделения памяти под прямоугольную динамическую матрицу

Входные данные: количество блоков данных (строк или столбцов), количество элементов в одном блоке (строке или столбце), размер одного элемента данных в байтах.

Результирующие данные: адрес начала блока указателей

Схема алгоритма:

Схема алгоритма в соответствии с ГОСТ 19.701-90.

Этот алгоритм описывается в программе функцией

прототип функции (Шрифт Courier New или FreeMono 10 nm)

первый параметр – *такой-то*

второй параметр – *такой-то*

третий параметр - ...

и т.д.

Возвращаемое значение – адрес начала блока указателей или NULL в случае невозможности выделить память

Побочный эффект отсутствует

2) Алгоритм освобождения памяти от динамической матрицы.

Входные данные: адрес начала блока указателей, количество блоков данных.

Результирующие данные: нет

Схема алгоритма:

Схема алгоритма в соответствии с ГОСТ 19.701-90.

Этот алгоритм описывается в программе функцией

прототип функции (Шрифт Courier New или FreeMono 10 nm)

Параметры:

первый параметр – *такой*

второй параметр – *такой*

третий параметр - ...

и т.д.

Возвращаемое значение отсутствует.

Побочный эффект отсутствует.

3) Алгоритм *такой-то*

...

и так далее для всех вспомогательных алгоритмов

Заголовочный файл *имя_файла.h*:

Сюда добавляем текст заголовочного файла. Шрифт Courier New или FreeMono 10 nm, междустрочный интервал одинарный.

Файл реализации *имя_файла.c*

Сюда добавляем текст файла, содержащего определения функций. Шрифт Courier New или FreeMono 10 nm, междустрочный интервал одинарный.

Полученный файл библиотеки *имя_файла.a* имеет размер 579 байт.

Задание 5. Обработка одномерных и многомерных массивов

5.1. Текст вариативной части задания 5.1.

Исходные данные:

описываем входные данные, их обозначение в программе и тип, для массивов и матриц

указываем способ выделения памяти (статический одномерный массив, динамический одномерный массив, статический двумерный массив, динамический двумерный массив, динамическая матрица).

Результирующие данные:

описываем выходные данные, их обозначение в программе и тип.

Таблица тестирования:

Входные данные	Ожидаемый результат	Результат работы программы
наборы данных должны соответствовать максимально большому количеству возможных вариантов		скриншот
		скриншот
		скриншот
		скриншот

Основной алгоритм:

Схема основного алгоритма в соответствии с ГОСТ 19.701-90. Схема может быть построена любым способом, в том числе начерчена вручную на листе бумаги с помощью карандаша и линейки и сфотографирована или отсканирована.

Вспомогательные алгоритмы:

- 1) Для ввода элементов массива используется функция `имя_функции` из созданной библиотеки `имя_библиотеки` (если это не так, то описываем всё полностью: входные и выходные данные, схему, функцию и т.д.)
- 2) Алгоритм поиска элемента по ключу

Входные данные: имя массива, размер, **искомый** ключ.

Результирующие данные: номер элемента в массиве или 0 в случае отсутствия искомого значения

Схема алгоритма:

Схема алгоритма в соответствии с ГОСТ 19.701-90.

Этот алгоритм описывается в программе функцией

прототип функции (Шрифт Courier New или FreeMono 10 nm)

Параметры:

первый параметр – адрес первого элемента массива

второй параметр – количество элементов массива

третий параметр – **искомый** ключ...

и т.д.

Возвращаемое значение – адрес найденного элемента или *NULL* в случае отсутствия искомого значения.

Побочный эффект отсутствует.

3) Алгоритм такой-то

...

и так далее для всех вспомогательных алгоритмов

Текст программы:

Сюда добавляем текст программы с комментариями. Шрифт *Courier New* или *FreeMono 10 nt*, междустрочный интервал одинарный.

5.2. Текст вариативной части задания 5.2.

Исходные данные:

описываем входные данные, их обозначение в программе и тип, для массивов и матриц указываем способ выделения памяти (статический одномерный массив, динамический одномерный массив, статический двумерный массив, динамический двумерный массив, динамическая матрица).

Результирующие данные:

описываем выходные данные, их обозначение в программе и тип.

Таблица тестирования:

Входные данные	Ожидаемый результат	Результат работы программы
наборы данных должны соответствовать максимально большему количеству возможных вариантов		скриншот
		скриншот
		скриншот
		скриншот

Основной алгоритм:

Схема основного алгоритма в соответствии с ГОСТ 19.701-90. Схема может быть построена любым способом, в том числе начерчена вручную на листе бумаги с помощью карандаша и линейки и сфотографирована или отсканирована.

Вспомогательные алгоритмы:

1) Для ввода элементов массива используется функция *имя_функции* из созданной библиотеки *имя_библиотеки* (если это не так, то описываем всё полностью: входные и выходные данные, схему, функцию и т.д.)

2) Алгоритм *поиска элемента по ключу и описывающая его функция те же, что и в предыдущей задаче (если это не так, то описываем всё полностью: входные и выходные данные, схему, функцию и т.д.)*

3) Алгоритм *такой-то*

(описываем всё полностью: входные и выходные данные, схему, функцию и т.д.)

...

и так далее для всех вспомогательных алгоритмов

Текст программы:

Сюда добавляем текст программы с комментариями. Шрифт Courier New или FreeMono 10 nt, междустрочный интервал одинарный.

5.3. Текст вариативной части задания 5.3.

Исходные данные:

описываем входные данные, их обозначение в программе и тип, для массивов и матриц указываем способ выделения памяти (статический одномерный массив, динамический одномерный массив, статический двумерный массив, динамический двумерный массив, динамическая матрица).

Результирующие данные:

описываем выходные данные, их обозначение в программе и тип.

Таблица тестирования:

Входные данные	Ожидаемый результат	Результат работы программы
<i>наборы данных должны соответствовать максимально большему количеству возможных вариантов</i>		<i>скриншот</i>
		<i>скриншот</i>
		<i>скриншот</i>
		<i>скриншот</i>

Основной алгоритм:

Схема основного алгоритма в соответствии с ГОСТ 19.701-90. Схема может быть построена любым способом, в том числе начерчена вручную на листе бумаги с помощью карандаша и линейки и сфотографирована или отсканирована.

Вспомогательные алгоритмы:

1) *Для ввода элементов массива используется функция имя_функции из созданной библиотеки имя_библиотеки (если это не так, то описываем всё полностью: входные и выходные данные, схему, функцию и т.д.)*

2) Алгоритм *поиска элемента по ключу* и описывающая его функция те же, что и в предыдущей задаче (если это не так, то описываем всё полностью: входные и выходные данные, схему, функцию и т.д.)

3) Алгоритм *такой-то* (описываем всё полностью: входные и выходные данные, схему, функцию и т.д.)

...

и так далее для всех вспомогательных алгоритмов

Текст программы:

Сюда добавляем текст программы с комментариями. Шрифт Courier New или FreeMono 10 nt, междустрочный интервал одинарный.

5.4. Текст вариативной части задания 5.4.

Исходные данные:

описываем входные данные, их обозначение в программе и тип, для массивов и матриц указываем способ выделения памяти (статический одномерный массив, динамический одномерный массив, статический двумерный массив, динамический двумерный массив, динамическая матрица).

Результирующие данные:

описываем выходные данные, их обозначение в программе и тип.

Таблица тестирования:

Входные данные	Ожидаемый результат	Результат работы программы
наборы данных должны соответствовать максимально большому количеству возможных вариантов		скриншот
		скриншот
		скриншот
		скриншот

Основной алгоритм:

Схема основного алгоритма в соответствии с ГОСТ 19.701-90. Схема может быть построена любым способом, в том числе начерчена вручную на листе бумаги с помощью карандаша и линейки и сфотографирована или отсканирована.

Вспомогательные алгоритмы:

1) Для ввода элементов массива используется функция `имя_функции` из созданной библиотеки `имя_библиотеки` (если это не так, то описываем всё полностью: входные и выходные данные, схему, функцию и т.д.)

2) Алгоритм *поиска элемента по ключу* и описывающая его функция те же, что и в предыдущей задаче (если это не так, то описываем всё полностью: входные и выходные данные, схему, функцию и т.д.)

3) Алгоритм *такой-то* (описываем всё полностью: входные и выходные данные, схему, функцию и т.д.)

...

и так далее для всех вспомогательных алгоритмов

Текст программы:

Сюда добавляем текст программы с комментариями. Шрифт *Courier New* или *FreeMono 10 nt*, междустрочный интервал одинарный.

Задание 6. Вычисление интеграла

Определить функцию *Integral()* для приближенного вычисления определенного интеграла вида $\int_a^b f(x) dx$ методом *прямоугольников* с задаваемой параметром точностью.

Использовать эту функцию для вычисления значений двух интегралов: *первый интеграл (подставляем формулу из вариативной части задания)*, *второй интеграл (подставляем формулу из вариативной части задания)*, передавая в функцию *Integral()* подынтегральную функцию, пределы интегрирования и точность вычислений.

Формула прямоугольников $\int_a^b f(x) dx \approx \frac{b-a}{N} \sum_{i=1}^N f(x_i)$

Исходные данные:

- нижний и верхний пределы интегрирования задаются константами (если они вводятся пользователем, описываем переменные, в которые помещаются значения),
- подынтегральные функции описываются функциями в программе,
- точность вычислений задается пользователем, используем для нее переменную *eps* типа *double*.

Результирующие данные:

значение интеграла – вещественное число

Таблица тестирования:

Входные данные	Ожидаемый результат	Результат работы программы
$\int_0^1 x dx$	0.5	скриншот
$\int_0^1 (-x) dx$	-0.5	скриншот

Вспомогательные переменные:

если нужны вспомогательные переменные, описываем, для чего они нужны, их обозначение и тип.

Таблица тестирования:

Входные данные	Ожидаемый результат	Результат работы программы
		<i>скриншот</i>

Схема программы

Здесь должна быть схема программы (основной алгоритм, вспомогательные алгоритмы) в соответствии с ГОСТ 19.701-90. Схема может быть построена любым способом, в том числе начерчена вручную на листе бумаги с помощью карандаша и линейки и сфотографирована или отсканирована.

Текст программы

Сюда добавляем текст программы с комментариями. Шрифт Courier New или FreeMono 10 pt, междустрочный интервал одинарный.

Задание 7. Сортировка

Напишите программу упорядочения прямоугольной целочисленной матрицы в соответствии с указанным в вариативной части задания правилом.

Текст вариативной части задания

Исходные данные:

описываем входные данные, их обозначение в программе и тип, для матрицы указываем способ выделения памяти (статический одномерный массив, динамический одномерный массив, статический двумерный массив, динамический двумерный массив, динамическая матрица).

Результирующие данные:

описываем выходные данные, их обозначение в программе и тип.

а) Для сортировки определяется отдельная функция, *реализующая метод пузырька*.

Таблица тестирования:

Входные данные	Ожидаемый результат	Результат работы программы
<i>наборы данных должны соответствовать максимально большему количеству возможных вариантов</i>		<i>скриншот</i>

		скриншот
		скриншот
		скриншот

Основной алгоритм:

Схема основного алгоритма в соответствии с ГОСТ 19.701-90. Схема может быть построена любым способом, в том числе начерчена вручную на листе бумаги с помощью карандаша и линейки и сфотографирована или отсканирована.

Вспомогательные алгоритмы:

1) Для выделения памяти под динамическую матрицу используется функция *имя_функции* из созданной библиотеки *имя_библиотеки* (если это не так, то описываем всё полностью: входные и выходные данные, схему, функцию и т.д.)

2) Для ввода элементов матрицы используется функция *имя_функции* из созданной библиотеки *имя_библиотеки* (если это не так, то описываем всё полностью: входные и выходные данные, схему, функцию и т.д.)

3) Алгоритм *сортировки последовательности*

Входные данные: количество чисел, последовательность чисел.

Результирующие данные: измененная последовательность чисел

Схема алгоритма:

Схема алгоритма в соответствии с ГОСТ 19.701-90.

Этот алгоритм описывается в программе функцией

прототип функции (Шрифт Courier New или FreeMono 10 nt)

Параметры:

первый параметр – адрес первого элемента массива

второй параметр – количество элементов массива

третий параметр – ...

и т.д.

Возвращаемое значение – нет.

Побочный эффект изменение последовательности элементов в массиве.

и так далее для всех вспомогательных алгоритмов

Текст программы:

Сюда добавляем текст программы с комментариями. Шрифт Courier New или FreeMono 10 nt, междустрочный интервал одинарный.

б) Для сортировки используется стандартная функция `qsort()`.

Таблица тестирования:

Входные данные	Ожидаемый результат	Результат работы программы
<i>наборы данных должны соответствовать максимально большому количеству возможных вариантов</i>		<i>скриншот</i>
		<i>скриншот</i>
		<i>скриншот</i>
		<i>скриншот</i>

Основной алгоритм:

Схема основного алгоритма не отличается от приведенной в варианте а) (если это не так, то приводим новую схему).

Вспомогательные алгоритмы:

1) *Для выделения памяти под динамическую матрицу используется функция `имя_функции` из созданной библиотеки `имя_библиотеки` (если это не так, то описываем всё полностью: входные и выходные данные, схему, функцию и т.д.)*

2) *Для ввода элементов матрицы используется функция `имя_функции` из созданной библиотеки `имя_библиотеки` (если это не так, то описываем всё полностью: входные и выходные данные, схему, функцию и т.д.)*

3) *Алгоритм сравнения строк матрицы*

Входные данные: две числовые последовательности.

Результирующие данные: значение ≤ 0 , если первая строка в упорядоченной матрице должна предшествовать второй, значение >0 , если вторая строка в упорядоченной матрице должна предшествовать первой.

Схема алгоритма:

Схема алгоритма в соответствии с ГОСТ 19.701-90.

Этот алгоритм описывается в программе функцией

прототип функции (Шрифт Courier New или FreeMono 10 nm)

Параметры:

первый параметр – адрес первой строки матрицы

второй параметр – адрес второй строки матрицы

Возвращаемое значение – целое положительное число, если вторая строка в упорядоченной матрице должна предшествовать первой, в противном случае 0 или отрицательное целое число.

Побочный эффект нет.

4) Алгоритм *вычисления суммы цифр в десятичной записи числа*

Входные данные: целое число.

Результирующие данные: сумма цифр в десятичной записи этого числа.

Схема алгоритма:

Схема алгоритма в соответствии с ГОСТ 19.701-90.

Этот алгоритм описывается в программе функцией

прототип функции (Шрифт Courier New или FreeMono 10 nt)

Параметр – анализируемое целое число

Возвращаемое значение – целое число – сумма цифр в десятичной записи анализируемого числа.

Побочный эффект нет.

и так далее для всех вспомогательных алгоритмов

Текст программы:

Сюда добавляем текст программы с комментариями. Шрифт Courier New или FreeMono 10 nt, междустрочный интервал одинарный.