**Оглавление:**

[Введение 2](#_Toc160516787)

[ТЕОРЕТИЧЕСКАЯ ЧАСТЬ 2](#_Toc160516788)

1. [Информация и данные 2](#_Toc160516789)
2. [Классификация структур данных 2](#_Toc160516790)
3. [Основные структуры данных 2](#_Toc160516791)

[Простые структуры данных 2](#_Toc160516792)

[Статические структуры данных 2](#_Toc160516793)

[Полустатические структуры данных 2](#_Toc160516794)

[Динамические структуры данных 2](#_Toc160516795)

[Нелинейные структуры данных 2](#_Toc160516796)

[Заключение 2](#_Toc160516797)

[ПРАКТИЧЕСКАЯ ЧАСТЬ 2](#_Toc160516798)

[Общая характеристика задачи 2](#_Toc160516799)

[Описание алгоритма решения задачи **Ошибка! Закладка не определена.**](#_Toc160516800)

[Список использованной литературы 2](#_Toc160516801)

# Введение

Веками человечество накапливало знания, навыки работы, сведения об окружающем нас мире, т.е. собирало информацию. Вна­чале информация передавалась из поколения в поколение в виде преданий и устных рассказов. Возникновение и развитие книжного дела позволило передавать и хранить информацию в более надежном письменном виде. Открытия в области электричества привели к появлению телеграфа, теле­фона, радио, телевидения — средств, позволяющих оперативно передавать и накапливать информацию. Развитие прогресса обусловило резкий рост информации, в связи, с чем вопрос о её сохранении и переработке стано­вился год от года острее. С появлением вычислительной техники значи­тельно упростились способы хранения, а главное, обработки информации. Развитие вычислительной техники на базе микропроцессоров приводит к совершенствованию компьютеров и программного обеспечения. Появля­ются программы, способные обработать большие потоки информации. С помощью таких программ создаются информационные системы. Целью любой информационной системы является обработка данных об объектах и явлениях реального мира и предоставление нужной человеку информа­ции о них.

В данной работе рассматривается что такое информация и данные, чем они различаются; как информация переходит в структурированные данные. Рассматриваются такие понятия, как «тип данных», «структура данных», «модель данных» и «база данных». В основной части работы приводится классификация структур данных, обширная информация о физическом и логическом представлении структур данных всех классов памяти ЭВМ: простых, статических, полустатических, динамических и нелинейных; а также, информация о возможных операциях над всеми перечисленными структурами.

# ТЕОРЕТИЧЕСКАЯ ЧАСТЬ

# ****Информация и данные****

В информатике различают два понятия «данные» и «информация». Данные представляют собой информацию, находящуюся в формализованном виде и предназначенную для обработки техническими системами. Под информацией понимается совокупность представляющих интерес фактов, событий, явлений, которые необходимо зарегистрировать и обработать. Информация в отличие от данных – это то, что нам интересно, что можно хранить, накапливать, применять и передавать. Данные только хранятся, а не используются. Но как только данные начинают использоваться, то они преобразуются в информацию. В процессе обработки информация изменяется по структуре и форме. Признаками структуры является взаимосвязь элементов информации. Структура информации классифицируется на формальную и содержательную. Формальная структура информации ориентирована на форму представления информации, а содержательная – на содержание.

Виды форм представления информации:

1. По способу отображения: символьная (знаки, цифры, буквы); графическая (изображения); текстовая (набор букв, цифр) и звуковая.
2. По месту появления: внутренняя (выходная) и внешняя (входная)
3. По стабильности: постоянная и переменная
4. По стадии обработки: первичная и вторичная.

Теперь можно дать более конкретное определение данных на машинном уровне представления информации. Независимо от содержания и сложности любые данные в памяти ЭВМ представляются последовательностью двоичных разрядов, или битов, а их значениями являются соответствующие двоичные числа. Для человека описывать и исследовать сколько-нибудь сложные данные в терминах последовательностей битов весьма неудобно. Более крупные и содержательные, нежели бит, "строительные блоки" для организации произвольных данных получаются на основе понятия "структуры данного".

# ****Классификация структур данных****

Структуры данных служат материалами, из которых строятся программы. Как правило, данные имеют форму чисел, букв, текстов, символов и более сложных структур типа последовательностей, списков и деревьев.

Для точного описания абстрактных структур данных и алгоритмов программ используются такие системы формальных обозначений, называемые языками программирования, в которых смысл всякого предложения определится точно и однозначно. Среди средств, представляемых почти всеми языками программирования, имеется возможность ссылаться на элемент данных, пользуясь присвоенным ему именем. Выбор правильного представления данных служит ключом к удачному программированию и может в большей степени сказываться на производительности программы, чем детали используемого алгоритма. Вряд ли когда-нибудь появится общая теория выбора структур данных.

Под ***структурой данных*** в общем случае понимают множество элементов данных и множество связей между ними. Такое определение охватывает все возможные подходы к структуризации данных, но в каждой конкретной задаче используются те или иные его аспекты. Поэтому вводится дополнительная классификация структур данных, направления которой соответствуют различным аспектам их рассмотрения. Прежде чем приступать к изучению конкретных структур данных, дадим их общую классификацию по нескольким признакам.

***Физическая структура данных*** отражает способ физического представления данных в памяти машины и называется еще *структурой хранения*, *внутренней структурой* или *структурой памяти*.

Рассмотрение структуры данных без учета её представления в машинной памяти называется ***абстрактной*** или ***логической*** ***структурой***. В общем случае между логической и соответствующей ей физической структурами существует различие, степень которого зависит от самой структуры и особенностей той среды, в которой она должна быть отражена. Вследствие этого различия существуют процедуры, осуществляющие отображение логической структуры в физическую и наоборот. Эти процедуры обеспечивают доступ к физическим структурам и выполнение над ними различных операций.

Различаются простые (базовые, примитивные) структуры (типы) данных и интегрированные (структурированные, композитные, сложные). ***Простыми*** называются такие структуры данных, которые не могут быть расчленены на составные части, большие, чем биты. ***Интегрированными*** называются такие структуры данных, составными частями которых являются другие структуры данных - простые или в свою очередь интегрированные. Интегрированные структуры данных конструируются программистом с использованием средств интеграции данных, предоставляемых языками программирования.

В зависимости от отсутствия или наличия явно заданных связей между элементами данных следует различать ***несвязные структуры*** (векторы, массивы, строки, стеки, очереди) и ***связные структуры*** (связные списки).

Весьма важный признак структуры данных - её изменчивость - изменение числа элементов или связей между элементами структуры. По признаку изменчивости различают структуры ***статические***, ***полустатические*** и ***динамические***. Классификация структур данных по признаку изменчивости приведена на рис. 1.



**Рис. 1.** Классификация структур данных

Базовые структуры данных, статические, полустатические и динамические характерны для оперативной памяти и часто называются ***оперативными структурами***. Файловые структуры соответствуют структурам данных для внешней памяти.

Второй важный признак структуры данных - характер упорядоченности её элементов. По этому признаку структуры можно делить на ***линейные и нелинейные*** ***структуры***. В зависимости от характера взаимного расположения элементов в памяти, линейные структуры можно разделить на *структуры с последовательным распределением элементов в памяти* (векторы, строки, массивы, стеки, очереди) и *структуры с произвольным связным распределением элементов в памяти* (односвязные, двусвязные списки). Пример нелинейных структур - многосвязные списки, деревья, графы.

В языках программирования понятие "структуры данных" тесно связано с понятием "типы данных". Любые данные, т.е. константы, переменные, значения функций или выражения, характеризуются своими типами.

Информация по каждому типу однозначно определяет:

1) структуру хранения данных указанного типа, т.е. выделение памяти и представление данных в ней, с одной стороны, и интерпретирование двоичного представления, с другой;

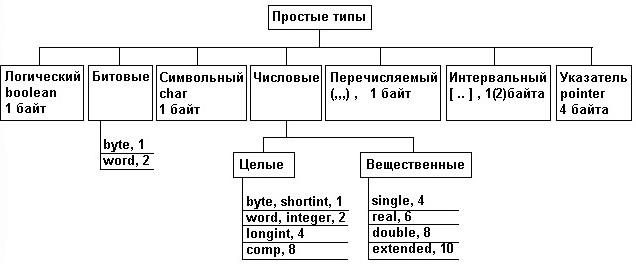
2) множество допустимых значений, которые может иметь тот или иной объект описываемого типа;

3) множество допустимых операций, которые применимы к объекту описываемого типа.

# Основные структуры данных

## Простые структуры данных

***Простые структуры данных*** называют также примитивными или базовыми структурами. Эти структуры служат основой для построения более сложных структур. В языках программирования простые структуры описываются простыми (базовыми) типами. К таким типам относятся: *числовые, битовые, логические, символьные, перечисляемые, интервальные и указатели*. В дальнейшем изложении мы будем ориентироваться на язык PASCAL. Структура простых типов PASCAL приведена на **рис 2.1** (через запятую указан размер памяти в байтах, требуемый для размещения данных соответствующего типа). В других языках программирования набор простых типов может несколько отличаться от указанного.



**Рис. 2.** Структура простых типов PASCAL

### Числовые типы

***Целые типы.*** С помощью целых чисел может быть представлено количество объектов, являющихся дискретными по своей природе (т.е. счетное число объектов).

***Вещественные типы.*** В отличии от порядковых типов (все целые, символьный, логический), значения которых всегда сопоставляются с рядом целых чисел и, следовательно, представляются в памяти машины абсолютно точно, значение вещественных типов определяет число лишь с некоторой конечной точностью, зависящей от внутреннего формата вещественного числа.

### Битовые типы

В ряде задач может потребоваться работа с отдельными двоичными разрядами данных. Чаще всего такие задачи возникают в системном программировании, когда, например, отдельный разряд связан с состоянием отдельного аппаратного переключателя или отдельной шины передачи данных и т.п. Данные такого типа представляются в виде набора битов, упакованных в байты или слова, и не связанных друг с другом. Операции над такими данными обеспечивают доступ к выбранному биту данного.

### Логические типы

Значениями логического типа BOOLEAN может быть одна из предварительно объявленных констант false (ложь) или true (истина). Данные логического типа занимают один байт памяти. При этом значению false соответствует нулевое значение байта, а значению true соответствует любое ненулевое значение байта. Над логическими типами возможны операции булевой алгебры - НЕ (not), ИЛИ (or), И (and), исключающее ИЛИ (xor) - последняя реализована для логического типа не во всех языках. В этих операциях операнды логического типа рассматриваются как единое целое - вне зависимости от битового состава их внутреннего представления. Результаты логического типа получаются при сравнении данных любых типов.

### Символьный тип

Значением символьного типа char являются символы из некоторого предопределенного множества. В большинстве современных персональных ЭВМ этим множеством является ASCII (American Standard Code for Information Intechange - американский стандартный код для обмена информацией). Это множество состоит из 256 разных символов, упорядоченных определенным образом и содержит символы заглавных и строчных букв, цифр и других символов, включая специальные управляющие символы. Допускается некоторые отклонения от стандарта ASCII, в частности, при наличии соответствующей системной поддержки это множество может содержать буквы русского алфавита. Значение символьного типа char занимает в памяти 1 байт. Код от 0 до 255 в этом байте задает один из 256 возможных символов ASCII таблицы.

### Перечислимый тип

Перечислимый тип представляет собой упорядоченный тип данных, определяемый программистом, т.е. программист перечисляет все значения, которые может принимать переменная этого типа. Значения являются неповторяющимися в пределах программы идентификаторами, количество которых не может быть больше 256.

### Интервальный тип

Один из способов образования новых типов из уже существующих - ограничение допустимого диапазона значений некоторого стандартного скалярного типа или рамок описанного перечислимого типа. Определяется заданием минимального и максимального значений диапазона. При этом изменяется диапазон допустимых значений по отношению к базовому типу, но представление в памяти полностью соответствует базовому типу.

### Указатели

Тип указателя представляет собой адрес ячейки памяти. При программировании на низком уровне - в машинных кодах, на языке Ассемблера и на языке C, который специально ориентирован на системных программистов, работа с адресами составляет значительную часть программных кодов. При решении прикладных задач с использованием языков высокого уровня наиболее частые случаи, когда программисту могут понадобиться указатели:

1) При необходимости представить одни и те же физические данные, как данные разной логической структуры.

2) При работе с динамическими структурами данных.

## Статические структуры данных

Статические структуры относятся к разряду непримитивных структур, которые представляют собой структурированное множество примитивных, базовых, структур. Например, вектор может быть представлен упорядоченным множеством чисел. Поскольку по определению статические структуры отличаются отсутствием изменчивости, память для них выделяется автоматически - как правило, на этапе компиляции или при выполнении - в момент активизации того программного блока, в котором они описаны. Выделение памяти на этапе компиляции является столь удобным свойством статических структур, что в ряде задач программисты используют их для представления объектов, обладающих изменчивостью. Например, когда размер массива неизвестен заранее, для него резервируется максимально возможный размер.

Статические структуры в языках программирования связаны со структурированными типами. Структурированные типы в языках программирования являются теми средствами интеграции, которые позволяют строить структуры данных сколь угодно большой сложности. К таким типам относятся: массивы, записи (в некоторых языках - структуры) и множества (этот тип реализован не во всех языках).

### Векторы

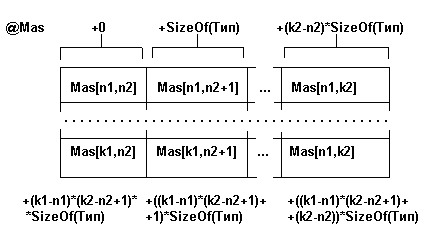
Вектор (одномерный массив) - структура данных с фиксированным числом элементов одного и того же типа. Каждый элемент вектора имеет уникальный в рамках заданного вектора номер. Обращение к элементу вектора выполняется по имени вектора и номеру требуемого элемента.

### Массивы

**Логическая структура.**

Массив - такая структура данных, которая характеризуется: фиксированным набором элементов одного и того же типа; каждый элемент имеет уникальный набор значений индексов; количество индексов определяют мерность массива (два индекса - двумерный массив, три индекса - трехмерный массив, один индекс - одномерный массив или вектор); обращение к элементу массива выполняется по имени массива и значениям индексов для данного элемента. Другими словами, массив - это вектор, каждый элемент которого - вектор.

**Физическая структура** - это размещение элементов массива в памяти ЭВМ. Для случая двумерного массива, состоящего из *(k1-n1+1)* строк и *(k2-n2+1)* столбцов физическая структура представлена на **рис. 3.**



**Рис. 3.** Физическая структура двумерного массива из (k1-n1+1) строк и (k2-n2+1) столбцов

**Многомерные массивы** хранятся в непрерывной области памяти. Размер слота определяется базовым типом элемента массива. Количество элементов массива и размер слота определяют размер памяти для хранения массива. Принцип распределения элементов массива в памяти определен языком программирования. Так в FORTRAN элементы распределяются по столбцам - так, что быстрее меняется левые индексы, в PASCAL - по строкам - изменение индексов выполняется в направлении справа налево.

**Специальные массивы.** На практике встречаются массивы, которые в силу определенных причин могут записываться в память не полностью, а частично. Это особенно актуально для массивов настолько больших размеров, что для их хранения в полном объёме памяти может быть недостаточно. К таким массивам относятся ***симметричные*** и ***разреженные*** ***массивы***.

**Симметричные массивы.** Двумерный массив, в котором количество строк равно количеству столбцов называется квадратной матрицей. Квадратная матрица, у которой элементы, расположенные симметрично относительно главной диагонали, попарно равны друг другу, называется симметричной.

**Разреженные массивы.** Разреженный массив - массив, большинство элементов которого равны между собой, так что хранить в памяти достаточно лишь небольшое число значений отличных от основного (фонового) значения остальных элементов. Различают два типа разреженных массивов:

1) массивы, в которых местоположения элементов со значениями отличными от фонового, могут быть математически описаны;

2) массивы со случайным расположением элементов.

### Множества

**Логическая структура:** Множество - такая структура, которая представляет собой набор неповторяющихся данных одного и того же типа. Множество может принимать все значения базового типа. Базовый тип не должен превышать 256 возможных значений. Поэтому базовым типом множества могут быть byte, char и производные от них типы.

**Физическая структура:** Множество в памяти хранится как массив битов, в котором каждый бит указывает является ли элемент принадлежащим объявленному множеству или нет. Т.о. максимальное число элементов множества 256, а данные типа множество могут занимать не более 32-ух байт.

**Числовые множества.** Стандартный числовой тип, который может быть базовым для формирования множества - тип byte.

**Символьные множества.** Символьные множества хранятся в памяти также как и числовые множества. Разница лишь в том, что хранятся не числа, а коды ASCII символов.

**Множество из элементов перечислимого типа.** Множество, базовым типом которого есть перечислимый тип, хранится также, как множество, базовым типом которого является тип byte. Но, в памяти занимает место, которое зависит от количества элементов в перечислимом типе.

**Множество от интервального типа.** Множество, базовым типом которого есть интервальный тип, хранится также, как множество, базовым типом которого является тип byte. В памяти занимает место, которое зависит от количества элементов, входящих в объявленный интервал.

### Записи

Запись - конечное упорядоченное множество полей, характеризующихся различным типом данных. Полями записи могут быть интегрированные структуры данных - векторы, массивы, другие записи. Записи являются чрезвычайно удобным средством для представления программных моделей реальных объектов предметной области, как правило, каждый такой объект обладает набором свойств, характеризуемых данными различных типов.

### Таблицы

Таблица - сложная интегрированная структура. С физической точки зрения таблица представляет собой вектор, элементами которого являются записи. Логической особенностью таблиц является то, что доступ к элементам таблицы производится не по номеру (индексу), а по ключу - по значению одного из свойств объекта, описываемого записью-элементом таблицы. ***Ключ*** - это свойство, идентифицирующее данную запись во множестве однотипных записей. Ключ может включаться в состав записи и быть одним из её полей, но может и не включаться в запись, а вычисляться по положению записи. Таблица может иметь один или несколько ключей. Основной операцией при работе с таблицами является операция доступа к записи по ключу. Она реализуется процедурой поиска. Поскольку поиск может быть значительно более эффективным в таблицах, упорядоченных по значениям ключей, довольно часто над таблицами необходимо выполнять операции сортировки.

Иногда различают таблицы с фиксированной и с переменной длиной записи. Очевидно, что таблицы, объединяющие записи совершенно идентичных типов, будут иметь фиксированные длины записей. Таблицы с записями переменной длины обрабатываются только последовательно - в порядке возрастания номеров записей.

## Полустатические структуры данных

Полустатические структуры данных характеризуются следующими признаками: они имеют переменную длину и простые процедуры её изменения; изменение длины структуры происходит в определенных пределах, не превышая какого-то максимального (предельного) значения.

Если полустатическую структуру рассматривать на логическом уровне, то о ней можно сказать, что это последовательность данных, связанная отношениями линейного списка. Доступ к элементу может осуществляться по его порядковому номеру. Физическое представление полустатических структур данных в памяти это обычно последовательность слотов в памяти, где каждый следующий элемент расположен в памяти в следующем слоте (т.е. вектор). Физическое представление может иметь также вид однонаправленного связного списка (цепочки), где каждый следующий элемент адресуется указателем находящемся в текущем элементе.

### Стеки

Стек - такой последовательный список с переменной длиной, включение и исключение элементов из которого выполняются только с одной стороны списка, называемого вершиной стека. Применяются и другие названия стека - магазин и очередь, функционирующая по принципу LIFO (Last - In - First- Out - "последним пришёл - первым вышел").

Основные операции над стеком - включение нового и исключение элемента из стека. Полезными могут быть также вспомогательные операции: определение текущего числа элементов в стеке и очистка стека. Некоторые авторы рассматривают также операции включения/исключения элементов для середины стека, однако структура, для которой возможны такие операции, не соответствует стеку по определению.

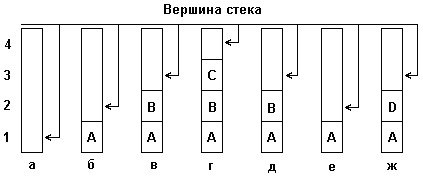
Для наглядности рассмотрим небольшой пример, демонстрирующий принцип включения элементов в стек и исключения элементов из стека. На рис. 4.1 (а, б ,с) изображены состояния стека:

а) пустого;

б, в, г) после последовательного включения в него элементов с именами 'A', 'B', 'C';

д, е) после последовательного удаления из стека элементов 'C' и 'B';

ж) после включения в стек элемента 'D'.



**Рис. 4.** Включение и исключение элементов из стека

Как видно из **рис.** **4**, стек можно представить, например, в виде стопки книг (элементов), лежащей на столе. Присвоим каждой книге свое название, например A,B,C,D... Тогда в момент времени, когда на столе книг нет, про стек аналогично можно сказать, что он пуст, т.е. не содержит ни одного элемента. Если же мы начнем последовательно класть книги одну на другую, то получим стопку книг (допустим, из *n* книг), или получим стек, в котором содержится *n* элементов, причем вершиной его будет являться элемент *n+1*. Удаление элементов из стека осуществляется аналогичным образом т. е. удаляется последовательно по одному элементу, начиная с вершины, или по одной книге из стопки.

### Очереди FIFO

Очередью FIFO (First - In - First- Out - "первым пришел - первым вышел") называется такой последовательный список с переменной длиной, в котором включение элементов выполняется только с одной стороны списка (эту сторону часто называют концом или хвостом очереди), а исключение - с другой стороны (называемой началом или головой очереди).

Основные операции над очередью - те же, что и над стеком - включение, исключение, определение размера, очистка, неразрушающее чтение.

### Деки

Дек - особый вид очереди. Дек (от англ. deq - double ended queue, т.е очередь с двумя концами) - это такой последовательный список, в котором как включение, так и исключение элементов может осуществляться с любого из двух концов списка. Частный случай дека - дек с ограниченным входом и дек с ограниченным выходом. Логическая и физическая структуры дека аналогичны логической и физической структуре кольцевой FIFO-очереди. Однако, применительно к деку целесообразно говорить не о начале и конце, а о левом и правом конце.

Операции над деком: включение элемента справа, слева; исключение элемента справа, слева; определение размера; очистка.

### Строки

Строка - это линейно упорядоченная последовательность символов, принадлежащих конечному множеству символов, называемому алфавитом. Строки обладают следующими важными свойствами: их длина, как правило, переменна, хотя алфавит фиксирован; обычно обращение к символам строки идёт с какого-нибудь одного конца последовательности, т.е важна упорядоченность этой последовательности, а не её индексация; в связи с этим свойством строки часто называют также цепочками; чаще всего целью доступа к строке является не отдельный её элемент (хотя это тоже не исключается), а некоторая цепочка символов в строке.

Говоря о строках, обычно имеют в виду текстовые строки - строки, состоящие из символов, входящих в алфавит какого-либо выбранного языка, цифр, знаков препинания и других служебных символов.

Базовыми операциями над строками являются: определение длины строки; присваивание строк; конкатенация (сцепление) строк; выделение подстроки; поиск вхождения.

Операция определения длины строки имеет вид функции, возвращаемое значение которой - целое число - текущее число символов в строке. Операция присваивания имеет тот же смысл, что и для других типов данных.

## Динамические структуры данных

Динамические структуры по определению характеризуются отсутствием физической смежности элементов структуры в памяти непостоянством и непредсказуемостью размера (числа элементов) структуры в процессе её обработки. Для установления связи между элементами динамической структуры используются указатели, через которые устанавливаются явные связи между элементами. Такое представление данных в памяти называется ***связным***. Элемент динамической структуры состоит из двух полей:

1. информационного поля или поля данных, в котором содержатся те данные, ради которых и создается структура; в общем случае информационное поле само является интегрированной структурой - вектором, массивом, записью и т.п.;
2. поле связок, в котором содержатся один или несколько указателей, связывающий данный элемент с другими элементами структуры;

Когда связное представление данных используется для решения прикладной задачи, для конечного пользователя "видимым" делается только содержимое информационного поля, а поле связок используется только программистом-разработчиком.

Достоинства связного представления данных: в возможности обеспечения значительной изменчивости структур; размер структуры ограничивается только доступным объёмом машинной памяти; при изменении логической последовательности элементов структуры требуется не перемещение данных в памяти, а только коррекция указателей.

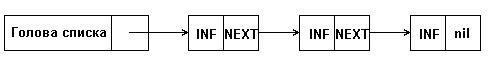
Вместе с тем связное представление не лишено и недостатков: работа с указателями требует, как правило, более высокой квалификации от программиста; на поля связок расходуется дополнительная память; доступ к элементам связной структуры может быть менее эффективным по времени.

Последний недостаток является наиболее серьёзным и именно им ограничивается применимость связного представления данных.

### Связные линейные списки.

Списком называется упорядоченное множество, состоящее из переменного числа элементов, к которым применимы операции включения, исключения. Список, отражающий отношения соседства между элементами, называется ***линейным***. Если ограничения на длину списка не допускаются, то список представляется в памяти в виде связной структуры. Линейные связные списки являются простейшими динамическими структурами данных. Графически связи в списках удобно изображать с помощью стрелок.

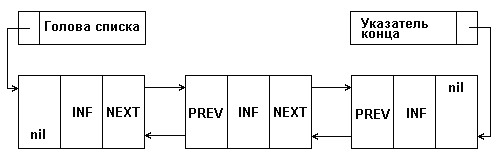
На **рис. 5.1** приведена структура односвязного списка. На нем поле INF - информационное поле, данные, NEXT - указатель на следующий элемент списка. Каждый список должен иметь особый элемент, называемый указателем начала списка, который по формату отличен от остальных элементов. В поле указателя последнего элемента списка находится специальный признак *nil*, свидетельствующий о конце списка.



**Рис. 5.1.** Структура односвязного списка

Но, обработка односвязного списка не всегда удобна, так как отсутствует возможность продвижения в противоположную сторону. Такую возможность обеспечивает ***двухсвязный список***, каждый элемент которого содержит два указателя: на следующий и предыдущий элементы списка. Структура линейного двухсвязного списка приведена на рис. 5.2, где поле NEXT - указатель на следующий элемент, поле PREV - указатель на предыдущий элемент. В крайних элементах соответствующие указатели должны содержать *nil*, как и показано на **рис. 5.2.**

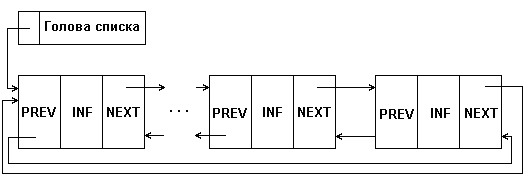
Для удобства обработки списка добавляют еще один особый элемент - указатель конца списка. Наличие двух указателей в каждом элементе усложняет список и приводит к дополнительным затратам памяти, но в то же время обеспечивает более эффективное выполнение некоторых операций над списком.



**Рис. 5.2.** Структура двухсвязного списка

Разновидностью рассмотренных видов линейных списков является кольцевой список, который может быть организован на основе как односвязного, так и двухсвязного списков. При этом в односвязном списке указатель последнего элемента должен указывать на первый элемент; в двухсвязном списке в первом и последнем элементах соответствующие указатели переопределяются, как показано на **рис.5.3.**

При работе с такими списками несколько упрощаются некоторые процедуры, выполняемые над списком. Однако, при просмотре такого списка следует принять некоторых мер предосторожности, чтобы не попасть в бесконечный цикл.



**Рис. 5.3.** Структура кольцевого двухсвязного списка

Линейные списки находят широкое применение в приложениях, где непредсказуемы требования на размер памяти, необходимой для хранения данных; большое число сложных операций над данными, особенно включений и исключений. На базе линейных списков могут строится стеки, очереди и деки. Представление очереди с помощью линейного списка позволяет достаточно просто обеспечить любые желаемые дисциплины обслуживания очереди. Особенно это удобно, когда число элементов в очереди трудно предсказуемо.

### Нелинейные разветвленные списки

Нелинейным разветвленным списком является список, элементами которого могут быть тоже списки. Если один из указателей каждого элемента списка задает порядок обратный к порядку, устанавливаемому другим указателем, то такой двусвязный список будет линейным. Если же один из указателей задает порядок произвольного вида, не являющийся обратным по отношению к порядку, устанавливаемому другим указателем, то такой список будет нелинейным.

В обработке нелинейный список определяется как любая последовательность атомов и списков (подсписков), где в качестве атома берется любой объект, который при обработке отличается от списка тем, что он структурно неделим.

## Нелинейные структуры данных

### Графы

Граф - это сложная нелинейная многосвязная динамическая структура, отображающая свойства и связи сложного объекта.

Многосвязная структура обладает следующими свойствами: на каждый элемент (узел, вершину) может быть произвольное количество ссылок; каждый элемент может иметь связь с любым количеством других элементов; каждая связка (ребро, дуга) может иметь направление и вес.

В узлах графа содержится информация об элементах объекта. Связи между узлами задаются ребрами графа. Ребра графа могут иметь направленность, показываемую стрелками, тогда они называются ***ориентированными***, ребра без стрелок - ***неориентированные***.

Граф, все связи которого ориентированные, называется ***ориентированным графом*** или ***орграфом***; граф со всеми неориентированными связями - ***неориентированным графом***; граф со связями обоих типов - ***смешанным графом***. Примеры изображений графов даны на **рис. 6**.

а). ((A,B),(B,A)) б). (< A,B >,< B,A >).



**Рис.6.** Граф неориентированный (а) и ориентированный (б).

Для ориентированного графа число ребер, входящих в узел, называется *полустепенью* *захода* узла, выходящих из узла - *полустепенью исхода*. Количество входящих и выходящих ребер может быть любым, в том числе и нулевым. Граф без ребер является ***нуль-графом***. Если ребрам графа соответствуют некоторые значения, то граф и ребра называются *взвешенными*. ***Мультиграфом*** называется граф, имеющий параллельные (соединяющие одни и те же вершины) ребра, в противном случае граф называется *простым*.

Путь в графе - это последовательность узлов, связанных ребрами; элементарным называется путь, в котором все ребра различны, простым называется путь, в котором все вершины различны. Путь от узла к самому себе называется циклом, а граф, содержащий такие пути - ***циклическим***.

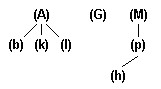
Два узла графа смежны, если существует путь от одного из них до другого. Узел называется инцидентным к ребру, если он является его вершиной, т.е. ребро направлено к этому узлу.

Логически структура-граф может быть представлена матрицей смежности или матрицей инцидентности.

### Деревья

Дерево - это граф, который характеризуется следующими свойствами:

1. Существует единственный элемент (узел или вершина), на который не ссылается никакой другой элемент - и который называется ***корнем*** (рис. 7; 8 - *A, G, M* - корни).
2. Начиная с корня и следуя по определенной цепочке указателей, содержащихся в элементах, можно осуществить доступ к любому элементу структуры.
3. На каждый элемент, кроме корня, имеется единственная ссылка, т.е. каждый элемент адресуется единственным указателем.



**рис. 7.** Дерево **рис. 8.** Лес

Название "дерево" проистекает из логической эквивалентности древовидной структуры абстрактному дереву в теории графов. Линия связи между парой узлов дерева называется обычно ***ветвью***. Те узлы, которые не ссылаются ни на какие другие узлы дерева, называются ***листьями*** или терминальными вершинами (рис. 7; 8 - *b, k, l, h* - листья). Узел, не являющийся листом или корнем, считается промежуточным или ***узлом ветвления*** (нетерминальной или внутренней вершиной).

Деревья нужны для описания любой структуры с иерархией. Традиционные примеры таких структур: генеалогические деревья, десятичная классификация книг в библиотеках, иерархия должностей в организации и т. д.

***Ориентированное дере***во - это такой ациклический орграф (ориентированный граф), у которого одна вершина, называемая корнем, имеет полустепень захода, равную 0, а остальные - полустепени захода, равные 1. Ориентированное дерево должно иметь по крайней мере одну вершину. Изолированная вершина также представляет собой ориентированное дерево.

Вершина ориентированного дерева, полустепень исхода которой равна нулю, называется ***концевой*** (висячей) ***вершиной*** или листом; все остальные вершины дерева называют вершинами ветвления. Длина пути от корня до некоторой вершины называется ***уровнем*** (номером яруса) этой вершины. Уровень корня ориентированного дерева равен нулю, а уровень любой другой вершины равен расстоянию (т.е. модулю разности номеров уровней вершин) между этой вершиной и корнем. Ориентированное дерево является ациклическим графом, все пути в нем элементарны.

# Заключение

Структуры данных – это абстрактные структуры или классы, которые используются для организации данных и предоставляют различные операции над этими данными.

В данной работе были описаны структуры данных всех классов памяти ЭВМ: простых, статических, полустатических, динамических и нелинейных, а также, информация о возможных операциях над всеми перечисленными структурами. С помощью операций можно организовывать поиск, сортировку и редактиро­вание данных. Способы анализа эффективности использования структур данных важны для подсчета времени исполнения различных операций структур данных и являются полезным инструментом при выборе той или иной структуры данных для конкретной программистской задачи.

Мы рассмотрели вопрос о важности структур данных и о том, как они влияют на эффективность алгоритмов. Выбор правильного представления данных служит ключом к удачному программированию и может в большей степени сказываться на производительности программы, чем детали используемого алгоритма. Для определения того, как структуры данных влияют на производительность программ, нужно рассмотреть, как можно строго проанализировать различные операции, выполняемые структурами данных. Вряд ли когда-нибудь появится общая теория выбора структур данных.

Стоит добавить, что совокупность структур данных и операций их обработки составляет модель данных, которая является ядром любой базы данных. Модель данных представляет собой множество структур данных, ограничений целостности и операций манипулирования данными. С помощью модели данных могут быть представлены объекты предметной области и взаимосвязи между ними. База данных основывается на использовании иерархической, сетевой или реляционной модели, на комбинации этих моделей или на некотором их подмножестве.

# ПРАКТИЧЕСКАЯ ЧАСТЬ

## Общая характеристика задачи

Рассмотрим следующую задачу.

**Вариант 21**

1. Построить таблицы по приведенным данным о доходах членов семьи (рис. 21.1, 21.2) и о расходах семьи (рис. 21.3) за квартал.
2. Заполнить таблицу на рис. 21.4 числовыми данными о доходах семьи за квартал, выполнив консолидацию по расположению данных.
3. Составить таблицу планирования бюджета семьи на квартал (рис. 21.5).
4. По данным о бюджете семьи на квартал (рис. 21.5) построить гистограмму.

**Доходы Чижовой М.А. за 1 квартал 2006 г., руб.**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Наименование доходов** | **Сентябрь** | **Октябрь** | **Ноябрь** | **Декабрь** |
| Зарплата | 4000 | 3000 | 2200 | 3200 |
| Прочие поступления | - | 500 | - | 1000 |
| Сумма дохода в месяц |  |  |  |  |

**Рис. 21.1.** Доходы Чижовой М.А. за квартал

**Доходы Чижова А.С. за 1 квартал 2006 г., руб.**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Наименование доходов** | **Сентябрь** | **Октябрь** | **Ноябрь** | **Декабрь** |
| Зарплата | 7000 | 7000 | 7500 | 7400 |
| Прочие поступления | 1200 | 500 | 500 | 1000 |
| Сумма дохода в месяц |  |  |  |  |

**Рис. 21.2.** Доходы Чижова А.С. за квартал

**Расходы семьи Чижовых за 1 квартал 2006 г., руб.**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Наименование расходов** | **Сентябрь** | **Октябрь** | **Ноябрь** | **Декабрь** |
| Коммунальные платежи | 630 | 670 | 700 | 800 |
| Оплата электроэнергии | 100 | 100 | 120 | 120 |
| Оплата телефонных счетов | 195 | 195 | 195 | 195 |
| Расходы на питание | 2500 | 2500 | 2600 | 3000 |
| Прочие расходы | 1000 | 1000 | 1500 | 2000 |
| Погашение кредита | 4000 | 4000 | 4000 | 4000 |
| Суммарный расход в месяц |  |  |  |  |

**Рис. 21.3.** Расходы семьи Чижовых за квартал

**Доходы семьи Чижовых за 1 квартал 2006 г.**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Наименование доходов** | **Сентябрь** | **Октябрь** | **Ноябрь** | **Декабрь** |
| Зарплата |  |  |  |  |
| Прочие поступления |  |  |  |  |
| Суммарный доход в месяц |  |  |  |  |

**Рис. 21.4.** Доходы семьи Чижовых за квартал

**Бюджет семьи Чижовых за 1 квартал 2006 г.**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Наименование** | **Сентябрь** | **Октябрь** | **Ноябрь** | **Декабрь** |
| Суммарный доход в месяц |  |  |  |  |
| Суммарный расход в месяц |  |  |  |  |
| Остаток |  |  |  |  |

**Рис. 21.5.** Бюджет семьи Чижовых за квартал

**2. Практическая часть**

***2.1. Общая характеристика задачи***

Расмотрим следующую задачу:

1. Построить таблицы по приведенным данным о доходах членов семьи (табл. 1, 2) и о расходах семьи (табл. 3) за квартал.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Доходы Чижовой М. А. за 1 квартал 2006 г., руб.** | | | | |
| **Наименование доходов** | **Сентябрь** | **Октябрь** | **Ноябрь** | **Декабрь** |
| Зарплата | 4000 | 3000 | 2200 | 3200 |
| Прочие поступления | - | 500 | - | 1000 |
| Сумма дохода в месяц |  |  |  |  |

**Таблица 1 Доходы Чижовой М. А. за квартал**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Доходы Чижова А. С. за 1 квартал 2006 г., руб.** | | | | |
| **Наименование доходов** | **Сентябрь** | **Октябрь** | **Ноябрь** | **Декабрь** |
| Зарплата | 7200 | 7000 | 7500 | 7400 |
| Прочие поступления | 1200 | 500 | 500 | 1000 |
| Сумма дохода в месяц |  |  |  |  |

**Таблица 2 Доходы Чижова А. С. за квартал**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Расходы семьи Чижовых за 1 квартал 2006 г., руб.** | | | | |
| **Наименование расходов** | **Сентябрь** | **Октябрь** | **Ноябрь** | **Декабрь** |
| Коммунальные платежи | 630 | 670 | 700 | 800 |
| Оплата электроэнергии | 100 | 100 | 120 | 120 |
| Оплата телефонных счетов | 195 | 195 | 195 | 195 |
| Расходы на питание | 2500 | 2500 | 2600 | 3000 |
| Прочие расходы | 1000 | 1000 | 1500 | 2000 |
| Погашение кредита | 4000 | 4000 | 4000 | 4000 |
| Суммарный расход в месяц |  |  |  |  |

**Таблица 3 Расходы семьи Чижовых за квартал**

1. Заполнить таблицу 4 числовыми данными о доходах семьи за квартал, выполнив консалидацию по расположению данных.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Доходы семьи Чижовых за 1 квартал 2006 г., руб.** | | | | |
| **Наименование доходов** | **Сентябрь** | **Октябрь** | **Ноябрь** | **Декабрь** |
| Зарплата |  |  |  |  |
| Прочие поступления |  |  |  |  |
| Сумма дохода в месяц |  |  |  |  |

**Таблица 4 Доходы семьи Чижовых за квартал**

1. Составить таблицу планирования бюджета семьи на квартал (табл. 5).

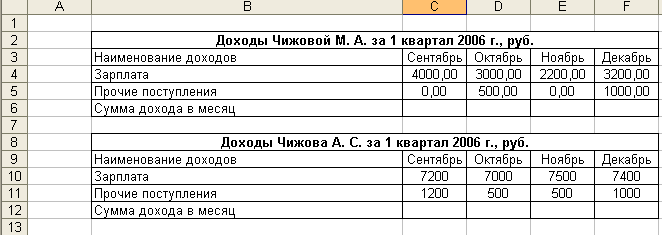
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Бюджет семьи Чижовых за 1 квартал 2006 г., руб.** | | | | |
| **Наименование** | **Сентябрь** | **Октябрь** | **Ноябрь** | **Декабрь** |
| Суммарный доход в месяц |  |  |  |  |
| Суммарный расход в месяц |  |  |  |  |
| Остаток |  |  |  |  |

**Таблица 5 Бюджет семьи Чижовых за квартал**

1. По данным о бюджете семьи на квартал (табл. 5) построить гистограмму.

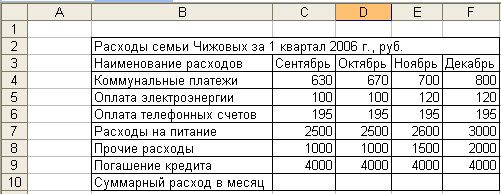
***Описание алгоритма решения задачи:***

1. Запустить табличный процессор MS Excel.
2. Создать книгу с именем «Бюджет».
3. Лист 1 переименовать в лист с названием **Доходы.**
4. На рабочем листе **Доходы** MS Excel создать таблицы «Доходы Чижовой М. А. за квартал» и «Доходы Чижова А. С. за квартал».
5. Заполнить таблицы доходов исходными данными (рис. 2.1).



**Рисунок 2. 1** Расположение таблиц **доходов** на рабочем листе **Доходы** MS Excel

1. Лист 2 переименовать в лист с названием **Расходы.**
2. На рабочем листе **Расходы** MS Excel создать таблицу расходов семьи Чижовых за квартал.
3. Заполнить таблицу расходов исходными данными (рис. 2.2)



**Рисунок 2. 2** Расположение таблиц **расходов** на рабочем листе **Расходы** MS Excel

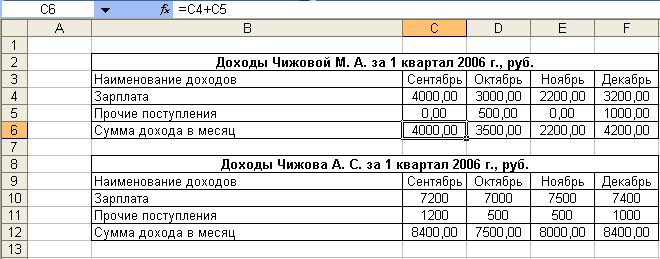
1. Заполнить строку **Сумма дохода в месяц** таблиц «Доходы Чижовой М. А. за 1 квартал 2006 г., руб.» и «Доходы Чижова А. С. за 1 квартал 2006 г., руб.», находящихся на листе **Доходы** следующим образом:

Занести в ячейку С6 формулу:

=С4+С5

Размножить введенную в ячейку С6 формулу для остальных ячеек (с С4 по F4 и с С12 по F12) данных строк.

Таким образом будет автоматически подсчитаны суммы дохода в месяц (рис. 2.3).



**Рисунок 2. 3** Автоматический подсчет суммы доходов в месяц

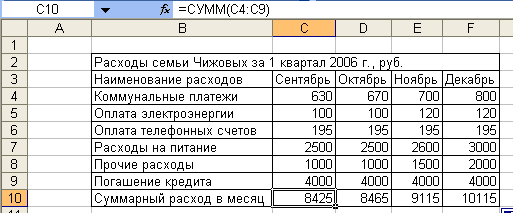
1. Заполнить строку **Суммарный расход в месяц** таблицы «Расходы семьи Чижовых за 1 квартал 2006 г., руб.», находящейся на листе **Расходы** следующим образом:

Занести в ячейку С10 формулу:

=СУММ(C4:C9)

Размножить введенную в ячейку С10 формулу для остальных ячеек (с С10 по F10) данной строки.

Таким образом будет автоматически подсчитана сумма расхода в месяц (рис. 2.4).



**Рисунок 2. 4** Автоматический подсчет суммы расходов в месяц

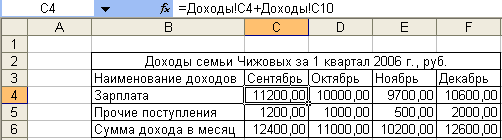
1. Лист 3 переименовать в лист с названием **Итоги.**
2. На рабочем листе **Итоги** MS Excel создать таблицу «Доходы семьи Чижовых за 1 квартал 2006 г., руб.»
3. Заполнить строки **Зарплата, Прочие поступления, Сумма дохода в месяц** таблицы «Доходы семьи Чижовых за 1 квартал 2006 г., руб.», находящейся на листе **Итоги** следующим образом:

Занести в ячейку С4 формулу:

=Доходы!C4+Доходы!C10

Размножить введенную в ячейку С4 формулу для остальных ячеек (с С4 по F6) данной таблицы.

Таким образом будут автоматически подсчитана сумма дохода в месяц (рис. 2.5).



**Рисунок 2. 5** Автоматический подсчет суммы доходов в месяц

1. На рабочем листе **Итоги** MS Excel создать таблицу «Бюджет семьи Чижовых за 1 квартал 2006 г., руб.»
2. Заполнить строку **Суммарный доход в месяц** таблицы «Бюджет семьи Чижовых за 1 квартал 2006 г., руб.», находящейся на листе **Итоги** следующим образом:

Занести в ячейку С11 формулу:

=C6

Размножить введенную в ячейку С11 формулу для остальных ячеек (с С11 по F11) данной таблицы.

1. Заполнить строку **Суммарный расход в месяц** таблицы «Бюджет семьи Чижовых за 1 квартал 2006 г., руб.», находящейся на листе **Итоги** следующим образом:

Занести в ячейку С12 формулу:

=Расходы!C10

Размножить введенную в ячейку С12 формулу для остальных ячеек (с С12 по F12) данной таблицы.

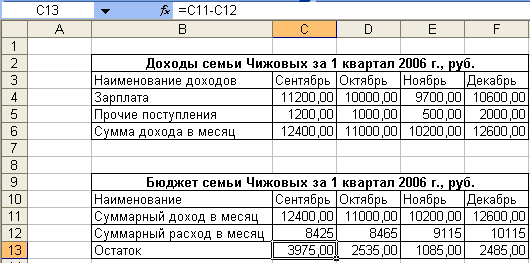
1. Заполнить строку **Остаток** таблицы «Бюджет семьи Чижовых за 1 квартал 2006 г., руб.», находящейся на листе **Итоги** следующим образом:

Занести в ячейку С13 формулу:

=C11-C12

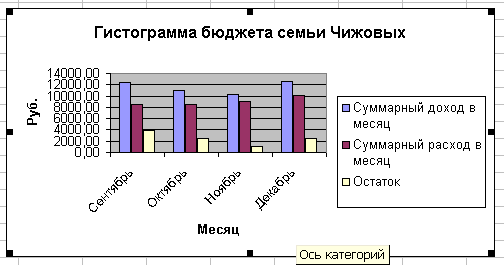
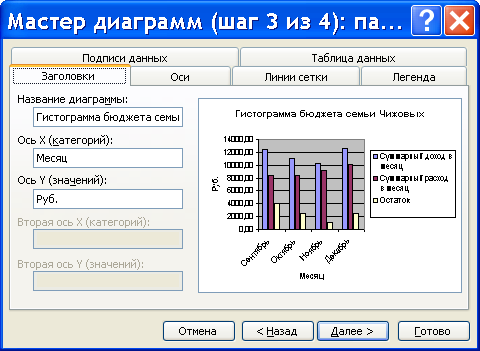
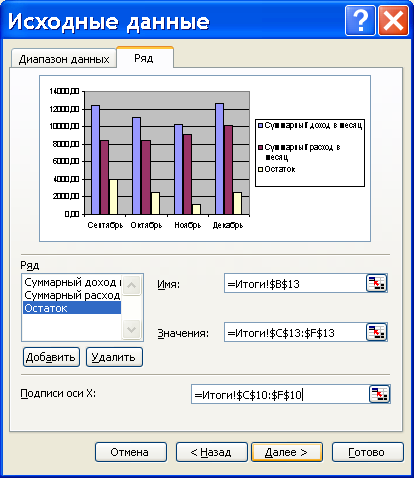
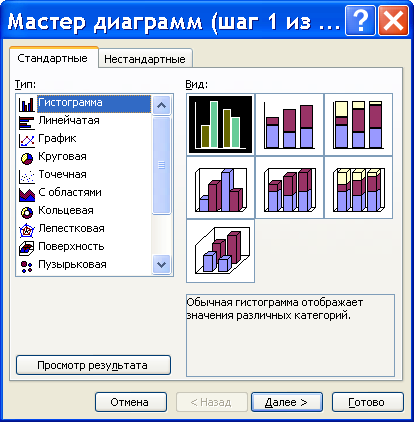
Размножить введенную в ячейку С13 формулу для остальных ячеек (с С13 по F13) данной таблицы.

Таким образом будет заполнена таблица «Бюджет семьи Чижовых за 1 квартал 2006 г., руб.» (рис. 2.6)



**Рисунок 2. 6** Автоматическое заполнение таблицы «**Бюджет семьи Чижовых за 1 квартал 2006 г., руб.**»

1. Лист 4 переименовать в лист с названием **Гистограмма.**
2. По данным таблицы «Бюджет семьи Чижовых за 1 квартал 2006 г., руб.» на листе **Гистограмма** построим гистограмму (рис. 2.7).



**Рисунок 2. 7** Гистограмма

# 

# Список использованной литературы

1. Барон Д. Введение в языки программирования. М.: Мир, 1980.- 190 с.
2. Вирт Н. Алгоритмы и структуры данных./ Пер. с англ. - М.: Мир, 1989. – 360 с.
3. Гуда А. Н., Бутакова М. А., Нечитайло Н. М., Чернов А.В. Информатика. Общий курс: Учебник / Под ред. академика РАН В. И. Колесникова. – М.: Издательско-торговая корпорация «Дашков и К»; Ростов н/Д: Наука-Пресс, 2007. – 400 с.
4. Каймин В. А. Информатика: Учебник. - М.: ИНФРА-М, 2000. - 232 с.
5. Кубенский, А. А.. Структуры и алгоритмы обработки данных. Объектно-ориентированный подход и реализация на С++ / А. А. Кубенский. – Спб: БХВ-Петербург, 2004. – 464 с.
6. Лэгсам Й., Огенстайн М.. Структуры данных для персональных ЭВМ – М: Мир, 1989. – 586 с.
7. Модели и структуры данных В. Д. Далека, А. С. Деревянко, О. Г. Кравец, Л. Е. Тимановская. Учебное пособие. Харьков: ХГПУ, 2000. - 241с.
8. Организация и обработка структур данных в вычислительных систем: Учебное пособие для вузов. Костин А.Е., Шаньгин В.Ф. 1987. - 248 с.
9. Советов Б. Я.. Информационные технологии. Учебник для студентов вузов. 2006. - 263 с.
10. Уоллес Вонг. Основы программирования для «чайников». ДИАЛЕКТИКА. Москва – Санкт-Петербург – Киев. 2001. – 335 с.
11. Хоор К. О структурной организации данных. В сб. "Дал У., Дейкстра Э., Хоор К. Структурное программирование". /Пер. с англ. - М.: Мир, 1975. – 197 с.