Министерство образования и науки Российской Федерации
Федеральное агенство по образованию
Ярославский государственный университет им. П.Г. Демидова

В.В. Васильчиков, Н.С. Лагутина, Ю.А. Ларина

Основы программирования на языке C

Учебное пособие

Рекомендовано
Научно–методическим советом университета
для студентов направления Прикладная математика и информатика и специальностей Прикладная математика и информатика, Математическое обеспечение и администрирование информационных систем, Прикладная информатика (в экономике)

Ярославль 2006

УДК 004.4
ББК В185.2я73+397.2-018я73
В19

Рекомендовано
Редакционно–издательским советом университета
в качестве учебного издания. План 2006 года

Рецензенты:
начальник отдела информатизации образования
департамента образования Ярославской области,
кандидат физико–математических наук С.И. Щукин;
кафедра теории и методики обучения информатике
Ярославского государственного педагогического
университета им. К.Д. Ушинского

Васильчиков,В.В. Основы программирования на языке С:
В19 учебное пособие/В.В. Васильчиков, Н.С. Лагутина, Ю.А. Ларина; Яросл. гос. ун–т. — Ярославль: ЯрГУ, 2006. — 80 с.
ISBN 5–8397–0443–1

Учебное пособие содержит описание базовых конструкций языка С, необходимых для изучения основных методов программирования и набор задач по программированию на языке С.

Предназначено для студентов первого курса факультета информатики и вычислительной техники ЯрГУ, обучающихся по направлению 010500 Прикладная математика и информатика; специальностям 010501 Прикладная математика и информатика, 010503 Математическое обеспечение и администрирование информационных систем, 080801 Прикладная информатика (в экономике); дисциплины «Основы программирования» (ПМИ), «Программирование» (МО), «Информатика и программирование» (ПИЭ), блок ЕН, ОПД; очной формы обучения. Может быть полезно для преподавателей, ведущих лабораторные занятия по программированию на младших курсах.

Рис. 14. Библиогр.: 13 назв.

УДК 004.4
ББК В185.2я73+397.2-018я73

ISBN 5–8397–0443–1

Введение

Учебное пособие предназначено для студентов I курса факультета информатики и вычислительной техники ЯрГУ. Пособие содержит описание ряда элементов и базовых конструкций языка C, набор заданий по программированию, а также требования по выполнению и оформлению заданий. Оно может также быть полезно для всех преподавателей, ведущих лабораторные занятия по программированию на I курсе, а также для тех, кто хочет получить практические навыки в программировании на языке C.

В начале пособия перечислены обязательные требования, которые предъявляются к выполнению, оформлению и тестированию программ.

Материал пособия разбит на шесть разделов, предназначенных для освоения базовых конструкций языка и получения навыков программирования. В начале каждого раздела содержится описание основных понятий и элементов языка C, необходимых для выполнения соответствующих заданий. Некоторые особенности языка проиллюстрированы примерами.

Задания условно разделены на три группы по возрастанию сложности. Умение выполнять задачи из части A определяет минимальный уровень, необходимый для освоения курса «Основы программирования». Успешное написание программ для решения задач из части B соответствует более высокому уровню владения языком C и программированием вообще. В часть C собраны задания повышенной сложности.

Седьмой раздел содержит набор дополнительных заданий различного уровня сложности. Постановка задач в этой части пособия не описана детально, поэтому соответствующее техническое задание необходимо согласовать с преподавателем.

В конце данного учебного пособия имеется список литературы, рекомендуемой для тех, кто хочет научиться программировать на языке C.

Предполагается, что в качестве инструментальной среды при выполнении лабораторных работ будет использоваться turboC версии 2.0 или BolandC версии 3.1. Поэтому в перечень навыков, которые должны быть выработаны студентом для получения зачета, кроме собственно умения программировать на языке C, входит свободное владение соответствующей инструментальной средой и встроенными средствами отладки.

Требования к лабораторным работам

Написание программы на языке С предусматривает выполнение ряда действий, которые можно разделить на следующие важнейшие этапы.

- Постановка задачи.
- Выбор метода решения задачи (разработка алгоритма).
- Написание и ввод текста программы с помощью текстового редактора — получение исходного файла с расширением `.c`.
- Обработка исходного текста программы для получения исполняемого модуля с расширением `.exe` (препроцессорное преобразование текста, компиляция, компоновка) см. рис. 1 на с. 5, на этапе компиляции программистом осуществляется исправление синтаксических ошибок.
- Запуск программы на выполнение.
- Тестирование программы и отладка алгоритма.

В ходе выполнения лабораторных работ необходимо соблюдать ряд требований, обязательных для каждой программы.

1) Требования к программе.

   - Программа должна обеспечивать выполнение всех условий задания.
   - Программа должна контролировать правильность вводимых исходных данных в соответствии с условиями задачи. Например, если в условии отмечено, что программа вводит последовательность идентификаторов, она должна анализировать считываемую цепочку символов и выдавать сообщения о встречаемых ошибках (недопустимый символ или первый символ — цифра).
   - Если в условии задачи нет явного указания на объем обрабатываемых входных данных, то программа должна обеспечивать корректную обработку информации любого объема.
   - Информация, выводимая программой, должна содержать краткое назначение программы, ее автора, исходные данные и получаемый результат. При этом результаты представляются в виде, удобном для восприятия, например, матрицы должны быть представлены в виде таблицы. Если программа вводит какую-то информацию из командной строки, то при запуске ее без аргументов или с непра-

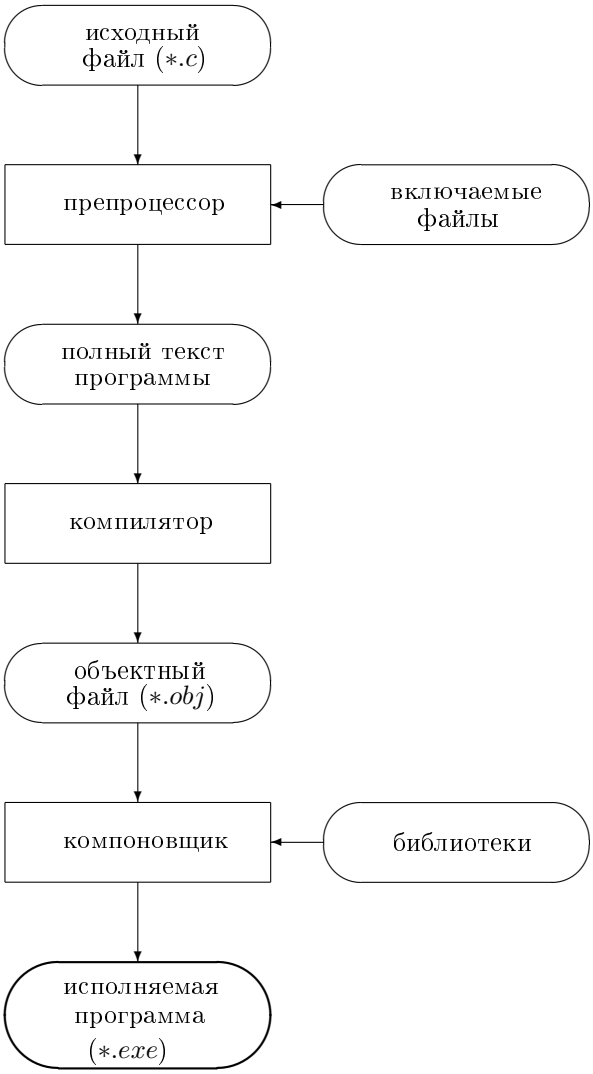

Рис. 1. Этапы формирования исполняемого модуля программы, если исходный код содержится в одном файле.

вильными аргументами она должна вывести сообщение о том, как правильно осуществить вызов.

2) Требования к тексту программы.

- Необходимой частью текста программы являются комментарии. Они должны нести информацию об алгоритме, различных частях программы, возможно, некоторых переменных. Программа начинается комментарием, описывающим ее назначение и автора. Если текст программы состоит из нескольких файлов, то в каждом из них в начале должен быть комментарий, описывающий назначение данного файла. В целом, комментариев в тексте программы должно быть достаточно для понимания работы программы посторонним человеком. При этом комментарии пишутся и корректируются одновременно с самой программой.
- Используемые идентификаторы должны нести информацию о назначении соответствующей переменной, константы, функции и т.д. Не следует злоупотреблять как односимвольными (конечно, если речь не идет о простеньком счетчике цикла или индексе в матрице), так и слишком длинными именами.
- Следует правильно использовать отступы. Текст программы должен начинаться с первого символа строки. Все операторы линейной части программы должны иметь слева одно и то же число пробелов. Нельзя использовать для создания отступа знак табуляции. При определении функции, цикла или условного оператора открывающую и закрывающую фигурные скобки следует располагать на уровне первой буквы первого слова той синтаксической конструкции, к которой они относятся. В случае условного оператора или цикла с одним оператором последний следует располагать либо на одной строке с управляющим оператором, либо на следующей строке с отступом слева от него. Величина отступов разного уровня вложенности относительно предыдущего уровня должна быть одинакова во всех файлах данной программы.
- Каждый оператор программы должен находиться на отдельной строке. Такая программа легче читается и проще отлаживается.

- Каждую фигурную скобку рекомендуется располагать на отдельной строке, за исключением директивы `asm`, которая требует применения открывающей скобки на той же строке, и определений структур данных, в которых открывающую скобку принято располагать на одной строке с описанием структуры.
- Перед знаками "запятая"и "точка с запятой"не следует ставить пробел. Перед открывающей круглой или квадратной скобкой оператора, условия, функции или массива, и внутри этих скобок (после открывающей и перед закрывающей) также не следует ставить пробел, если от этого не ухудшается визуальное восприятие текста. Это не относится к фигурным скобкам составного оператора, структуры или тела функции.
- Для имен макросов не следует использовать строчные буквы, а для всех остальных идентификаторов не следует использовать только прописные буквы.
- Строки исходного текста не должны выходить за пределы экрана.

3) Требования к тестам.

- Тесты должны разрабатываться на стадиях выбора метода решения задания, проектирования программы и ее отладки. При этом необходимо обеспечить полноту набора тестов: каждая ветвь алгоритма, каждый оператор программы должны быть протестированы.
- При разработке тестов должны быть рассмотрены варианты как правильного ввода исходных данных, так и неправильного. При этом должны быть учтены все возможные виды ошибок пользователя, а программа должна в этом случае обеспечить соответствующую защиту (ни в каком случае программа не должна «зависать»). Если возможны какие-либо вырожденные случаи при работе алгоритмы, то набор тестов непременно должен включать в себя варианты для всех таких случаев.
- Разработав тест, необходимо просчитать результат вручную, только после этого можно сделать контрольный запуск программы.

1. Последовательности и одномерные массивы

Основные понятия

В этом разделе дано описание основных элементов языка C, необходимое для создания первых простейших программ. Естественно, для полноценного изучения языка следует использовать и другую литературу, например [1—13]. Часть понятий (некоторые стандартные функции, операторы) проиллюстрирована на примерах, включающих в себя небольшие программы. Чтобы лучше понять эти примеры, необходимо набрать соответствующую программу в системе Borland C++, откомпилировать и запустить на выполнение. Следует обратить внимание на задания к каждому примеру, их выполнение дает возможность лучше узнать и понять особенности изучаемого языка.

При написании программ на языке C используются следующие понятия:

- алфавит;
- идентификаторы;
- ключевые слова;
- комментарии.

Алфавитом языка называется совокупность символов, используемых в языке. Важно отметить, что язык C различает строчные и прописные буквы. Например, идентификаторы `COLOR`, `Color` и `color` определяют три различных имени переменных.

Идентификаторы в языке программирования используются для обозначения имен переменных, функций и меток, применяемых в программе. Идентификатором может быть произвольная последовательность латинских букв (прописных и строчных), цифр и символа подчеркивания, которая начинается с буквы или с символа подчеркивания. В языке C идентификатор может состоять из произвольного количества символов, однако два идентификатора считаются различными, если у них различаются первые 32 символа.

Некоторые идентификаторы употребляются как служебные слова, имеющие специальное значение для компилятора. Их употребление строго определено, и эти слова не могут использоваться иначе. *Ключевыми словами* являются, например, названия типов данных (`int`, `char`, `float`), операторы языка (`if`, `while`, `for`), и т.п.

Комментарий — это часть программы, которая игнорируется компилятором и служит для пояснения исходного текста программы. Комментарием в языке C является любая последовательность символов, заключенная между парами символов `/*` и `*/`, они могут располагаться в любом месте программы, где допустимо использование пробелов.

Каждая программа на языке C состоит из следующих частей:

- препроцессорные директивы;
- описания и определения глобальных объектов;
- функции.

Препроцессорные директивы управляют преобразованием текста программы до ее компиляции. Каждая директива начинается с символа `\#` и располагается, как правило, на отдельной строке. Например, `#define` указывает правила замены в тексте. Директива `#include` определяет, какие текстовые файлы нужно включить в этом месте программы. Использовать директивы препроцессора не обязательно, но обойтись без них трудно.

Описания уведомляют компилятор о свойствах и именах объектов и функций, определенных в других частях программы. Они необходимы в больших программах, состоящих из нескольких функций и использующих данные со сложной структурой. *Определения* вводят объекты, необходимые для представления в программе обрабатываемых данных. Примером таких объектов могут служить переменные разных типов.

Одним из основных понятий языка C является *переменная*. Переменная — это именованная часть памяти, необходимая для хранения и использования данных в программе, именем служит идентификатор. Каждая переменная перед ее использованием в программе должна быть определена. Простейшая форма определения выглядит так:

Тип переменная, переменная,...;

Можно выделить пять основных типов данных в языке C:

- `char` — символьный;
- `int` — целый;
- `float` — вещественный одинарной точности;
- `double` — вещественный удвоенной точности;
- `void` — пустой.

Приведем пример определения двух вещественных переменных `x` и `y`:

```
float x, y;
```

Отметим, что после определения по умолчанию переменные имеют неопределенное значение, поэтому программист должен сам позаботиться о присвоении им начального значения. Это можно сделать в тексте программы с помощью оператора присваивания:

```
x=0.25; y=4.0;
```

или непосредственно в определении (такой прием называется инициализацией):

```
float x=0.25, y=4;
```

Переменной можно присвоить не только конкретное значение (константу), но и допустимое в языке С выражение, т.е. комбинацию переменных, констант и операций.

В арифметических выражениях допустимы следующие операции:

- `+` — сложение;
- `-` — вычитание;
- `*` — умножение;
- `/` — деление, если делимое и делитель — целые числа, то результатом будет целая часть от частного (без округления);
- `%` — деление по модулю, т.е. получение остатка от деления двух чисел (применяется только к целым числам).

Операции умножения, деления и деления с остатком выполняются первыми, затем вычисляется результат умножения и вычитания. Для изменения порядка выполнения операций используются круглые скобки.

Язык С предоставляет пользователю еще две очень полезные операции, специфичные именно для языка С, — это `++` и `--` . Операция `++` прибавляет к операнду единицу, операция `--` вычитает единицу из операнда. Обе операции могут следовать как перед операндом (префиксная форма, при которой значение операнда увеличивается или уменьшается до его использования в выражении), так и после него (постфиксная форма, при которой значение операнда увеличивается или уменьшается после его использования). Три написанные ниже оператора дадут один и тот же результат:

```
int x;
x=x+1; ++x; x++;
```

Основной частью программы на языке С являются функции. *Функция* — это самостоятельная единица программы, созданная для

решения конкретной задачи. Функция состоит из заголовка и тела. В начале заголовка необходимо указать тип возвращаемого значения (тип результата). Если функция ничего не возвращает, то указывается тип `void`. Далее следует имя функции — идентификатор, после него ставятся круглые скобки. В скобках могут находится аргументы, передаваемые функции, если их нет, то скобки оставляют пустыми. После заголовка размещается тело функции — последовательность описаний, определений и исполняемых операторов, заключенная в фигурные скобки. Каждое определение, описание и каждый опреатор завершается символом ';' (точка с запятой).

Среди функций программы всегда должна присутствовать функция с именем `main`, одинаковым для всех программ на языке С. Она является главной, и именно с нее начинается выполнение программы. Если используется только одна функция, исходный текст программы имеет следующий вид:

директивы препроцессора
`void main()`
`{`
определения объектов;
исполняемые операторы;
`}`

В языке С нет специальных операторов, осуществляющих ввод и вывод информации. Для этих целей используются различные готовые, иначе говоря, библиотечные функции. Достаточно часто для вывода информации на экран используется функция `printf()`. Для ввода данных с клавиатуры в программе можно использовать функцию `scanf()`. При этом в обоих случаях у программиста имеется возможность форматировать данные, т.е. влиять на их внешнее представление.

Рассмотрим пример применения этих функций.

Пример 1.1

```
#include<stdio.h>                              /* 1 */
void main()                                    /* 2 */
{                                              /* 3 */
  int x;                                       /* 4 */
  printf("\nВведите целое число\n");             /* 5 */
  scanf("%d",&x);                              /* 6 */
  printf("\n Вы ввели число\t %d",x);          /* 7 */
}                                              /* 8 */
```

Задание

- попробуйте удалить управляющую последовательность `\n` и `\t`;
- вместо `%d` поставьте `%f` или `%c`;
- удалите знак `&` в функции `scanf()`.

Детально поясним эту программу. В первой строке (номер каждой строки указан справа в комментариях) расположена директива препроцессора, подключающая файл `stdio.h`. Этот файл содержит информацию, необходимую для правильного выполнения функций библиотеки стандартного ввода и вывода языка С, в нашем случае это функции `printf()` и `scanf()`. Язык С предусматривает использование файлов такого типа, которые называются заголовочными файлами. Они необходимы для корректного использования библиотечных функций, например математических, функций работы со строками и т.п.

Во второй строке находится заголовок основной и единственной функции `main`.

Третья строка содержит открывающую фигурную скобку, за которой начинается тело функции `main`, состоящее из одного определения и трех операторов.

Далее следует определение целочисленной переменной `x`.

В пятой строке вызывается функция `printf()`. В общем виде оператор вызова этой функции можно представить так:

`printf(`*управляющая строка, список аргументов*`);`

Управляющая строка содержит символы, непосредственно выводимые на экран, управляющие символы (в нашем примере, `\n`) и команды формата. Команды формата используются для вывода значений переменных, каждая команда начинается с символа '%', за которым следует код формата, определяющий вид и тип выводимого значения. Список аргументов содержит переменные, значение которых должно быть выведено на экран. Имена переменных перечисляются через запятую в нужном поряке, они должны соответствовать командам формата, указанным в управляющей строке. Список аргументов может отсутствовать, например, как в рассматриваемом операторе.

Этот оператор предлагает пользователю ввести данные с клавиатуры. В любой программе необходимо в дружелюбной манере, но без излишнего многословия пригласить пользователя ввести исходные данные, указав при этом порядок их ввода и тип вводимой информации.

Ввод данных осуществляется с помощью функции `scanf()` (шестая строка). Общий вид этого оператора следующий:

`scanf(`*управляющая строка, список аргументов*`);`

Эта функция предназначена для ввода данных с клавиатуры. Она преобразует введенные символы во внутренний формат, в зависимости от спецификаторов формата, находящихся в управляющей строке. Затем данные передаются в программу через переменные, указанные в списке аргументов. Управляющая строка и список аргументов обязательны для функции `scanf()`. Спецификаторы или команды формата начинаются с символа '%' (те же самые спецификаторы используются в функции `printf()`). Перечислим некоторые из них:

- `%d` — чтение целого десятичного числа (результат должен быть записан в переменную типа `int` в списке аргументов);
- `%f` — чтение вещественного числа (результат должен быть записан в переменную типа `float` в списке аргументов);
- `%lf` — чтение вещественного числа двойной точности (результат должен быть записан в переменную типа `double` в списке аргументов);
- `%c` — чтение символа (результат должен быть записан в переменную типа `char` в списке аргументов).

В отличие от функции `printf()` аргументами для функции ввода `scanf()` могут быть только адреса переменных (в более общем случае объектов программы). Чтобы получить адрес переменной, необходимо перед ее именем поставить символ '&'. Например, адрес переменной `x` выглядит так: `&x`.

В рассматриваемом примере функция `scanf()` используется для ввода целого числа, значение которого записывается в переменную `x`. В этот момент выполнение программы будет приостановлено до тех пор, пока пользователь не введет необходимые данные с клавиатуры. Нажатие клавиши <Enter> означает конец ввода.

Последний оператор изучаемой программы, расположенный в седьмой строке, необходим для того, чтобы сообщить пользователю о результатах работы программы. Для этого в конец управляющей строки помещена команда формата `%d`, а в списке аргументов указана переменная, значение которой будет выведено в этом месте на экран.

В последней строке программы стоит закрывающая фигурная скобка, определяющая конец функции `main`, а в нашем случае и всей программы.

Предлагаем самостоятельно разобраться, как работает программа, демонстрирующая ввод и вывод символа.

Пример 1.2

```
#include<stdio.h>
void main()
{
  char x;
  printf("\nВведите символ: ");
  scanf("%c",&x);
  printf("\nВы ввели символ %c",x);
  return;
}
```

Задание

- попробуйте в процессе работы программы ввести букву, цифру, другие знаки;
- вместо `%c` поставьте `%f` или `%d`.

Функция `printf()` позволяет программисту выводить данные в нужном ему формате. Между знаком '%' и форматом команды может стоять целое число. Оно указывает наименьшее число позиций курсора, отводимое для печати. Выравнивание выводимой информации происходит по правому краю, «лишние» позиции слева заполняются пробелами. Если число или строка длиннее указанного количества позиций, то они печатаются полностью, при этом обозначенная ширина поля игнорируется.

Попробуйте изменить оператор вывода в седьмой строке примера 1.1 на следующий:

```
  printf("\n Вы ввели число %3d",x);
```

Запустите программу на выполнение и введите однозначное число, затем двузначное, трехзначное, четырехзначное.

Чтобы указать количество цифр в дробной части внешнего представления вещественного числа, после ширины поля надо поставить точку, а затем целое число, определяющее количество десятичных знаков.

Эту возможность демонстрирует простой пример. При помощи команды формата `%5.2f` для вывода на экран вещественного числа отводится 5 позиций, из них 2 для дробной части:

Пример 1.3

```
#include<stdio.h>
void main()
{
  float x;
  printf("\nВведите число: ");
  scanf("%f",&x);
  printf("\nВы ввели число %f",x);
  printf("\nОграничим количество цифр,);
  printf(" выводимых после запятой %5.2f",x);
  return;
}
```

Задание

- попробуйте использовать другие числа в команде формата, например 2 и 1 вместо 5 и 2;
- вместо `%f` поставьте `%c` или `%d`.

В следующих двух примерах рассматривается программа, которая находит сумму двух введенных чисел. Эти примеры показывают различные возможности функций ввода и вывода, в частности одновременное применение нескольких команд формата в одном операторе, использование выражений в списке аргументов.

Пример 1.4

```
#include<stdio.h>
void main()
{
  int a,b,sum;
  printf("\nВведите два числа\n");
  scanf("%d%d",&a,&b);
  sum=a+b;
  printf("Сумма чисел равна %d",sum);
  return;
}
```

Задание

- попробуйте написать аналогичную программу для вещественных чисел;
- уберите знак `&` в функции `scanf()`.

Пример 1.5

```
#include<stdio.h>
void main()
{
  int a,b;
  printf("\nВведите два числа\n a=");
  scanf("%d",&a);
  printf("\n b=");
  scanf("%d",&b);
  printf("\n %d+%d=%d",a,b,a+b);
  return;
}
```

Операторы ввода и вывода информации, оператор присваивания относятся к операторам преобразования данных. В языке С есть еще один тип операторов — операторы управления работой программы, используемые для изменения линейной последовательности действий. К ним относятся:

- операторы выбора (например `if`);
- операторы циклов (например `for`, `while`);
- операторы переходов (например `break`, `continue`, `return`).

Рассмотрим оператор выбора. Условный оператор имеет две формы. Сокращенная форма выглядит так:

`if(`*условие*`)` *оператор;*

В качестве *условия* может быть использовано логическое или арифметическое выражение. *Оператор* (это может быть один оператор языка С или группа операторов, заключенная в фигурные скобки) выполняется только в случае истинности условия, т.е. при ненулевом значении. Логическое выражение может включать в себя операции отношения:

- `<` — меньше;
- `<=` — меньше или равно;
- `>` — больше;
- `>=` — больше или равно;
- `==` — равно;
- `!=` — не равно;

и логические операции:

- `&&` — и;
- `||` — или;
- `!` — не.

Приведем пример использования сокращенного условного оператора для определения соотношения между двумя числами (больше, меньше или равно).

Пример 1.6

```
#include<stdio.h>
void main()
{
  int x,y;
  printf("\nВведите два числа\n");
  scanf("%d%d",&x,&y);
  if(x>y)
    printf("%d > %d",x,y);
  if(x<y)
    printf("%d < %d",x,y);
  if(x==y)
    printf("%d = %d",x,y);
}
```

Полная форма условного оператора имеет вид:

`if(`*условие*`)` *оператор1;* `else` *оператор2;*

Если условие истинно, выполняется только первый оператор (если используется группа операторов, заключенная в фигурные скобки, то ';' перед `else` не ставится), в противном случае — только второй.

Та же задача, что и в предыдущем примере, может быть решена с использованием полного условного оператора.

Пример 1.7

```
#include<stdio.h>
void main()
{
  int x,y;
  printf("\nВведите два числа\n");
  scanf("%d%d",&x,&y);
  if(x>y)
    printf("%d > %d",x,y);
  else
    if(x<y)
      printf("%d < %d",x,y);
    else
      printf("%d = %d",x,y);
  return;
 }
```

Группа операторов цикла позволяет программисту организовать повторение некоторых действий несколько раз.

Параметрический цикл `for` имет вид:

`for(`*выражение1; условие; выражение2*`)` *тело цикла;*

Выражение1 инициализирует, иначе говоря, задает начальные значения для параметров цикла и любых других переменных, используемых в цикле. Оно определяет действия, выполняемые до начала цикла. Так же, как и в операторе `if`, *условие* является логическим или арифметическим выражением. Выполнение тела цикла (это может быть один оператор или несколько, заключенных в фигурные скобки) происходит до тех пор, пока условие истинно. *Выражение2* определяет изменение параметров цикла или других переменных, необходимое для следующих повторений цикла. Отметим, что *выражение1 и выражение2* могут состоять из нескольких операторов, разделенных запятыми.

Работу цикла `for` можно обобщить следующим образом: *выражение1* выполняется только один раз перед началом работы цикла, затем условие проверяется на истинность (если оно ложно, то цикл заканчивается и начинает выполняться следующий за ним оператор), далее выполняются операторы тела цикла, *выражение2*, снова проверка условия, и т.д.

Приведем пример использования цикла `for` в задаче, где нужно подсчитать сумму целых чисел из заданного промежутка.

Пример 1.8

```
#include<stdio.h>
void main()
{
  int i,sum,n,N;
  printf("Введите начало и конец промежутка\n");
  scanf("%d%d",&n,&N);
  for(i=n,sum=0;i<=N;i++)
    sum+=i;
  printf("Сумма целых чисел от %d до %d равна %d",n,N,sum);
  return;
}
```

Задание

- попробуйте ввести начало и конец промежутка так, чтобы начало по значению было больше конца;

- обратите внимание на операцию `+=`, замените `sum+=i;` на оператор `sum=sum+i;`
- удалите из цикла оператор, присваивающий 0 переменной `sum`.

Рассмотрим пример использования оператора выбора в теле цикла. В задаче требуется найти сумму целых четных чисел из заданного промежутка.

Пример 1.9

```
#include<stdio.h>
void main()
{
  int i,sum,n,N;
  printf("\nВведите начало и конец промежутка\n");
  scanf("%d%d",&n,&N);

  for(i=n,sum=0;i<=N;i++)
    if(i%2==0)
    {
      sum+=i;
      printf("%d+",i);
    }
  printf("\b=%d",sum);
  return;
}
```

Задание

- удалите управляющую последовательность `\b` из последнего оператора вывода;
- замените условие в операторе `if` на `i%2!=1`, а затем на `i%2==1`.

Другой оператор цикла в языке С — это `while`. Его основная форма имеет следующий вид:

`while(`*условие*`)` *тело цикла;*

Операторы тела цикла выполняются до тех пор, пока условие не станет ложным. Обратим внимание, что проверка истинности условия осуществляется перед каждым выполнением действий в теле цикла, поэтому, если условие заведомо ложное, то они не выполнятся ни разу, а если условие всегда истинное, цикл станет бесконечным.

Снова рассмотрим задачу вычисления суммы целых чисел из заданного промежутка. Обратите внимание на использование директивы препроцессора `#define` для задания границ числового промежутка. Эта директива заменяет определенные программистом идентификаторы (в нашем случае `n` и `N`) на указанную после них последовательность символов (соответственно 9 и 10).

Пример 1.10

```
#include<stdio.h>
#define n 9
#define N 10
void main()
{
  int i,sum;
  i=n;
  sum=0;
  while(i<=N)
  {
    sum+=i;
    i++;
  }
  printf("Сумма целых чисел от %d до %d равна %d",n,N,sum);
}
```

Задание

- измените в препроцессорной директиве 10 на 100;
- измените в условии цикла `<=` на `>`.

При написании любого цикла надо обращать особое внимание на инициализацию параметров цикла и их корректное изменение. Ошибки в этих частях циклов приводят к получению неверных результатов или даже к зацикливанию программы.

Если программа должна обрабатывать некоторое множество одинаковых данных, например несколько целых чисел, удобно хранить эти данные в массиве. Массив представляет собой совокупность элементов, имеющих одни и те же характеристики. Определение одномерного массива выглядит так:

тип имя `[`*количество элементов*`];`

Количество элементов должно быть целой константой. Для задания массива переменного размера служит специальный механизм, называемый динамическим выделением памяти. Каждому элементу

массива соответствует его порядковый номер или индекс. Обратиться к соответствующему элементу можно так:

имя массива [*индекс элемента*]

В языке С индексы в массивах начинаются с нуля. Если объявлен массив `int a[100]` , это значит, что массив содержит 100 целочисленных элементов от `a[0]` до `a[99]`.

Рассмотрим пример ввода и вывода одномерного массива целых чисел. В этом примере определен массив для 100 элементов, но пользователь может по своему желанию использовать меньшее количество.

Пример 1.11

```
#include<stdio.h>
#define N 20
void main()
{
  int M[N],i,n;
  printf("\nВведите количество элементов массива\n");
  scanf("%d",&n);
  printf("\nВведите %d целых чисел\n",n);
  for(i=0;i<n;i++)
    scanf("%d",&M[i]);
  printf("\nВведен массив из %d элементов\n",n);
  for(i=0;i<n;i++)
    printf("%3d",M[i]);
}
```

Задание

- попробуйте вводить элементы массива через пробел, а при следующем запуске программы — через клавишу <Enter>;
- измените в команде формата цифру 3 на любую другую или удалите ее;
- введите значение переменной `n` больше константы `N`;
- подумайте, какие ограничения накладываются на значение переменной `n` и добавьте в программу проверку правильности значения, введенного пользователем;
- измените программу так, чтобы элементы массива выводились вместе с их индексами, например `M[0]=2 M[1]=0`...

Следует отметить, что в языке С не считается ошибкой обращение к участкам памяти, находящимся за пределами объявленного

массива, например M[N] или M[-1]. Контроль за значениями, которые в ходе работы программы принимает индекс массива, должен осуществлять программист.

В следующем примере массив определяется внутри программы с помощью инициализации. Программа решает несколько задач: находит индексы элементов, равных нулю, вычисляет сумму и количество четных элементов массива. Обратите внимание на особенности использования второй директивы препроцессора #define.

Пример 1.12

```
#include<stdio.h>
#include<conio.h>
#define N 10
#define STAR printf("\n*******************************\n");
void main()
{
  int M[N]={2,0,0,6,1,0,0,7,0,2},i,k,sum;
  STAR;
  printf("\nИсходный массив:");
  for(i=0;i<N;i++)
    printf("\nM[%d]=%d",i,M[i]);
  STAR;
  printf("Нулевые элементы массива:\n");
  for(i=0;i<N;i++)     /*Определение нулевых элементов*/
    if(M[i]==0)
      printf("M[%d]\t",i);
  k=0;
  sum=0;
  for(i=0;i<N;i++)       /*Сумма и количество нечетных*/
    if(M[i]%2==0)
    {
      k++;
      sum+=M[i];
    }
  printf("\nКоличество четных элементов %d",k);
  printf("\nСумма четных элементов %d",sum);
}
```

Задание

- попробуйте использовать директиву `#define` для определения заголовка цикла `for`;
- измените программу так, чтобы найти индексы элементов, равных числу, заданному пользователем;
- измените программу так, чтобы элементы массива вводились пользователем с клавиатуры.

Каждый объект в языке C, например переменная, кроме имени и значения имеет свой адрес, соответствующий ее размещению в памяти компьютера. В программе можно использовать это свойство, а также объявлять переменные, содержащие такие адреса. Эти переменные называются указателями. Описать указатель можно так:

*тип *имя переменной;*

Для работы с указателями существуют две специальные операции: `&` (взятие адреса) и `*` (содержимое по адресу). Например, если объявлены две переменные типа `float` и один указатель на переменную типа `float`:

```
float x, y, *z;
```

то им можно присваивать такие значения:

```
x=0.5;
z=&x;
y=*z;
```

Тогда `z` будет указывать на переменную действительного типа, имеющую значение `0.5`, а переменной `y` будет присвоено значение `0.5`.

Кроме описанных выше операций к указателям могут применяться сложение и вычитание. В результате этих операций значение указателя меняется по–разному в зависимости от размера типа данных, на который указывает переменная. Например, при изменении указателя на единицу он сдвигается на количество байтов, соответствующее длине его типа.

В языке C имя массива является указателем на его первый элемент. Если массив определен как `int A[N]`, то запись `A` тождественна записи `&A[0]`. Прибавив к имени массива целую величину, получим адрес соответствующего элемента массива: `&A[i]` и `A+i` — два способа определения адреса одного и того же элемента, а выражение `A[i]` — то же самое, что и `*(A+i)`.

Рассмотрим пример ввода и вывода массива целых чисел с применением указателей и операций над ними:

Пример 1.13

```
#include<stdio.h>
#define N 20
void main()
{
  int M[N],i,n;
  printf("\nВведите количество элементов массива\n");
  scanf("%d",&n);
  printf("\nВведите %d целых чисел\n",n);
  for(i=0;i<n;i++)
    scanf("%d",M+i);
  printf("\nВведен массив из %d элементов\n",n);
  for(i=0;i<n;i++)
    printf("%3d",*(M+i));
}
```

Для представления символьных данных в языке С используется тип `char`. Везде, где синтаксис позволяет использовать целые числа, можно использовать и символы, которые при этом представляются числовыми значениями своих внутренних кодов. Такой подход позволяет использовать операции сложения, вычитания, сравнения и т.п. при работе с символьными переменными. Эти возможности демонстрирует пример, выводящий на экран коды латинских букв, соответствующие таблице кодировки ASCII.

Пример 1.14

```
#include<stdio.h>
void main()
{
  char c;

  for(c='A';c<='Z';c++)
   printf("\n|\t%c\t-\t%d\t|\t%c\t-\t%d\t|",c,c,c+32,c+32);
  return;
}
```

Задание

- обратите внимание на использование команд формата `%c` и `%d` для вывода на экран символов и их кодов, попробуйте изменить эти команды;

- напишите программу, выводящую на экран коды только строчных или только прописных латинских букв, цифр, русских букв.

Требования к лабораторной работе №1

Для выполнения данной лабораторной работы студентам необходимо знание стандартных типов данных языка C, основных арифметических операций и умение использовать условный оператор, оператор цикла `while`, а также стандартные функции форматированного ввода и вывода `printf()` и `scanf()`.

Задачи

Часть A

A1.1 Для последовательности целых чисел заданной длины N (вводится в диалоге с пользователем) подсчитать общую сумму и количество элементов заданной последовательности, которые расположены:

1) между первым четным элементом последовательности и последним элементом, значение которого равно квадрату его порядкового номера;
2) между первым положительным элементом последовательности и последним элементом последовательности, значение которого больше его порядкового номера;
3) между первым кратным семи элементом последовательности и последним элементом последовательности, значение которого меньше его порядкового номера;
4) между первым принадлежащим диапазону от -10 до 10 элементом последовательности и последним элементом последовательности, значение которого равно квадрату его порядкового номера;
5) между первым нечетным элементом последовательности и последним элементом, значение которого равно его порядковому номеру;
6) между первым не принадлежащим диапазону от -5 до 7 элементом последовательности и последним элементом последо-

вательности, значение которого меньше его порядкового номера;

7) между первым элементом последовательности, значение которого равно квадрату его порядкового номера, и последним кратным трем элементом последовательности;
8) между первым элементом последовательности, значение которого больше его порядкового номера, и последним положительным элементом последовательности;
9) между первым элементом последовательности, значение которого меньше его порядкового номера, и последним нечетным элементом последовательности;
10) между первым элементом последовательности, значение которого равно его порядковому номеру, и последним неположительным элементом последовательности;
11) между первым элементом последовательности, значение которого равно квадрату его порядкового номера, и последним четным элементом последовательности;
12) между первым элементом последовательности, значение которого меньше его порядкового номера, и последним элементом последовательности, принадлежащим диапазону от -10 до 10.

В случае, если не существует такого первого элемента, значения искомых сумм полагаются равными нулю, в случае отсутствия описанного последнего элемента суммирование производится до конца последовательности.

A1.2 Произвести одно из перечисленных ниже преобразований входной последовательности символов c_i , $i = 1, 2, 3, ...$ Длина последовательности не более 100 символов. Считается, что последовательность считана вся, если уже прочитано 100 символов, или любое другое заданное заранее пользователем количество элементов.

1) Если c_i — буква, то заменить ее на следующую букву латинского алфавита. При этом считается, что следующей за буквой 'Z' является буква 'A', а следующей за 'z' является 'a'.
2) Если c_i — цифра, то заменить ее на цифру $9 - c_i$, буквы оставить без изменения.
3) Буквы 'A', 'O', 'U', стоящие на четных местах, заменить на '*', стоящие на нечетных местах заменить на '1', '2', '3' соответственно. Остальные символы оставить без изменений.
4) Разбить последовательность на группы по 5 символов (в последней группе сколько останется) и выдать каждую последовательность в обратном порядке.

5) Определить количество цифр, расположенных в последовательности после первой буквы ‘A’.
6) Определить количество символов между первыми двумя звездочками.
7) Определить, можно ли из букв, встречающихся в последовательности, составить Вашу фамилию.
8) Определить, сколько раз в последовательности встречаются пары одинаковых символов.
9) Определить, является ли последовательность идентификатором языка C.

Часть B

B1.3 По заданным вещественным числам x и ε произвести вычисление суммы ряда до тех пор, пока абсолютная величина очередного члена ряда не станет меньше ε. Вычисления должны быть произведены максимально экономно, так, чтобы значение каждого последующего слагаемого вычислялось исходя из значения предыдущего и без использования библиотечной функции возведения в степень. При необходимости следует проверить попадание x в область сходимости (она указана в скобках). Введенное значение ε следует проверить на положительность. В случае несоблюдения перечисленных требований программа должна выдать соответствующее сообщение и завершить работу. Программа должна вывести на печать значения x, ε, вычисленной суммы ряда, точного значения функции $f(x)$, указанной в левой части равенства, абсолютной величины последнего вычисленного члена ряда, а также разности значений вычисленной суммы и функции $f(x)$.

1) $\exp(-x) = 1 - x + \frac{x^2}{2!} - \frac{x^3}{3!} + \ldots$
2) $\sin x = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \ldots$
3) $\cos x = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \frac{x^6}{6!} + \ldots$
4) $\ln(1+x) = x - \frac{x^2}{2} + \frac{x^3}{3} - \frac{x^4}{4} + \ldots, (-1 < x \leq 1)$
5) $\operatorname{Arth} x = \frac{\ln((1+x)/(1-x))}{2} = x + \frac{x^3}{3} + \frac{x^5}{5} + \frac{x^7}{7} + \ldots, (-1 < x < 1)$
6) $\operatorname{Arsh} x = \ln(x + (x^2+1)^{\frac{1}{2}}) = x - \frac{x^3}{2\times 3} + \frac{1\times 3\times x^5}{2\times 4\times 5} - \frac{1\times 3\times 5\times x^7}{2\times 4\times 6\times 7} + \ldots, (-1 \leq x \leq 1)$

B1.4 По заданному вещественному x и целому n произвести вычисление суммы n членов ряда. При необходимости следует проверить попадание x в область сходимости (она указана в скобках). Вве-

денное значение n следует проверить на положительность. В случае несоблюдения перечисленных требований программа должна выдать соответствующее сообщение и завершить работу. Программа должна вывести на печать значения x, n, вычисленной суммы ряда, точного значения функции $f(x)$, указанной в левой части равенства, абсолютной величины последнего вычисленного члена ряда, а также разности значений вычисленной суммы и функции $f(x)$.

1) $\operatorname{arctg} x = x - \frac{x^3}{3} + \frac{x^5}{5} - \frac{x^7}{7} + \ldots, (-1 \le x \le 1)$
2) $\arcsin x = x + \frac{x^3}{2\times3} + \frac{1\times3\times x^5}{2\times4\times5} + \frac{1\times3\times5\times x^7}{2\times4\times6\times7} + \ldots, (-1 < x < 1)$
3) $(1+x)^{-3} = 1 - \frac{2\times3\times x}{2} + \frac{3\times4\times x^2}{2} - \frac{4\times5\times x^3}{2} + \ldots, (-1 < x < 1)$
4) $(1+x)^{\frac{1}{2}} = 1 + \frac{x}{2} - \frac{x^2}{2\times4} + \frac{1\times3\times x^3}{2\times4\times6} - \frac{1\times3\times5\times x^4}{2\times4\times6\times8} + \ldots, (-1 < x \le 1)$
5) $(1+x)^{-\frac{1}{2}} = 1 - \frac{x}{2} + \frac{1\times3\times x^2}{2\times4} - \frac{1\times3\times5\times x^3}{2\times4\times6} + \ldots (-1 < x \le 1)$
6) $(1-x^2)^{-\frac{1}{2}} = 1 + \frac{x^2}{2} + \frac{1\times3\times x^4}{2\times4} + \frac{1\times3\times5\times x^6}{2\times4\times6} + \ldots, (-1 < x < 1)$

B1.5 Произвести одно из перечисленных ниже преобразований входной последовательности c_i $(i = 1, 2, 3, ...)$ латинских букв и цифр. Длина последовательности не более 100 символов. Считается, что последовательность считана вся, если уже прочитано 100 символов, или если встретился символ, не являющийся ни латинской буквой, ни цифрой. Последовательность хранится в одномерном массиве.

1) Буквы, стоящие на нечетных местах заменить на буквы, симметричные им относительно латинского алфавита (‘A’ на ‘Z’, ‘B’ на ‘Y’ и т.д.). Остальные символы оставить без изменений.
2) Удалить из последовательности все цифры.
3) Заменить все строчные латинские буквы на соответствующие им прописные.
4) Определить максимальную длину подпоследовательности, состоящей только из цифр.
5) Определить, существует ли подпоследовательность, состоящая только из латинских букв, являющаяся палиндромом (читающаяся одинаково справа налево и наоборот).

B1.6 Для последовательности целых чисел заданной длины N (N вводится в диалоге с пользователем) подсчитать сумму модулей нечетных и количество четных элементов заданной последовательности, которые расположены:

1) между первым являющимся точным квадратом элементом последовательности и последним элементом последовательности, значение которого по модулю больше его порядкового номера;

2) между первым максимальным по модулю элементом последовательности и последним элементом последовательности, значение которого равно квадрату его порядкового номера;
3) между первым являющимся точным квадратом элементом последовательности и последним элементом последовательности, значение которого равно квадрату его порядкового номера;
4) между первым минимальным положительным элементом последовательности и последним элементом последовательности, значение которого по модулю больше квадрата его порядкового номера;
5) между первым максимальным отрицательным элементом последовательности и последним элементом последовательности, значение которого по модулю меньше его порядкового номера;

В случае, если не существует такого первого элемента, значения искомых сумм полагаются равными нулю, в случае отсутствия описанного последнего элемента суммирование производится до конца последовательности.

Часть C

C1.7 Вводится последовательность не более чем из 256 символов, ввод заканчивается символом '*'. Последовательность является набором предложений, в которых слова состоят из латинских букв и разделены произвольным количеством пробелов. Каждое предложение заканчивается точкой. Проверить правильность вводимой последовательности, в случае ошибки, выдать соответствующее сообщение и завершить работу. Отредактировать введенную последовательность: удалить все лишние пробелы; заменить первую букву каждого предложения на прописную. Распечатать каждое предложение, расположив слова в обратном порядке.

C1.8 Вводится последовательность не более чем из 256 символов, ввод заканчивается символом '#'. Последовательность является набором предложений, в которых слова состоят из латинских букв и разделены произвольным количеством пробелов. Каждое предложение заканчивается точкой. Проверить правильность вводимой последовательности, в случае ошибки, выдать соответствующее сообщение и завершить работу. Отредактировать введенную последовательность: в каждом предложении расположить слова по возрастанию длины, предложения расположить по убыванию количества слов.

C1.9 Преобразовать число заданное пользователем в некоторой системе счисления, в число другой системы счисления по желанию пользователя. Если основание системы счисления превышает 10, то необходимые символы берутся по порядку из латинского алфавита.

C1.10 Вводятся координаты трех точек на плоскости. Определить, являются ли данные точки вершинами треугольника. Если да, то найти уравнения сторон, биссектрис, медиан и высот треугольника. По введенным координатам четвертой точки определить, находится она внутри или вне треугольника.

C1.11 Задан набор a_1, a_2, ... ,a_n вещественных чисел, упорядоченный:

1) по возрастанию,
2) по убыванию.

Концом набора является символ '*'. Количество чисел n в наборе заранее неизвестно, $n \leq 40$. Для заданных вещественных чисел x, $\varepsilon > 0$ и монотонной функции $G(x)$ найти методом:

1) перебора в следующем порядке: a_1, a_n, a_2, a_{n-1}, ...
2) дихотомии

и напечатать i и a_i, для которых $G(a_i) = x$ с точностью ε. Функция G определяется как суперпозиция трех функций:

$$G(x) = F_k(F_l(F_m(x))),$$

которые выбираются из следующего списка:

1) $F_1(x) = x/2 - 1$,
2) $F_2(x) = 2x + 1$,
3) $F_3(x) = 3x - 2$,
4) $F_4(x) = x^3 + x$,
5) $F_5(x) = 2x^3 + 3x$.

Если введенные данные не удовлетворяют сформулированным требованиям, следует выдать соответствующее сообщение и завершить работу программы.

2. Двумерные массивы

Основные понятия

В языке С можно определять многомерные массивы, простейшей формой которых является двумерный массив, или матрица. Его можно описать так:

тип имя [*количество строк*][*количество столбцов*];

При этом обе размерности массива должны быть константными выражениями. Память для всех массивов, определенных таким образом, выделяется в процессе компиляции и сохраняется до конца работы программы.

Для доступа к отдельному элементу массива применяется конструкция вида `a[i][j]`, где `i` (индекс, определяющий номер строки) и `j` (индекс, определяющий номер столбца) — выражения целого типа. Каждый индекс может изменяться от 0 до значения соответствующей размерности, не включая ее.

Двумерный массив, объявленный как `int a[3][4]`, можно представить в виде таблицы из трех строк и четырех столбцов. В памяти компьютера массив располагается непрерывно по строкам:

`a[0][0], a[0][1], a[0][2], a[0][3], a[1][0],..., a[2][3],`

Рассмотрим программу, осуществляющую ввод и вывод матрицы, ее размеры с некоторыми ограничениями задает пользователь.

Пример 2.1

```
#include<stdio.h>
#define N 5
#define M 5
void main()
{
  int A[N][M],i,j,n,m;
  printf("\n Введите размер матрицы\n количество строк ");
  scanf("%d",&n);
  printf("количество столбцов ");
  scanf("%d",&m);
  printf("\n Введите матрицу размера %d x %d \n",n,m);
  for(i=0;i<n;i++)
    for(j=0;j<m;j++)
      scanf("%d",&A[i][j]);
```

```
  printf("\n Введена матрица\n");
  for(i=0;i<n;i++)
  {
    for(j=0;j<m;j++)
      printf("%3d",A[i][j]);
    printf("\n");
  }
}
```

Задание

- попробуйте вводить элементы матрицы через пробел, а после окончания ввода строки нажимать клавишу “Enter”;
- удалите в цикле вывода матрицы оператор `printf("\n");`
- добавьте в программу проверку правильности количества строк и столбцов, введенных пользователем;
- подумайте и напишите, как изменится программа, если известно, что матрица квадратная.

При описании массива можно задать начальные значения его элементов, их записывают в фигурных скобках. Элементы массива инициализируются в порядке их расположения в памяти. Можно задавать начальные значения не для всех элементов матрицы, для этого список значений для каждой строки заключается в дополнительные фигурные скобки.

Приведем пример программы, которая находит максимальный элемент в матрице из целых чисел.

Пример 2.2

```
#include<stdio.h>
#define N 3
#define M 5
void main()
{
  int A[N][M]={1,2,1,3,5,2,3,4,5,1,1,3,2,6,1},i,j,n,m,max;
  max=A[0][0];              /* Инициализация максимума */
  for(i=0;i<N;i++)
    for(j=0;j<M;j++)
      if(A[i][j]>max)  /*Сравнение максимума и текущего*/
        max=A[i][j];   /*элемента и изменение максимума*/
  printf("\n Максимальный элемент матрицы %d", max);
}
```

Задание

- попробуйте изменить программу так, чтобы элементы матрицы вводил пользователь;
- удалите оператор `max=A[0][0];`
- выведите на экран максимальный элемент матрицы вместе с его индексами, например `A[2][3]=6`;
- напишите программу нахождения минимума в матрице из действительных чисел.

Следующая программа находит и выводит на печать сумму элементов в каждом столбце матрицы. Обратите внимание, что при этом вначале записывается цикл, изменяющий номер столбца, а уже затем цикл, изменяющий номер строки.

Пример 2.3

```
#include<stdio.h>
#define N 3
#define M 5
void main()
{
  int A[N][M]={{1,2,1,3,5},{2,3,4,5,1},{1,3,2,6,1}};
  int i,j,n,m,sum;
  printf("\nСумма элементов по столбцам\n ");
  for(j=0;j<M;j++)
  {
    for(i=0,sum=0;i<N;i++)
      sum+=A[i][j];
    printf("\nстолбец %d - сумма %d ",j,sum);
  }
}
```

Задание

- переставте оператор `sum=0;` в начало программы;
- измените программу так, чтобы пользователь вводил элементы матрицы по столбцам, а не по строкам, как в предыдущих примерах;
- напишите программу, считающую сумму элементов в каждой строке матрицы.

Требования к лабораторной работе №2

Вторая лабораторная работа преследует цель развития навыков обработки структурированных данных на примере работы с матрицами и форматированного ввода и вывода данных. При вводе и выводе элементы матриц должны быть расположены в их естественном виде (прямоугольная или квадратная таблица).

Задачи

Часть А

А2.1 Задана матрица A вещественных чисел размера $N \times N$ ($N \leq 20$ задается как параметр). Построить по ней матрицу B того же размера, элемент $b_{i,j}$ которой равен:

1) минимуму,
2) сумме,
3) максимуму,
4) сумме модулей,
5) разности между максимумом и минимумом

всех тех элементов матрицы A, которые расположены в некоторой ее области (на соответствующем рисунке закрашена), определяемой по номеру строки i и номеру столбца j так, как показано на рис. 2.1 на с. 38 (границы входят в область).

На печать следует вывести как исходную, так и результирующую матрицу.

А2.2 Задана матрица A целых чисел размера $N \times M$ ($N, M \leq 20$, N и M задаются как параметры). Преобразовать ее в матрицу B путем удаления:

1) строки этой матрицы с минимальным номером, в которой все элементы положительны и упорядочены по возрастанию;
2) строки этой матрицы с максимальным номером, в которой все элементы четны и упорядочены по возрастанию;
3) строки этой матрицы с минимальным номером, в которой все элементы отрицательны и упорядочены по убыванию;
4) строки этой матрицы с максимальным номером, в которой все элементы нечетны и упорядочены по убыванию;
5) строки этой матрицы с минимальным номером, в которой все элементы делятся на 3 и упорядочены по возрастанию;

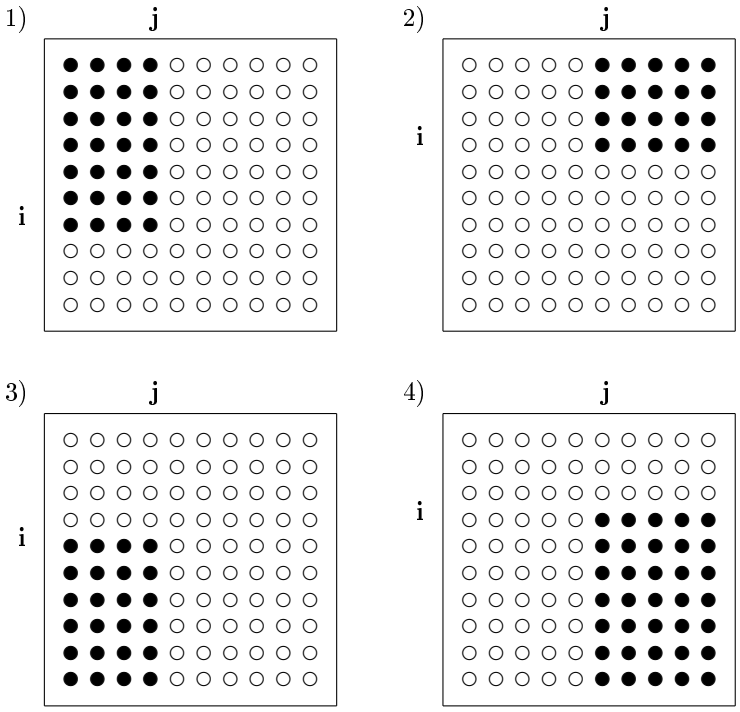

Рис. 2.1.

6) столбца этой матрицы с максимальным номером, в котором все элементы неотрицательны и упорядочены по убыванию;
7) столбца этой матрицы с минимальным номером, в котором все элементы не делятся на 5 и упорядочены по убыванию;
8) столбца этой матрицы с максимальным номером, в котором все элементы больше 1 и упорядочены по возрастанию;
9) столбца этой матрицы с минимальным номером, в котором все элементы неположительны и упорядочены по возрастанию;
10) столбца этой матрицы с максимальным номером, в котором все элементы отрицательны и упорядочены по убыванию.

Если таковой строки (столбца) не найдется, программа должна выдать соответствующее сообщение. На печать следует вывести как исходную, так и результирующую матрицу.

Часть B

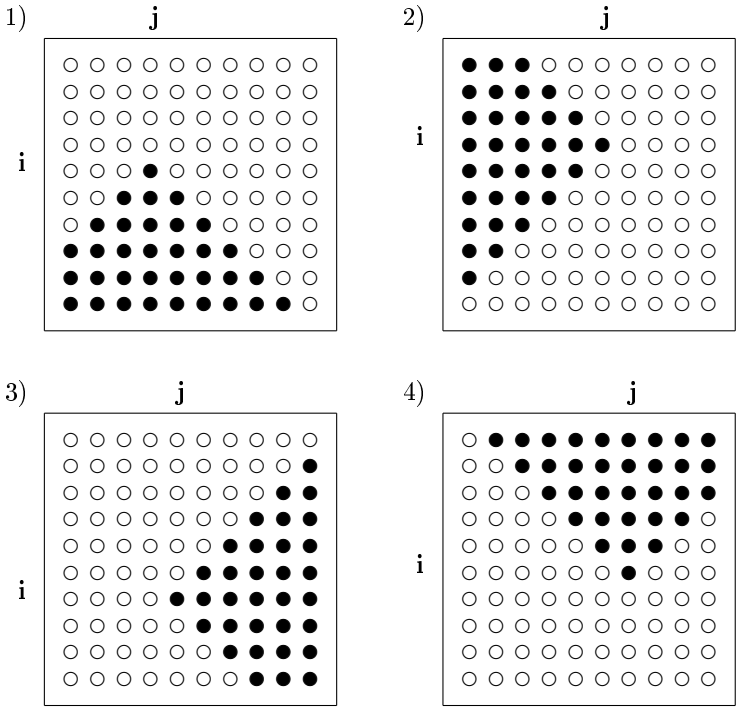

Рис. 2.2.

B2.3 Задана матрица A вещественных чисел размера $N \times N$ ($N \leq 20$, задается как параметр). Построить по ней матрицу B того же размера, элемент $b_{i,j}$ которой равен:

1) минимуму,
2) сумме,
3) максимуму,
4) сумме модулей,
5) разности между максимумом и минимумом

всех тех элементов матрицы A, которые расположены в некоторой ее области (на соответствующем рисунке закрашена), опреде-

ляемой по номеру строки i и номеру столбца j так, как показано на рис. 2.2 на с. 39 (границы входят в область).

На печать следует вывести как исходную, так и результирующую матрицу.

B2.4 Задана матрица A вещественных чисел размера $N \times N$ ($N \le 20$, задается как параметр). Построить по ней матрицу B такого же размера, элементы которой получаются:

1) симметричным отражением элементов матрицы A относительно центра матрицы A;
2) симметричным отражением элементов матрицы A относительно главной диагонали;
3) симметричным отражением элементов матрицы A относительно побочной диагонали;
4) симметричным отражением элементов матрицы A относительно горизонтальной оси симметрии;
5) симметричным отражением элементов матрицы A относительно вертикальной оси симметрии;
6) поворотом матрицы A по часовой стрелке на 90 градусов;
7) поворотом матрицы A по часовой стрелке на 180 градусов;
8) поворотом матрицы A по часовой стрелке на 270 градусов;

и вычислить матрицу C, которая получается как:

1) $C = (A + E) \times B$;
2) $C = (B - E) \times A$;
3) $C = A \times (B + E) + E$;
4) $C = (B + E) \times (A - E)$.

Здесь E — матрица размера $N \times N$, элементы которой, лежащие на главной диагонали, равны единице, остальные равны нулю. На печать следует выдать в виде квадратных таблиц матрицы A, B, C. Для хранения всех основных и промежуточных результатов разрешается использовать не более трех массивов.

B2.5 Задана матрица A целых чисел размера $N \times M$ ($N, M \le 20$, N и M задаются как параметры). Осуществить циклический сдвиг:

1) строк матрицы вниз или вверх в зависимости от введенного режима;
2) столбцов матрицы вправо или влево в зависимости от введенного режима

на n (параметр) позиций.

B2.6 Задана матрица A целых чисел размера $N \times M$ ($N, M \leq 20$, N и M задаются как параметры). Уплотнить матрицу, «удаляя» из нее строки и столбцы, заполненные нулями.

Часть C

C2.7 Задана матрица A целых чисел размера $N \times M$ ($N, M \leq 20$, N и M задаются как параметры). Осуществить циклический сдвиг элементов матрицы следующим образом: элементы нулевой строки сдвигаются в последний столбец сверху вниз, из него — в последнюю строку справа налево, из нее — в нулевой столбец снизу вверх, из него — в первую строку слева направо и далее «по спирали» (см. рис. 2.3). Последний элемент матрицы, соответствующий описанному обходу, сдвигается в первый ее элемент с индексами $(0, 0)$. Каждый шаг сдвига осуществляется с некоторой временной задержкой (интервал задержки — параметр программы). Работа программы завершается нажатием клавиши.

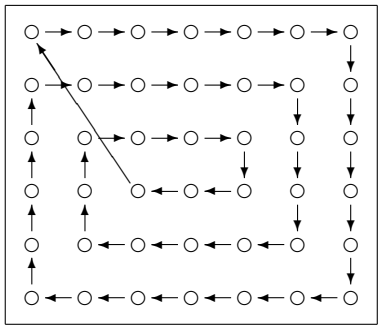

Рис. 2.3.

C2.8 Задана матрица A целых чисел размера $N \times M$ ($N, M \leq 20$, N и M задаются как параметры). Осуществить циклический сдвиг элементов матрицы следующим образом: элементы нулевой строки сдвигаются в нулевой столбец сверху вниз, из него — в последнюю строку слева направо, из нее — в последний столбец снизу вверх, из него — в первую строку справа налево и далее «по спирали» (см. рис. 2.4 на с. 42). Последний элемент матрицы, соответствующий

описанному обходу, сдвигается в последний элемент нулевой строки. Каждый шаг сдвига осуществляется с некоторой временной задержкой (интервал задержки — параметр программы). Работа программы останавливается нажатием клавиши "пробел"и продолжается повторным нажатием этой клавиши.

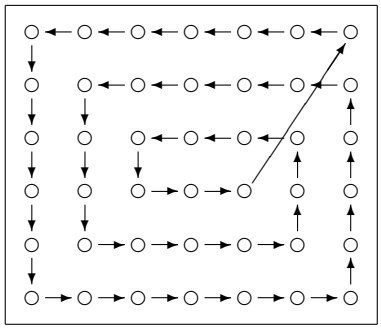

Рис. 2.4.

C2.9 Задана матрица A символов размера $N \times N$ (N задается как параметр). Осуществить циклический сдвиг элементов матрицы следующим образом: элементы нулевой строки сдвигаются в нулевой столбец сверху вниз, из него — в последнюю строку слева направо, из нее — в последний столбец снизу вверх, из него — в нулевую строку справа налево и далее «по кругу против часовой стрелки». Элементы первой строки сдвигаются аналогично «по кругу по часовой стрелке», и так далее, чередуя направления сдвига (см. рис. 2.5 на с. 43). Каждый шаг сдвига осуществляется с некоторой временной задержкой (интервал задержки — параметр программы). Работа программы останавливается нажатием клавиши "пробел"и продолжается повторным нажатием этой клавиши.

C2.10 Задана матрица A символов размера $N \times N$ (N задается как параметр). Осуществить циклический сдвиг элементов матрицы следующим образом: элементы нулевой строки сдвигаются в последний столбец сверху вниз, из него — в последнюю строку справа налево, из нее — в нулевой столбец снизу вверх, из него — в нулевую строку слева направо и далее «по кругу по часовой стрелке». Элементы первой строки сдвигаются аналогично «по кругу против часовой стрелки», и так далее, чередуя направления сдвига (см.

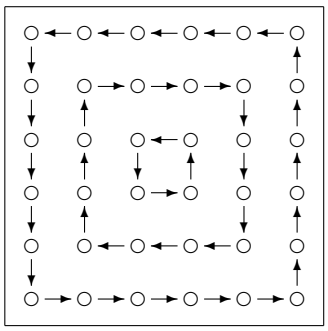

Рис. 2.5.

рис. 2.6 на с. 43). Каждый шаг сдвига осуществляется с некоторой временной задержкой, причем сдвиг «по часовой стрелке» происходит быстрее, чем «против». Работа программы останавливается нажатием клавиши.

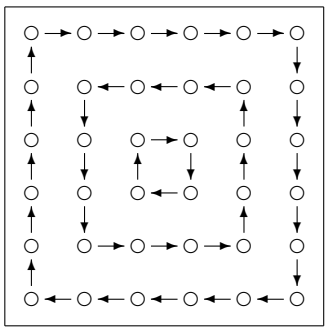

Рис. 2.6.

3. Строки

Основные понятия

В языке С не введено отдельного типа для представления строк. Роль строки играет массив символов, заканчивающийся нулевым байтом: '\0'. Поэтому при объявлении массива, с которым вы собираетесь работать как со строкой, следует зарезервировать место под нулевой байт. Например, если хранить слово "hello" как строку символов, то необходимо объявить массив из шести элементов: пять символов для букв и шестой для символа '\0'.

Для ввода и вывода символьных значений в управляющих строках библиотечных функций `printf()` и `scanf()` используется спецификатор `%c`. Кроме этих функций для ввода символов можно использовать функцию без параметров `getchar()`, которая позволяет читать из входного потока (обычно с клавиатуры) по одному символу. Как и при использовании функции `scanf()`, чтение вводимых данных начинается после нажатия клавиши <Enter>.

Если строка вводится в программе по одному символу (например, с помощью функции `scanf()` или `getchar()`), то признак конца строки (нулевой байт) необходимо присвоить соответствующему элементу массива в явном виде, обычно с помощью отдельного оператора:

```
s[0]='h';
s[1]='e';
s[2]='l';
s[3]='l';
s[4]='o';
s[5]='\0';
```

Иногда строка вводится внутри программы как константа:

```
char s[]="hello";
```

В этом случае признак конца строки будет поставлен компилятором независимо от программиста.

Рассмотрим пример программы, которая вводит и выводит полное имя пользователя. Вводимое имя не должно быть больше сорока символов. Информация хранится в массиве `FIO[N]`. Ввод и вывод

данных осуществляется посимвольно. При выводе элементов массива на экран признак конца строки является признаком окончания вывода символов.

Пример 3.1

```
#include<stdio.h>
#define N 41
void main()
{
  char FIO[N],ch;
  int i;
  printf("\nВведите Ваши фамилию, имя и отчество через ");
  printf("пробел\n для окончания ввода нажмите <Enter>\n");
  i=0;
  while(1)
  {
   /*ввод очередного символа*/
    ch=getchar();
   /*выход из цикла по нажатию клавиши <Enter>*/
    if(ch=='\n') break;
   /*выход из цикла при вводе <<лишнего>> символа*/
    if(i>=N) break;
   /*запись введенного символа в массив и
     увеличение переменной цикла*/
    FIO[i++]=ch;
  }
   /*запись признака конца строки*/
  FIO[i]='\0';
  printf("\n Здравствуйте, ");
   /*Посимвольный вывод строки*/
  for(i=0;FIO[i]!='\0';i++)
    printf("%c",FIO[i]);
  printf("!");
  return;
}
```

Задание

- попробуйте убрать условие `i>=N` и ввести имя длиной более сорока символов;
- удалите оператор, записывающий в массив признак конца строки.

Обратите внимание на использование оператора ++ для изменения индекса массива в цикле, осуществляющем ввод данных. Так как используется постфиксная форма этого оператора, сначала происходит запись символа в текущий элемент массива, и только затем индекс увеличивается на единицу.

В языке С существует возможность вводить строку не посимвольно, а целиком. Для этого можно использовать функцию `scanf()` с управляющей последовательностью `%s`. При этом надо помнить, что функция `scanf()` вводит символы только до пробела. Второй способ ввести строку — использовать функцию `gets()`. Эта функция считает все символы, в том числе и пробелы до нажатия клавиши <Enter>, т.е. до символа `\n`. Обе функции автоматически ставят в конец строки нулевой байт. Параметром функций служит имя массива символов.

Вывод строки можно осуществить с помощью функции `printf()`. Для этого в управляющую строку необходимо включить спецификатор `%s`. При этом будет выведена последовательность символов до признака конца строки.

Следующий пример демонстрирует использование функций `scanf()` и `printf()` соответственно для ввода и вывода строк символов.

Пример 3.2

```
#include<stdio.h>
#define F 15
#define I 10
void main()
{
  char name[I],fam[F];
  printf("\nВведите Ваши имя и фамилию\n");
  scanf("%s",name);
  scanf("%s",fam);
  printf("\nПривет, %s %s!",name,fam);
  return;
}
```

Обратите внимание, как используются имена массивов в аргументах функций `scanf()` и `printf()`.

Рассмотрим программу, определяющую длину строки (количество введенных символов). Для ввода строки используется функция `gets()`.

Пример 3.3

```
#include<stdio.h>
#define N 51
void main()
{
  char str[N],ch;
  int i;
  printf("\nВведите строку символов:\n");
  gets(str);
  i=0;
  while(str[i]!='\0')
    i++;
  printf("\nДлина строки %d символов.",i);
  return;
}
```

В этом примере используется тот факт, что индекс массива одновременно несет информацию о количестве введенных символов. Следует отметить, что в программе невозможно осуществить проверку соответствия длины вводимой строки с ограничением, задаваемым константой `N`.

Для упрощения структуры программы удобно разбить задачу на отдельные подзадачи и оформить каждую из них в виде функции. К функции можно обратиться по имени, передать ей значения параметров и получить результат. Передача в функцию различных аргументов позволяет, записав ее один раз, использовать многократно для разных данных.

Функция должна быть определена до момента ее использования в программе. Основная форма описания функции имеет вид:

тип имя(*список параметров*){*тело функции*}

Тип определяет тип значения, возвращаемого функцией. Имя функции — это идентификатор, выбираемый программистом, как правило соответствующий по смыслу задаче, решаемой данной функцией. Имя не должно совпадать со служебными словами языка и с именами других объектов программы. Список параметров состоит из перечня типов и имен параметров (для каждого отдельного параметра обязательно указывается его тип), разделенных запятыми. Список параметров может быть пуст. Приведем пример правильного объявления функции:

```
int f(int x, int y, float z) {...}
```

Внутри тела функции находятся описания объектов и операторы, необходимые для решения поставленной задачи. Обязательным, но не всегда явно используемым здесь оператором является `return` — оператор возврата из функции в точку программы, где эта функция была вызвана. Он может быть использован в двух формах:

1) `return;`
2) `return` *выражение*.

Первая форма соответствует завершению функции, не возвращающей никакого значения (тип функции — `void`). Во второй форме оператора *выражение* должно иметь тип, соответствующий типу функции, указанному в ее заголовке.

Для вызова функции в соответствующем месте программы необходимо указать ее имя, а затем в круглых скобках перечислить конкретные значения ее аргументов (возможно, с помощью выражений), обязательно в том порядке, в каком они указаны в определении функции.

Рассмотрим пример программы, в которой определяется и используется функция, вычисляющая модуль действительного числа. В соответствии с решаемой задачей, аргументом функции является действительное число, а возвращаемым значением — его модуль, также действительное число.

Пример 3.4

```
#include<stdio.h>
float mod(float x)
{
  if(x>=0)
    return x;
  else
    return -x;
}

void main()
{
  float a;
  printf("\nВведите число\t");
  scanf("%f",&a);
  printf("|%f|=%f",a,mod(a));
  return;
}
```

Обратите внимание, что функция вызывается внутри оператора, выводящего на экран результат ее работы.

Вернемся теперь к задаче определения длины введенной строки. Программу для решения этой задачи можно разделить на две части. В главной функции `main()` осуществляется ввод исходных данных и вывод результата работы программы. Вторая часть задачи — определение длины строки символов, заканчивающейся нулевым байтом, решается отдельной функцией. Ее аргументом является указатель на начало строки, а возвращаемым результатом — целое число.

Пример 3.5

```
#include<stdio.h>
#define N 51
int STR_len(char *STR)  /* Заголовок функции */
{
  int i;
  i=0;
  while(STR[i]!='\0')   /* Определение длины строки */
    i++;
  return i;             /* Возвращение значения */
}
void main()
{
  char str[N],ch;
  int l;
  printf("\nВведите строку символов\n");
  gets(str);
  l=STR_len(str);       /* Вызов функции */
  printf("\n Длина строки %d символов.",l);
  return;
```

Обратите внимание, что при вызове функции `STR_len` в основной функции программы ее аргументом служит имя массива символов. Результат работы функции `STR_len` сначала присваивается целочисленной переменной `l`, а затем уже эта переменная используется в следующем операторе программы.

Требования к лабораторной работе №3

При разработке программы в данной лабораторной работе необходимо использовать по крайней мере одну функцию. Все данные программы, используемые функцией, должны передаваться ей в списке параметров. При вводе исходных данных должен быть обеспечен программный контроль их правильности. Если введенные данные не удовлетворяют сформулированным требованиям, следует выдать соответствующее сообщение и завершить работу программы. Если для выполнения задания можно анализировать вводимую последовательность не целиком, а по частям (например, в ряде заданий можно рассматривать по одному слову из предложения), то следует вводить данные соответствующими «порциями».

Задачи

Часть А

А3.1 Задана последовательность идентификаторов (разделителем является запятая), оканчивающаяся точкой сразу за последним идентификатором. Каждый идентификатор содержит не более 40 символов. Длина последовательности не боле 30 идентификаторов. Найти и напечатать все идентификаторы с четным числом цифр в них.

А3.2 Задана последовательность идентификаторов, разделенных одним или несколькими пробелами, оканчивающаяся запятой сразу за последним идентификатором. Каждый идентификатор содержит не более 20 символов. Длина последовательности не более 35 идентификаторов. Найти и напечатать все идентификаторы, не содержащие цифр.

А3.3 Задана последовательность идентификаторов, разделенных запятой, оканчивающаяся звездочкой сразу за последним идентификатором. Каждый идентификатор содержит не более 15 символов. Длина последовательности не более 15 идентификаторов. Найти и напечатать все идентификаторы, состоящие из четного количества символов.

А3.4 Задана последовательность идентификаторов, разделенных одним или несколькими пробелами, оканчивающаяся точкой

сразу за последним идентификатором. Каждый идентификатор содержит не более 12 символов. Длина последовательности не более 36 идентификаторов. Найти и напечатать все идентификаторы, состоящие из заглавных букв.

А3.5 Словом назовем всякую последовательность букв латинского алфавита длины не более 20. Предложением — всякую последовательность не более 30 слов, разделенных одним или несколькими пробелами, оканчивающуюся символом '.', '!' или '?'. Написать программу, читающую заданное предложение, контролирующую его правильность и выполняющую функцию, которая:

1) по заданной букве дает количество слов, начинающихся с этой буквы, и печатает эти слова;
2) по заданному целому K печатает все слова длины K;
3) по заданному слову P дает минимальный порядковый номер этого слова в предложении, если оно там встречается, или 0, если его там нет;
4) выдает вектор длин слов в предложении;
5) по заданному целому K и букве P дает количество вхождений буквы P в K-е слово предложения;
6) по слову печатает количество вхождений каждой его буквы в предложение (повторно встречающиеся в слове буквы игнорировать);
7) по букве дает максимальный номер слова, оканчивающегося этой буквой;
8) по букве и целому K строит вектор номеров строк, содержащих эту букву в K-й позиции;
9) по букве и целому K заменяет K–ю от конца букву каждого слова предложения на заданную букву и все измененные слова распечатывает;
10) по заданному слову P определяет, встречается ли это слово в предложении не менее двух раз;
11) распечатывает все слова, состоящие из одинаковых букв и содержащие не менее двух символов;
12) распечатывает все самые длинные слова предложения;
13) распечатывает каждое слово предложения, меняя его первую букву на прописную, если необходимо;
14) распечатывает предложение, убирая лишние пробелы.

Часть B

B3.6 Задана последовательность идентификаторов, разделенных одним или несколькими пробелами, оканчивающаяся подряд тремя символами '*'. Каждый идентификатор содержит не более 15 символов. Длина последовательности не ограничена. Найти и напечатать все идентификаторы, в точности совпадающие с предыдущим идентификатором.

B3.7 Задана последовательность идентификаторов, разделенных одним или несколькими пробелами, оканчивающаяся подряд тремя символами '#'. Каждый идентификатор содержит не более 20 символов. Длина последовательности не ограничена. Найти и напечатать все идентификаторы, в которых буквы и цифры чередуются (первой может быть как буква, так и цифра).

B3.8 Словом назовем всякую последовательность букв латинского алфавита длины не более 20. Предложением — всякую последовательность не более 30 слов, разделенных одним или несколькими пробелами, оканчивающуюся символом '.', '!' или '?'. Написать программу, читающую заданное предложение, контролирующую его правильность и выполняющую функцию, которая:

1) определяет, упорядочены ли слова в предложении в лексикографическом порядке по возрастанию;
2) упорядочивает слова в предложении по убыванию в лексикографическом порядке;
3) по целым K и N находит и меняет в предложении слова с номерами K и N;
4) распечатывает все слова, у которых совпадают первые три символа;
5) определяет, сколько раз каждое слово встречается в предложении;
6) распечатывает все слова предложения, у которых доля гласных букв максимальна;
7) определить, является ли предложение палиндромом (пробелы при этом игнорируются);
8) распечатывает слова предложения, в соответствии с ростом доли согласных букв;
9) распечатывает слова предложения, упорядочивая их по длине, начиная с минимального.

B3.9 Словом назовем всякую последовательность букв латинского алфавита длины не более 10. Предложением — всякую последовательность не более 40 слов, разделенных одним или несколькими пробелами, оканчивающуюся символом '.', '!' или '?'. Написать программу, читающую два заданных предложения, контролирующую их правильность и выполняющую функцию, которая:

1) распечатывает все слова, которые встречаются в каждом из двух предложений;
2) находит самое короткое из слов первого предложения, которого нет во втором;
3) удаляет из первого предложения все слова, которые совпадают со словами из второго предложения по написанию и расположению (порядковому номеру в предложении).

B3.10 Словом назовем всякую последовательность букв латинского алфавита длины не более 20. Предложением — всякую последовательность из произвольного количества слов, разделенных одним или несколькими пробелами, оканчивающуюся символом '.', '!' или '?'. Написать программу, читающую заданное предложение, контролирующую его правильность и выполняющую функцию, которая:

1) по целому K ($20 \leq K \leq 80$) распечатывает все слова предложения «столбиком» так, чтобы все они оканчивались в K-й позиции строк (т.е. выравнивались по правому краю);
2) распечатывает слова с четными номерами в обратном порядке;
3) распечатывает слова, которые состоят из букв, содержащихся в предыдущем слове;
4) распечатывает слова, в которых чередуются гласные и согласные буквы.

Часть C

C3.11 Вводится текст, содержащий слова из русских букв. Слова разделены одним или несколькими пробелами, предложения разделены '.', '!' или '?'. Внутри предложения могут встречаться ',', '-', ';' и ':'(но не подряд). Проверить правильность вводимого текста и отредактировать его. Удалить лишние пробелы, поставить в начале каждого предложения заглавную букву. Вывести текст на печать, отформатировав его по заданной ширине и предусмотрев перенос слов.

C3.12 Разработать библиотеку функций для работы со строками. Библиотека должна включать следующие функции: определение длины строки, сравнение строк в лексикографическом порядке, сравнение строк по длине, копирование строк, поиск подстроки в строке, исключение лишних пробелов, перевод строки цифр в число, перевод числа в строку символов, перекодировка букв латинского алфавита из нижнего регистра в верхний, перекодировка букв латинского алфавита из верхнего регистра в нижний, инвертирование строки. Функции должны возвращать ноль в случае невозможности корректно завершить работу.

C3.13 Вводится текст линейной программы на языке C (отсутствуют операторы управления программой). Удалить все повторяющиеся операторы присвоения, если между двумя такими операторами, нет других, изменяющих значение переменных, содержащихся в исключаемом операторе. Например:

```
x=y+z;
a=x+y;
x=y+z;
```

в приведенной последовательности действий третий оператор должен быть исключен.

C3.14 В текстовом файле содержится набор строк. Необходимо выдать на экран все строки, включающие в себя подстроки, соответствующие заданному шаблону. Элементы шаблона:

- `.` — один любой символ;
- `*` — любое количество предыдущего символа, в том числе его отсутствие, например шаблону `a*` соответствуют: пустая подстрока, ‘a’, ‘aa’,... и т.д.;
- `+` — любое ненулевое количество предыдущего символа;
- `^` — символы подстроки расположены только с начала строки;
- `$` — символы подстроки расположены только с конца строки;
- `()` — группировка элементов шаблона, например шаблону `(ab)+` соответствуют: ‘ab’, ‘abab’, ‘abababab’ и т.п.
- `[]` — выбор одного символа из набора в скобках.

Если в шаблоне необходим один из управляющих символов в явном виде, то перед ним ставится \, например, если в подстроке должна присутствовать `*`, то в шаблоне указывается `\*`. Шаблон, несоответствующий указанным правилам, считается ошибочным и не обрабатывается.

4. Линейные списки

Основные понятия

Из базовых типов данных в языке С можно формировать производные типы, к которым относятся структуры. Структура объединяет несколько элементов (компонентов) данных в одно целое. Компоненты структуры могут быть разного типа. В соответствии со смыслом решаемой задачи компоненты могут иметь любой из типов данных, допустимых в языке.

Структурный тип определяется с помощью служебного слова `struct`:

`struct` *имя* `{` *определения элементов* `}`

Имя структуры — это идентификатор, выбираемый программистом. В фигурных скобках перечисляются описания отдельных компонентов структуры.

Компонентом структуры может быть указатель на структуру того же типа. Такая особенность позволяет создавать из элементов типа структура множества данных различной конфигурации, которые могут изменять свой размер, состав, конфигурацию во время работы программы, иначе говоря, динамически. Примером динамической организации данных служат линейные списки. Элемент списка состоит из информационного поля (или нескольких информационных полей, возможно различных типов) и указателя на следующий элемент списка.

Рассмотрим элемент списка, в информационном поле которого содержится целое число. Структура, описывающая такой тип данных, выглядит следующим образом:

```
struct List {
              int Info;
              struct List *Next;
            };
```

Для работы с элементами динамических структур данных удобно использовать указатели:

```
struct List *u;
```

В тексте программы необходимо выделить память для конкретного элемента списка.

Динамическое выделение памяти осуществляется с помощью функции `malloc()`:

`malloc(`*размер выделяемой памяти*`);`

Функция возвращает указатель на первый байт выделенной памяти. Если количества свободной памяти недостаточно, то возвращается нулевой указатель, обычно обозначаемый константой `NULL`. Запишем оператор, выделяющий память для одного элемента описанной структуры `List`:

```
u=(struct List *) malloc(sizeof(struct List));
```

Обращение к отдельным компонентам структуры осуществляется с помощью операции `->`. Заполненить поля элемента после выделения памяти можно с помощью следующих операторов:

```
u->Info=2;
u->Next=NULL;
```

Элемент можно наглядно представить так:

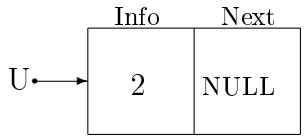

Рис. 4.1.

При формировании линейного списка в поле `Next` необходимо записывать адрес элемента следующего за рассматриваемым. У последнего элемента в этом поле должен находиться нулевой указатель, таким образом определяется окончание списка. Выделение памяти для очередного элемента и заполнение всех его полей осуществляется в зависимости от того, как пользователь вводит необходимые данные.

Обязательным условием существования списка является наличие отдельного указателя на первый элемент. Этот указатель обычно устанавливается в начале формирования списка и изменяется только в случае удаления или добавления первого элемента. Указатель на начало служит отправной точкой для всех алгоритмов обработки списка.

Список, соответствующий приведенному описанию, можно изобразить графически:

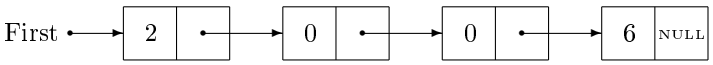

Рис. 4.2.

В этом списке каждый элемент связан только с одним своим соседом, поэтому списки такого типа принято называть односвязными. Если в структуру элемента добавить еще один указатель, то можно организовать двусвязный список, каждый элемент которого будет связан не только со следующим элементом, но и с предыдущим.

Алгоритмы обработки списка удобно анализировать с помощью графического изображения. На рисунке можно более наглядно определить, какие связи и, соответственно, какие поля элементов изменятся в процессе преобразования списка. Удаление элемента из начала списка выглядит следующим образом:

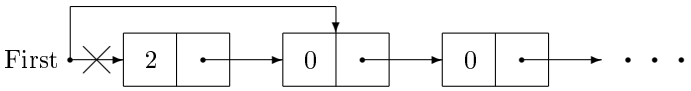

Рис. 4.3.

Удалить элемент из центральной части списка следует так:

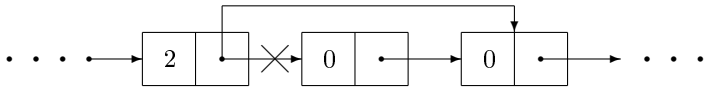

Рис. 4.4.

При работе со списком обычно используется дополнительный указатель, который можно перемещать от одного элемента к дру-

гому. Таких вспомогательных указателей в программе может быть несколько.

Рассмотрим пример включения элемента в список после элемента, на который указывает текущий указатель `u`, включаемый элемент определяется указателем `v`:

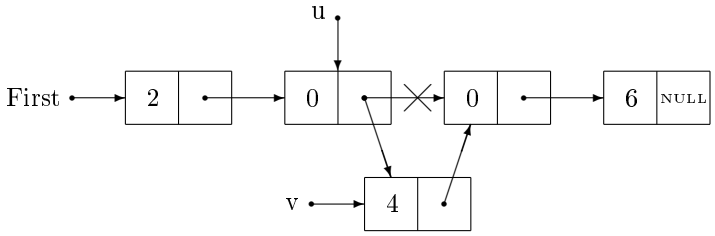

Рис. 4.5.

Операторы, изменяющие при этом список, выглядят так:

```
v->Next=u->Next;
u->Next=v;
```

По окончании работы со списком память, выделенную с помощью функции `malloc()`, следует освободить. Для удаления отдельного элемента используется функция `free()`:

```
free(u);
```

Следует отметить, что изменяя значение полей–указателей элементов списка или освобождая память, важно учитывать порядок применения соответствующих операторов. Ошибки могут привести к некорректному изменению конфигурации списка и выходу за пределы выделенных участков памяти.

Требования к лабораторной работе №4

Данная лабораторная работа предназначена для формирования основных навыков работы с динамическими структурами на примере линейных списков. При разработке программы требуется определить структуру, содержащую поля в соответствии с условием задачи. Формирование списка следует осуществлять с помощью динамического выделения памяти с обязательной проверкой корректности

этой операции. Перед завершением работы программы выделенная память должна быть освобождена. Программа должна корректно обрабатывать все исключительные ситуации, например выдавать соответствующее сообщение, если список пуст .

Задачи

Часть А

А4.1 Написать программу, которая из последовательности целых чисел формирует односвязный список. Длина последовательности может быть произвольной. В списке осуществить следующее преобразование:

1) найти и удалить все четные элементы списка;
2) найти и удалить все элементы списка, равные максимальному;
3) найти минимальный нечетный элемент и переставить его в конец списка;
4) найти и удалить элементы, равные сумме своих соседей;
5) найти элементы списка, которые делятся на 10 и не делятся на 3, и переставить их в начало списка;
6) оставить в списке только элементы из заданного интервала от A до B;
7) удалить из списка первый и последний четный элементы;
8) удалить средний (два средних, если количество элементов в списке четно) элемент списка;
9) найти все элементы, большие среднего арифметического элементов списка и переставить их в начало списка;
10) все нечетные отрицательные элементы переставить в конец списка;
11) четные положительные элементы переставить в начало списка;
12) удалить элементы списка, большие первого его элемента;
13) первые пять минимальных элементов переставить в конец списка;
14) поменять местами пары соседних элементов списка (первый и второй, третий и четвертый, и т.д.); если последним остается только один элемент, то его удалить;
15) в каждой пятерке элементов списка поменять порядок элементов на обратный, в последней группе элементов поменять порядок на обратный независимо от количества элементов;

16) разбить список пополам, если количество элементов в списке нечетное, то первая часть списка должна быть длиннее второй на один элемент, и поменять части местами.
17) все элементы списка, которые больше своего порядкового номера переписать в новый список;
18) все элементы списка из интервала от A до B переписать в новый список;
19) переставить все положительные элементы списка в новый список в порядке убывания;
20) переставить все отрицательные элементы списка в новый список в порядке возрастания;
21) переставить в новый список все элементы списка, меньшие максимального четного элемента.

Часть В

В4.2 Двусвязный список содержит в качестве информационного поля одно данное типа `int`. Требуется организовать процедуру, которая осуществляет «склеивание» двух таких списков, адресуемых указателями `First1` и `First2`, в следующем порядке: сначала идет первый элемент первого списка, затем первый элемент второго списка, затем второй элемент первого списка, второй элемент второго списка, и так далее. При этом, если один из списков окажется длиннее, то требуется:
1) дописать остаток в конец результирующего списка и вывести на печать сообщение о количестве «лишних» элементов;
2) оформить остаток в виде отдельного списка и вывести его содержимое на печать;
3) выбросить остаток списка, освободив при этом занимаемую им память. Итоговый (объединенный) список вывести на печать.

В4.3 Двусвязный список содержит в качестве информационного поля `Info` типа `int`. Список упорядочен по возрастанию этого поля. Это означает, что упорядоченность должен обеспечить программист в процессе ввода, то есть если данные вводятся произвольным образом, то вновь создаваемые элементы списка следует связывать так, чтобы список получался упорядоченным. Требуется организовать процедуру, которая по заданным числам K и L ($K < L$):
1) удаляет из списка все элементы, значение поля `Info` удовлетворяет условию $K <$`Info`$< L$;

2) удаляет из списка все элементы, значение поля `Info` удовлетворяет либо $K \geq$`Info`, либо `Info`$\geq L$. Проверку условия $K < L$ необходимо осуществить при вводе данных.

B4.4 Вводится предложение — набор символов, заканчивающийся точкой (из файла или с клавиатуры по желанию пользователя). Необходимо разбить его на слова (последовательности букв латинского алфавита произвольной длины, все прочие символы считаются разделителями) и сформировать:

1) односвязный,
2) двусвязный

список, информационное поле элементов которого содержит ссылку на очередное слово. Написать программу, которая:

1) упорядочивает слова в списке по алфавиту, затем распечатывает список слов в прямом и обратном порядке;
2) удаляет из списка все пары слов, являющиеся обратными друг другу, например “dog” и “god”;
3) для каждого элемента списка определяет, как часто данное слово встречается в предложении (информация может быть записана в дополнительное поле элемента списка), удаляет повторяющиеся слова, а остальные располагает по убыванию числа повторений каждого слова;
4) оставляет в списке только слова заданной длины, а затем определяет два максимально удаленных друг от друга слова (максимально отличающихся друг от друга), расстоянием между словами равной длины считается количество позиций, в которых соответствующие символы слов различаются между собой;
5) оставляет в списке только те слова, которые соответствуют шаблону, введенному пользователем, шаблон может состоять из латинских букв и символа ‘*’, этот символ означает, что на его месте в слове может стоять произвольная последовательность латинских букв (например, шаблону “*a*” соответствуют слова “rama”, “ma”, “art” и т.д.);
6) удаляет из списка все слова, получающиеся перестановкой букв слова, заданного пользователем.

B4.5 Вводится последовательность, элементом которой являются длины трех отрезков. Если из этих отрезков можно составить треугольник, включить его в список треугольников. Элемент списка должен содержать длины сторон, площадь и тип треугольни-

ка («остроугольный», «тупоугольный», «прямоугольный», «равносторонний», «равнобедренный»). Например, возможные типы треугольника: прямоугольный, прямоугольный и равнобедренный. Из полученного списка удалить треугольники, соответствующие типу, указанному пользователем, и поместить их в новый список.

B4.6 Вводятся два множества точек на плоскости. Каждая точка определяется двумя числами — ее координатами. Построить три списка точек. Первый соответствует пересечению этих множеств, второй — объединению, третий — разности между первым и вторым множествами.

Часть C

C4.7 Вводится множество точек на плоскости (с помощью координат этих точек). Построить список, каждый элемент которого содержит координаты соответствующей точки. Удалить из списка минимальное количество элементов, так чтобы оставшиеся точки лежали на одной прямой.

C4.8 Вводится текст — набор символов, заканчивающийся точкой (из файла или с клавиатуры по желанию пользователя). Программа должна разбить его на слова (последовательности букв латинского алфавита произвольной длины, все прочие символы считаются разделителями) и сформировать несколько списков. В каждом списке информационное поле элементов содержит ссылку на очередное слово, при чем последняя буква каждого слова должна совпадать с первой буквой следующего. Количество списков должно быть минимально возможным.

C4.9 Многочлен хранится в памяти в виде списка, содержащего его ненулевые коэффициенты и соответствующие им степени аргумента. Написать программу, которая осуществляет:

- ввод многочлена;
- вычисление значения многочлена при заданном аргументе (число операций при этом должно быть минимальным);
- сложение и вычитание двух многочленов, результат — третий многочлен;
- умножение двух многочленов;
- деление с остатком многочлена на многочлен, результат — два многочлена: частное и остаток.

5. Файлы

Основные понятия

Для работы с файлами в языке С используются указатели. Файловый указатель — это указатель на структуру `FILE`, которая определена в файле `stdio.h`. Перечислим ряд функций, используемых для работы с файлами:

- `fopen()` — функция открывающая файл;
- `fclose()` — функция закрывающая файл;
- `fprintf()` — функция для форматной записи в файл;
- `fscanf()` — функция для форматного чтения из файла;
- `fputc()` — функция, записывающая в файл один символ;
- `fgetc()` — функция, считывающая из файла один символ.

Описание функции, открывающей файл выглядит так:

`fopen(`*имя файла, режим открытия*`);`

Под именем файла подразумевается текстовая строка, содержащая имя открываемого файла или указатель на массив символов, в котором записано это имя. *Режим открытия* — это строка, содержащая спецификаторы, определяющие, для каких действий открывается файл, например:

- `"rt"` — открыть текстовый файл для чтения;
- `"wt"` — открыть или создать текстовый файл для записи;
- `"at"` — открыть текстовый файл для добавления информации в файл.

Если файл открывается только для записи, существующий файл уничтожается и создается новый, если файл открывается для чтения, необходимо, чтобы он существовал.

Пусть требуется открыть файл `test.txt` только для чтения, тогда следует сначала описать для него файловый указатель:

```
FILE *f;
```

а затем в тексте программы проинициализировать его следующим оператором:

```
f=fopen("test.txt","rt");
```

Открываемый файл `test.txt` должен находиться в том же самом каталоге, что и исполняемый модуль программы, если файл расположен в другой папке, вместе с именем необходимо указать соответствующий путь к файлу.

В случае ошибки, возникшей при открытии файла, функция `fopen()` возвращает нулевой указатель (`NULL`). Любая программа,

использующая эту функцию, должна осуществлять проверку на ошибку открытия файла, если ошибка произошла, выдавать соответствующее сообщение и завершать работу.

Доступ к данным из файла осуществляется с помощью указателя и соответствующих функций. Описание функций форматного чтения и записи следующее:

`fprintf(`*файловый указатель, управляющая строка, список аргументов*`);`

`fscanf(`*файловый указатель, управляющая строка, список аргументов*`);`

Например, прочитать из файла целое число и записать его в переменную соответствующего типа (`int x`) можно с помощью такого оператора:

```
fscanf(f,"%d",&x);
```

Следующий оператор, наоборот, записывает в файл значение переменной `int x`:

```
fprintf(f,"%d",x);
```

После окончания работы с файлом следует закрыть его с помощью функции `fclose()`, аргументом которой является указатель на закрываемый файл.

Если необходимо передать информацию программе одновременно с ее вызовом, например указать имена файлов, с которыми она должна работать, то необходимо использовать два аргумента функции `main()`. В эти аргументы записываются данные из командной строки операционной системы, в которой запускается исполняемый модуль программы. Определить главную функцию программы при этом можно так:

```
void main(int argc, char *argv[])
```

Первый аргумент содержит количество параметров командной строки. Второй является массивом указателей на стороки, каждая из который содержит по одному параметру из командной строки. Имя исполняемого модуля программы рассматривается как первый параметр. Один параметр командной строки отделяется от другого пробелами. Следует отметить, что идентификаторы для имен аргументов функции `main()` могут быть выбраны произвольно, обязательным является только их тип.

Пусть программа запускается из такой командной строки:

```
C:\USER\PROG\example.exe a test.txt
```

Тогда аргументы функции `main()` получат следующие значения:

```
argc=3
argv[0][]="C:\\USER\\PROG\\example.exe"
argv[1][]="a"
argv[2][]="test.txt"
```

Рассмотрим пример программы, в которой открываются два файла — один для чтения (для работы с ним определяется файловый указатель `FA`), другой для записи (ему соответствует файловый указатель `FB`). Содержимое первого файла посимвольно переписывается во второй. Для определения конца файла используется константа `EOF`. Имена файлов передаются через командную строку.

Пример 5.1

```
#include<stdio.h>
FILE *FA,*FB;
void main(int argc, char *argv[])
{
  char c;
 /* Открытие файлов, отсутствует проверка корректности */
  FA=fopen(argv[1],"rt");
  FB=fopen(argv[2],"wt");
 /* Чтение первого символа из файла FA */
  c=fgetc(FA);
 /* Цикл для чтения всех символов из файла FA
    и запись каждого в файл FB */
  while(c!=EOF)
  {
    fputc(c,FB);
    c=fgetc(FA);
  }
 /* Закрытие файлов */
  fclose(FA);
  fclose(FB);
}
```

В этом примере не осуществляется проверка успешности операции открытия файлов, кроме того при использовании аргументов функции `main()` необходимо следить за возможными ошибками пользователя, примером ошибки может быть неверное количество аргументов в командной строке.

Требования к лабораторной работе №5

Данная лабораторная работа предназначена для формирования основных навыков по обработке файлов. Во всех заданиях предполагается, что имена файлов, с которыми имеет дело программа, передаются ей в командной строке (через аргументы функции `main()`). В соответствие с этими условиями, обязательна проверка правильности ввода командной строки пользователем, корректности открытия файлов, и т.д. Если не оговорено обратное, то предполагается, что файлы текстовые. Предполагается также, что для хранения и обработки вводимых данных там, где это диктуется логикой задания, используются структуры.

Задачи

Часть А

А5.1 Написать программу, которая читает строки из файла и выводит их на экран «порциями», не превышающими количество строк, заданное пользователем.

А5.2 Написать программу сбора статистики использования служебных слов в текстах программ на языке С.

А5.3 Написать программу, которая переписывает исходный текстовый файл в результирующий задом наперед.

А5.4 Файл содержит некоторый набор символов. Назовем словом произвольную последовательность латинских букв, разделителями являются все остальные символы. Написать программу, которая:

1) переписывает исходный текстовый файл в результирующий переворачивая все слова задом наперед;
2) определяет и выводит на экран самое длинное слово в файле (если таковых несколько, следует выдать их все);
3) определяет количество симметричных слов (симметричным назовем слово, совпадающее со своим обратным написанием), все найденные слова выводятся на экран;
4) определяет слово с самым большим количеством заглавных букв (если таковых несколько, следует выдать их все);

5) выводит текст из файла на экран, меняя местами каждые два соседних слова;
6) по заданному K определяет количество слов в файле, которые состоят из не более чем K букв, и выводит все такие слова на экран;
7) определяет количество слов, в которых заглавных букв больше, чем маленьких (все найденные слова выводятся на экран).

A5.5 Файл содержит некоторый набор символов. Назовем предложением произвольную последовательность слов из латинских букв, оканчивающееся на '.', '?' или '!'. Все остальные символы являются разделителями. Написать программу, которая:
1) выводит на экран все вопросительные предложения;
2) выводит на экран предложения, состоящие из заданного количества слов.

A5.6 Файл содержит некоторый набор символов. Назовем словом произвольную последовательность латинских букв, разделителями являются все остальные символы. Отредактировать файл путем удаления из него:
1) слов, состоящих из совпадающих букв (например, LLLL);
2) симметричных слов (симметричным назовем слово, совпадающее со своим обратным написанием);
3) слов, в которых нет ни одной гласной буквы ('a', 'e', 'i', 'o', 'u'), все равно: заглавной или маленькой;
4) слов, в которых после маленькой буквы идет заглавная;
5) слов, в которых идут подряд 3 или более одинаковых букв;

A5.7 Написать программу, которая считывает текст из файла и выводит его в новый файл, заменив цифры от 0 до 9 на слова «ноль», «один»,..., «девять».

Часть B

B5.8 Файл содержит некоторый набор символов. Назовем словом произвольную последовательность латинских букв, разделителями являются все остальные символы. Написать программу, которая:
1) определяет для каждого из слов файла — сколько раз оно встречается в файле, результат выводится в отдельный файл и на экран;

2) удаляет из файла слова, которые уже встречались в нем раньше;
3) во всех словах, имеющих окончание ing, заменяет его на ed.

B5.9 Файл содержит некоторый набор символов. Назовем предложением произвольную последовательность слов из латинских букв, оканчивающееся на '.', '?' или '!'. Все остальные символы являются разделителями. Написать программу, которая:
1) выводит в новый файл все предложения в обратном порядке;
2) выводит в новый файл все предложения, меняя в них порядок слов на обратный;
3) выводит в новый файл все предложения, содержащие максимальное количество знаков препинания.

B5.10 Дан файл, компонентами которого являются массивы по N целых чисел. Написать программу, которая:
1) вычисляет количество массивов, упорядоченных по возрастанию, в файл результата выводятся все такие массивы;
2) вычисляет количество массивов, состоящих только из четных элементов, в файл результата выводятся все такие массивы.

B5.11 Дан файл, компонентами которого являются массивы по N целых чисел. Написать программу, в результате выполнения которой в каждом массиве определяется минимальный элемент, после чего выводится один (первый) из массивов, содержащий наибольший из всех минимальных элементов, в файл результата выводятся все такие массивы.

B5.12 Компонентами вводимого файла являются пары вещественных чисел X и Y. Каждая такая пара соответствует точке на плоскости с координатами (X, Y). Написать программу, которая для множества точек, записанного в входном файле, определяет:
1) пару точек с минимальным расстоянием между ними;
2) три разные точки, составляющие треугольник наибольшего периметра;
3) точку, для которой круг заданного радиуса с центром в данной точке содержит максимальное количество точек из данного множества;
4) три разные точки A, B, C такие, что внутри треугольника ABC содержится максимальное количество точек множества;
5) точку, сумма расстояний от которой до остальных минимальна.

B5.13 Файлы FA и FB состоят из символов. Назовем словом произвольную последовательность латинских букв, разделителями являются все остальные символы. Написать программу, которая:

1) находит все слова, которые встречаются и в FA, и в FB;
2) находит самое короткое из слов файла FA, которого нет в файле FB;
3) находит самое длинное общее слово обоих файлов.

B5.14 Компонентами вводимого файла являются пары целых чисел: *Numerator* (числитель) и *Denominator* (знаменатель). Каждая пара задает рациональное число с соответствующим числителем и знаменателем. Числа не обязаны иметь несократимую форму. Написать программу, которая:

1) позволяет определить, есть ли в файле равные числа;
2) вычисляет наибольшее и наименьшее число файла;
3) вычисляет сумму всех чисел (в рациональном виде);
4) вычисляет количество чисел, попадающих в диапазон от -1 до 1;
5) формирует файл результата путем выбрасывания повторяющихся чисел (имеется в виду совпадение по значению);
6) формирует файл результата путем выбрасывания чисел, не попадающих в диапазон от -1 до 1, оставшиеся приводит к виду с положительным знаменателем.

B5.15 В текстовом файле содержится ведомость экзамена группы студентов. Первая строка файла содержит название предмета (30 позиций), вторая — название группы (6 позиций). Каждая следующая строка файла содержит запись об одном студенте. Формат записи:

- фамилия и инициалы (30 позиций), фамилия должна начинаться с первой позиции;
- оценка (число от 2 до 5).

Требуется составить программу, которая формирует файл результата путем переписывания туда исходной ведомости и добавления в конец информации о количестве студентов в группе, количестве студентов получивших каждую оценку, а также средний балл в группе.

B5.16 Файлы FA и FB содержат компоненты целого типа, упорядоченные по возрастанию. Написать программу создания упорядоченного по возрастанию файла FC, компонентами которого являются компоненты файлов FA и FB, причем дубликаты в результирующем файле не допускаются (каждое число должно присутство-

вать в одном экземпляре). Программа должна обеспечивать контроль за правильностью исходных данных. При этом допускается, чтобы исходные файлы содержали одинаковые компоненты.

Часть С

С5.17 Написать программу, шифрующую содержимое файлов в соответствии с введенным паролем и восстанавливающую содержимое только в случае правильно введенного пароля.

С5.18 Написать программу сжатия и последующего восстановления файла.

С5.19 Написать программу приведения исходных текстов на языке C к «каноническому» виду.

С5.20 В файле задано множество окружностей на плоскости. Каждая окружность задается тремя числами: координатами центра и величиной радиуса. Две окружности назовем связанными, если они пересекаются хотя бы в одной точке, либо существует третья окружность из заданного множества, связанная одновременно и с первой окружностью и со второй. Найти и записать в файл результата максимальное подмножество попарно не связанных друг с другом окружностей. Вывести на экран графики окружностей, отметив другим цветом окружности, попавшие в искомое подмножество.

С5.21 В файле задано множество точек в трехмерном пространстве. Найти шар с минимальным радиусом, центр которого находится в одной из точек множества, содержащий ровно n (n — параметр, задаваемый пользователем) точек множества. Информацию о всех шарах, удовлетворяющих условию задачи и соответствующих им точках, вывести в файл результата и на экран в виде таблицы из $(n+2)$ столбцов. В первых двух графах поместить координаты центра шара и его радиус, в остальных координаты точек.

6. Динамические структуры данных

Основные понятия

Во многих задачах невозможно определить заранее, сколько памяти потребуется для хранения используемых данных. Для решения этой проблемы в языке С существуют механизмы, позволяющие выделять память во время работы программы по мере необходимости отдельными блоками и связывать их между собой различными способами при помощи указателей. Такой способ организации данных называется динамическими структурами данных. В программах чаще всего используются линейные списки, стеки, очереди, бинарные деревья. Для описания отдельного элемента таких данных используются структуры.

Линейный список может рассматриваться как простейшая база данных. Каждый элемент списка должен содержать информационную часть, состоящую обычно из нескольких полей, и указатель на следующий элемент (если список является односвязным). Признаком конца списка, как обычно, служит нулевой указатель. При работе с любой базой данных следует уделить внимание интерфейсу программы. Удобнее всего организовать работу с пользователем в виде меню, предоставляющем ему следующие возможности:

1) добавление элемента;
2) удаление элемента;
3) корректировка данных;
4) вывод базы данных на экран;
5) вывод базы данных в файл;
6) выход из программы.

В меню могут быть добавлены и другие пункты, в соответствии с условиями конкретной задачи. Каждый пункт меню обычно оформляется в виде отдельной функции.

Другой динамической структурой, используемой для обработки и хранения данных, является бинарное дерево. В отличие от линейного списка, каждый элемент дерева может быть связан с двумя другими аналогичными элементами, называемыми левым сыном и правым сыном, они в свою очередь могут «иметь» своих сыновей, и т.д. В структуре элемента содержатся информационная часть, ука-

затель на левого сына, указатель на правого сына, иногда указатель на предыдущий элемент дерева (отца):

```
struct Tree {
             int inf;        /*информационное поле*/
             struct Tree *L; /*указатель на левого сына*/
             struct Tree *R; /*указатель на правого сына*/
             struct Tree *F; /*указатель на отца*/
            };
```

Такой способ организации данных удобен для поиска элемета по конкретному значению одного из полей информационной части (обычно требуется уникальность значения этого поля для каждого элемента), если дерево «упорядочено» по этому полю (значение поля у левого сына меньше, чем значение того же поля у отца, а у правого сына больше).

Каждое дерево, так же как и список, должно иметь указатель на элемент, с которого оно начинается (корень).

Легко заметить, что дерево — рекурсивная структура (указатель на сына является указателем на структуру того же типа — поддерево, соответственно левое или правое), поэтому многие алгоритмы работы с деревом являются рекурсивными. Например, алгоритмы обхода дерева:

1) прямой обход — попасть в корень, пройти в прямом порядке левое поддерево, пройти в прямом порядке правое поддерево;
2) обратный обход — пройти в обратном порядке левое поддерево, пройти в обратном порядке правое поддерево, попасть в корень;
3) симметричный обход — пройти в симметричном порядке левое поддерево, попасть в корень, пройти в симметричном порядке правое поддерево.

При обработке динамических структур отдельные действия удобно организовывать в виде самостоятельных функций, например, добавление элемента, удаление элемента, поиск элемента, вывод структуры на экран, и т.п.

Требования к лабораторной работе №6

При разработке программы в данной лабораторной работе необходимо уметь организовать работу с динамическими структурами.

Везде, где требуется организовать обработку какой-либо динамической структуры, предполагается, что формирование этой структуры в памяти производится самим программистом, то есть данные для заполнения вводятся из стандартного устройства ввода и используются для заполнения информационных полей вновь формируемой динамической структуры, а лишь затем происходит необходимая ее обработка. В целях облегчения контроля за правильностью работы программа должна иметь возможность распечатывать промежуточные результаты (например, сформированные списки).

Задачи

Часть А

А6.1 Описать структуру с именем Student, содержащую следующие поля:

- фамилия и инициалы;
- номер группы;
- успеваемость (массив из пяти элементов);
- указатель на следующую структуру, того же типа.

Написать программу, которая читает соответствующие данные из файла, строит в памяти список элементов с информацией о студентах и в процессе его обработки выводит на экран информацию о студентах (фамилия, инициалы, номер группы):

1) средний балл, которых больше четырех;
2) имеющих оценки четыре и пять;
3) имеющих хотя бы одну оценку два;

если таких студентов нет, вывести соответствующее сообщение.

А6.2 Описать структуру с именем Aeroflot, содержащую следующие поля:

- название пункта назначения рейса;
- номер рейса;
- тип самолета;
- указатель на следующую структуру, того же типа.

Написать программу, которая читает соответствующие данные из файла, строит в памяти список элементов с информацией о рейсах и в процессе его обработки выводит на экран информацию:

1) о номерах рейсов и типах самолетов, вылетающих в пункт назначения, название которого вводится пользователем с клавиатуры;
2) о пунктах назначения и номерах рейсов, обслуживаемых самолетом, тип которого введен пользователем с клавиатуры;

если такой информации нет, вывести соответствующее сообщение.

А6.3 Описать структуру с именем Worker, содержащую следующие поля:

- фамилия и инициалы;
- должность;
- год поступления на работу;
- указатель на следующую структуру, того же типа.

Написать программу, которая читает соответствующие данные из файла, строит в памяти список элементов с информацией о работниках и в процессе его обработки выводит на экран информацию о работниках (фамилия, инициалы, должность):

1) чей стаж работы превышает значение, введенное пользователем с клавиатуры;
2) поступивших на работу в заданный период времени;

если таких работников нет, вывести соответствующее сообщение.

А6.4 Описать структуру с именем Train, содержащую следующие поля:

- название пункта назначения;
- номер поезда;
- время отправления;
- указатель на следующую структуру, того же типа.

Написать программу, которая читает соответствующие данные из файла, строит в памяти список элементов с информацией о поездах и в процессе его обработки выводит на экран информацию:

1) о времени отправления и номерах поездов, направляющихся в пункт назначения, название которого вводится пользователем с клавиатуры;
2) о поездах, отправляющихся после введенного пользователем времени;
3) о поезде, номер которого указывает пользователь;

если такой информации нет, вывести соответствующее сообщение.

А6.5 Описать структуру с именем Marsh, содержащую следующие поля:

- название начального пункта маршрута;

- название конечного пункта маршрута;
- номер маршрута;
- указатель на следующую структуру, того же типа.

Написать программу, которая читает соответствующие данные из файла, строит в памяти список элементов с информацией о маршрутах и в процессе его обработки выводит на экран информацию:

1) о маршрутах, которые начинаются или оканчиваются в пункте, название которого вводится пользователем с клавиатуры;
2) о маршруте, номер которого указывает пользователь;

если такой информации нет, вывести соответствующее сообщение.

А6.6 Описать структуру с именем Note, содержащую следующие поля:

- фамилия имя;
- номер телефона;
- дата рождения (массив из трех элементов);
- указатель на следующую структуру, того же типа.

Написать программу, которая читает соответствующие данные из файла, строит в памяти список элементов с информацией из записной книжки и в процессе его обработки выводит на экран информацию:

1) о человеке, номер телефона которого введен с клавиатуры;
2) о людях, чьи дни рождения приходятся на месяц, указанный пользователем;
3) о человеке, чья фамилия введена с клавиатуры;

если такой информации нет, вывести соответствующее сообщение.

А6.7 Описать структуру с именем Znak, содержащую следующие поля:

- фамилия имя;
- знак Зодиака;
- дата рождения (массив из трех элементов);
- указатель на следующую структуру, того же типа.

Написать программу, которая читает соответствующие данные из файла, строит в памяти список элементов с информацией о знаках Зодиака и в процессе его обработки выводит на экран информацию:

1) о людях, родившихся в месяц, название которого введено с клавиатуры;
2) о людях, родившихся под знаком, название которого указано пользователем;
3) о человеке, чья фамилия введена с клавиатуры;

если такой информации нет, вывести соответствующее сообщение.

A6.8 Описать структуру с именем Price, содержащую следующие поля:

- название товара;
- название магазина, в котором продается товар;
- стоимость товара в рублях;
- указатель на следующую структуру, того же типа.

Написать программу, которая читает соответствующие данные из файла, строит в памяти список элементов с информацией о товарах и в процессе его обработки выводит на экран информацию:

1) о товаре, название которого вводится пользователем с клавиатуры;
2) о товарах, продающихся в магазине, название которого указывает пользователь;
3) о товарах, стоимость которых не превышает значение, указанное пользователем;

если такой информации нет, вывести соответствующее сообщение.

Часть B

B6.9 Разреженная матрица `A` (матрица с относительно малым количеством ненулевых элементов) может храниться в памяти в виде набора списков `STR`, как показано на рис. 6.1 на с. 79.

Каждый ненулевой элемент матрицы хранится в структуре, имеющей три поля: `int Val;` — значение элемента, `int Row;` — номер столбца, `Next` — указатель на следующий ненулевой элемент строки. Компонета `STR[i]` массива `STR` — это указатель на первый ненулевой элемент `i`-й строки. Программа должна ввести матрицу `A` (или две матрицы - в зависимости от задания) из файла, в котором она хранится:

1) в разреженном виде,
2) в виде полной матрицы,

разместить ее в памяти в соответствии с рис. 6.1 на с. 79 и произвести следующее преобразование:

1) найти и обнулить (то есть выбросить) минимальный элемент, если таких несколько, то выбросить все;
2) найти и обнулить (то есть выбросить) элемент с максимальной суммой соседей справа и слева (для первого и последнего элемента в строке, естественно, определен только один сосед),

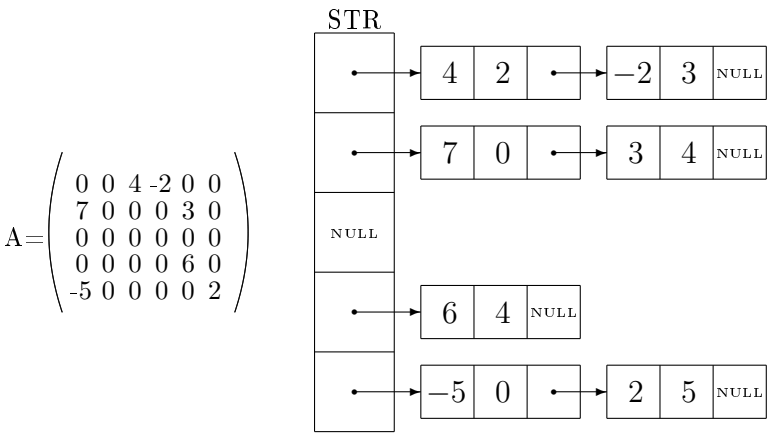

Рис. 6.1.

если таких элементов несколько, то выбросить следует последних из встретившихся;

3) найти номер столбца, в котором находится самая длинная последовательность ненулевых элементов;
4) вычислить сумму двух разреженных матриц, результат также должен быть представлен в виде разреженной матрицы;
5) умножить разреженную матрицу на вектор, исходный вектор и вектор-результат хранятся в памяти в виде массива;
6) транспонировать матрицу, результат должен быть представлен в виде разреженной матрицы;
7) проверить наличие седловой точки, то есть проверить равенство

$$\min_i \max_j \{A_{ij}\} = \max_j \min_i \{A_{ij}\}.$$

В6.10 Написать программу, моделирующую заполнение гибкого магнитного диска. Общий объем памяти на диске задается пользователем. Файлы имеют произвольную длину. В процессе работы файлы либо записываются на диск, либо удаляются с него. В начале работы файлы записываются подряд друг за другом. После удаления файла на диске образуется свободный участок памяти, и вновь записываемый файл либо размещается на свободном участке, либо, если не помещается, дописывается после последнего записанно-

го файла. Требование на запись или удаление файла содержит имя файла, его длину в байтах, признак записи или удаления. Программа должна выдавать по запросу сведения о занятых (в т.ч. список соответствующих файлов) и свободных участках памяти на диске.

B6.11 Каталог файлов организован в виде линейного списка. Для каждого файла содержатся следующие сведения:

- имя файла;
- дата и время создания;
- размер в байтах.

Написать программу, которая обеспечивает:

- начальное формирование каталога файлов;
- вывод каталога файлов, упорядоченного по заданному параметру;
- удаление файлов, дата создания которых меньше заданной.

Часть C

C6.12 Бинарное дерево - это динамическая структура данных, имеющая вид, представленный на рис. 6.2 на с. 81. Каждый узел задается в программе переменной типа структура, имеющей три части: `LS`, `Info` и `RS`. Здесь `Info` — это информационная составляющая (может быть представленная в виде нескольких полей), а `LS` и `RS` — указатели на левого и правого сына соответственно. Как обычно, если сына нет, то соответствующий указатель имеет нулевое значение. Пусть поле `Info` представляет из себя строку символов длиной не более 10. Требуется разработать способ представления дерева в текстовом файле и написать программу, которая считывает этот файл, формирует в памяти дерево и вызывает функцию, которая выводит на экран значения поля `Info` узлов дерева в виде аналогичном рис. 6.2 на с. 81, при этом узлы дерева появляются на экране в порядке, соответствующем:

- прямому обходу,
- обратному обходу,
- симметричному обходу.

Параметром функции должна являться переменная `ROOT` — указатель на корень дерева.

C6.13 Узлы бинарного дерева (см. текст предыдущего задания) в качестве информационного поля имеют значение целого беззнакового типа, которое имеет смысл удаленности данного узла от своего

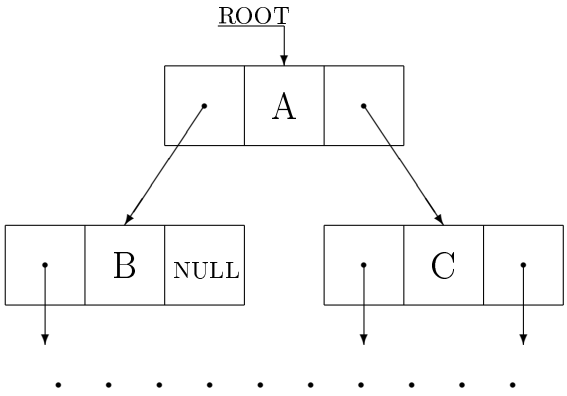

Рис. 6.2.

отца. Требуется разработать способ представления дерева в текстовом файле и написать программу, которая считывает этот файл, формирует в памяти дерево и вычисляет:

1) расстояние от корня до самого удаленного узла,
2) расстояние от корня до самого близкого листьевого узла,
3) диаметр дерева, то есть расстояние между двумя самыми удаленными друг от друга узлам,

и выводит на экран соответствующую картинку.

С6.14 Написать программу, которая формирует и обрабатывает базу данных о книгах в библиотеке. Сведения о книге включают:

- номер УДК (код жанра);
- фамилия инициалы автора;
- название;
- год издания;
- количество экземпляров книги в библиотеке.

Программа должна обеспечивать:

- добавление данных с контролем ошибок;
- изменение данных с контролем ошибок;
- удаление данных;
- поиск по различным параметрам;
- формирование сведений о книгах, упорядоченных по параметру, заданному пользователем.

7. Дополнительные задачи

В данном разделе собраны задачи для студентов, успешно освоивших обязательную часть курса и желающих изучить дополнительные возможности языка C.

Задачи

D7.1 Программа, выводящая на экран правильный календарь на любой вводимый год.

D7.2 Разработать калькулятор, осуществляющий арифметические действия с числами сколь угодно большой длины.

D7.3 Написать программу, выводящую на экран циферблат идущих механических часов с секундной стрелкой.

D7.4 Написать программу, создающую таблицу значений функции и строящую ее график. Вместе с графиком функции другим цветом следует вывести график ее 1-й производной, либо цвет графика функции должен изменяться в зависимости от значения 1-й производной. Таблица должна содержать значения аргумента, функции и ее 1-й производной на интервале ее построения, не менее 10 значений. Таблица должна быть на том же экране, что и график функции.

D7.5 Разработать программу-тренажер работы на клавиатуре.

D7.6 Текстовый редактор.

D7.7 Графический редактор.

D7.8 Нотный редактор.

D7.9 Разработать синтаксический анализатор математических формул.

D7.10 Написать одну из следующих игр:
1) «морской бой>,>
2) «тетрис»,
3) «шарики»,
4) «удав»,
5) «ветка»,
6) «крестики-нолики» на бесконечном поле,
7) «сапер».

Литература

[1] Белецкий Я. Энциклопедия языка Си.М.: Мир, 1992.

[2] Березин Б.И., Березин С.Б. Начальный курс С и С/++. М.: Диалог–МИФИ, 1998.

[3] Берри Р., Микинз Б. Язык Си. Введение для программистов.М.: Финансы и статистика, 1988.

[4] Болски М.И. Язык программирования Си. Справочник. М.: Радио и связь, 1988.

[5] Васильчиков В.В. Библиотека С. Ярославль. ЯрГУ, 2003.

[6] Керниган Б., Ритчи Д. Язык программирования Си. М.: Финансы и статистика, 1992.

[7] Керниган Б., Ритчи Д., Фьюэр А. Язык программирования Си. Задачи по языку Си. М.: Финансы и статистика, 1985.

[8] Павловская Т.А., Щупак Ю.А. С/С++. Структурное программирование. Практикум. СПб.: Питер, 2004.

[9] Подбельский В.В., Фомин С.С. Программирование на языке Си. М.: Финансы и статистика, 2003.

[10] Подбельский В.В. Практикум по программированию на языке Си. М.: Финансы и статистика, 2004.

[11] Уинер Р. Язык Турбо Си. М.: Мир, 1991.

[12] Шилдт Г. Полный справочник по С. 4–е издание. М.;СПб.;Киев: Издательский дом "Вильямс", 2004.

[13] Яншин В.В., Калинин Г.А. Обработка изображений на языке Си для IBM PC. Алгоритмы и программы. М.: Мир, 1994.

Оглавление

Учебное издание

Васильчиков Владимир Васильевич
Лагутина Надежда Станиславовна
Ларина Юлия Александровна

Основы программирования на языке С

Учебное пособие

Редактор, корректор А.А. Антонова
Компьютерный набор, верстка Н.С. Лагутина

Подписано в печать .04.2006 г. Формат 60×84/16.
Бумага тип. Усл. печ. л. 5,0. Уч.-изд.л. 5,0
Тираж 200 экз. Заказ

Оригинал–макет подготовлен в редакционно–издательском отделе
Ярославского государственного университета.

Отпечатано на ризографе ЯрГУ

Ярославский государственный университет
150000, Ярославль, ул. Советская, 14.