МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное
образовательное учреждение высшего образования
«Юго-Западный государственный университет»
(ЮЗГУ)

Кафедра космического приборостроения и средств связи

УТВЕРЖДАЮ
Проректор по учебной работе
_____О.Г.Локтионова
«__» _____2017 г.

ВВЕДЕНИЕ В ПРОГРАММИРОВАНИЕ НА ЯЗЫКЕ СИ

Методические указания по выполнению лабораторных работ
для студентов направления подготовки 11.03.02, 11.03.03

Курск 2017

УДК 681.3

Составитель В.Н. Усенков

Рецензент

Кандидат технических наук, доцент *Т.М. Белова*

Введение в программирование на языке Си: методические указания по выполнению лабораторных работ / Юго-Зап. гос. ун-т; сост.: В.Н. Усенков. - Курск, 2017. - 50 с.: ил. 1, прилож. 3. - Библиогр.: с. 27.

Приводятся краткие сведения об используемых версиях языка Турбо Си. Приводятся методические указания по проектированию программ на языке Си, ориентированных для применения в системах, построенных на базе микро-ЭВМ. Указывается порядок выполнения лабораторных работ. Приводятся рекомендации по оформлению отчетов и контрольные вопросы.

Методические указания соответствуют требованиям программы, утвержденной учебно-методическим объединением по специальностям автоматики и электроники (УМО АЭ).

Предназначены для студентов специальностей 11.03.02, 11.03.03 дневной и заочной форм обучения.

Текст печатается в авторской редакции

Подписано в печать . Формат 60×84 1/16.
Усл. печ. л. *2,91*. Уч.-изд. л. *2,63*. Тираж 100 экз. Заказ. Бесплатно.

Юго-Западный государственный университет.
305040, г. Курск, ул. 50 лет Октября, 94.

Содержание

1. Цель работ

- Изучение средств разработки и отладки программного обеспечения для микропроцессорных систем;
- Изучение особенностей разработки программного обеспечения на языке С;
- формирование навыков работы в среде турбо С.

2. Введение

Простые программные проекты для встраиваемых систем на базе микроЭВМ пишутся, как правило, на языке ассемблера. Для более сложных проектов более адекватным представляется использование трансляторов с языков высокого уровня, причем в последнее время предпочтение чаще отдается языку С.

Для практического освоения программирования на языке С удобно использовать программный продукт Турбо С. Это, фактически, система программирования, содержащая развитые отладочные средства и предоставляющая удобства работы в интегрированной оболочке.

При переходе на трансляторы С, специально ориентированные на промышленные микроЭВМ, будут, конечно заметны отличия. Прежде всего, в большинстве случаев используется кросс трансляция на персональных компьютерах, что влечет за собой усложнение этапа прогона программных продуктов. Кроме того, будут наблюдаться отличия в составе и объеме библиотечных функций. Тем не менее, традиционная мобильность программ на языке С предопределяет сравнительно легкую адаптацию к новым условиям.

В лабораторных работах используются программные продукты Турбо С версий 1.5 или 2.0, выполняемые на IBM PC компьютерах.

3. Лабораторная работа 1 "Система программирования Турбо С"

3.1. Цель работы

изучение возможностей интегрированной среды Турбо С и подготовка к проектированию программ на языке С.

3.2. Подготовка к работе

Изучить основные особенности Турбо С:
- встроенный редактор
- создание проекта
- трансляция программы
- запуск программы для отладки

3.3 Вопросы для самоконтроля

Можно ли использовать внешний редактор для формирования текста программы?
Имеются ли отличия языка Турбо С от стандарта ANSI?
Как к тексту программы подключаются библиотеки?
Как запускаются на исполнение программы?

3.4 Программа работ

Подготовить рабочую папку для сохранения рабочих файлов
Проверить наличие в рабочей папке Турбо С всех необходимых файлов
Загрузить интегрированную среду Турбо С.
Создать новый проект для отладки.
Включить в проект текст программы с выбранным именем
Набрать текст демонстрационной программы, предложенный преподавателем или выбранный самостоятельно.
Провести компиляцию программы и устранить при необходимости выявленные транслятором ошибки.

Отработать проект.
Запустить полученную программу и наблюдать процесс ее выполнения.

3.5 Методические указания

Простая программа, пригодная для первых шагов освоения языка, выполняет вывод на экран текстовой строки, заданной в виде константы:

```
main()
{
  printf(“This is simple TEXT”);
}
```

В приложении 2 имеется подробное описание библиотечной функции printf() и некоторых других часто используемых функций.

Так, совместно с функцией printf() часто применяют функцию форматированного ввода scanf():

```
main()
{
  char ss[30];

  puts(“Vvedite imja: \n”);
  scanf(“%s”,ss);
  printf(“\n Spasibo,  %s”,ss);
}
```

3.6. Содержание отчета

Отчет должен содержать:
- титульный лист с указанием темы работы;
- цель работы;
- текст программы, снабженный содержательными комментариями;
- протокол работы, содержащий скриншоты для основных этапов работ;
- ответы на контрольные вопросы.

4. Лабораторная работа 2 "Средства отладки в Турбо С "

4.1. Цель *работы*

Изучение базовых средств отладки программ, существующих в прототипных системах.

4.2. Подготовка к работе

Изучить основные особенности отладчика Турбо С [1]

4.3. Вопросы для самоконтроля

Каковы основные возможности отладчика?
Как установить и снять точки останова?
Как разрешить проблему "зависания" программы в процессе отладки?

4.4. Программа работ

Загрузить интегрированную среду
Загрузить проект с указанной преподавателем программой для отладки.
Слежение за ходом выполнения программы
Работа с точками останова
Квалификация идентификаторов
Вывод значений выражений
Слежение за вызовами

4.5. Методические указания

Для выполнения лабораторный работы необходимо изучить описание пунктов меню (рис.1):
- Debug
- Break/watch

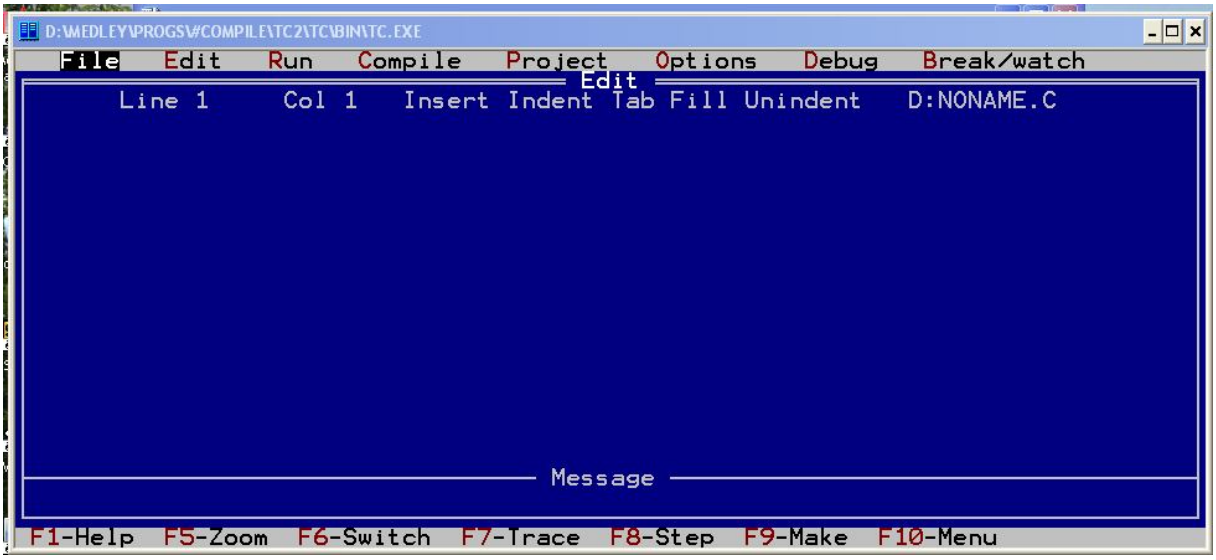

Рис. 1. Главное меню программы

Все необходимые сведения подробно описаны в [1]. Текст демонстрационной программы, содержащей ошибки, подлежащие нахождению, выдается преподавателем. Типичный пример приведен ниже.

```
main()
{
  Int n;

  puts(“Vvedite 4islo: \n”)
  scanf(“%d”,n);
  printf(“/n Vvedeno:  %d”,nn);
}
```

Необходимо выделить ошибки:
- обнаруживаемые при компиляции программы
- обнаруживаемые при отладке программы

4.6. Контрольные вопросы

- Как влияет подключение к проекту отладочных средств на программный код?
- Можно ли отлаживать программы, сгенерированные вне интегрированной среды (TCC)?

- В чем заключается сущность процесса квалификации идентификаторов?
- Установите несколько точек останова и продемонстрируйте процесс отладки, продолжая выполнение после каждой точки до конца программы.
- Покажите значения переменных на каждой из точек останова.
- Как исключить одну из нескольких точек останова на этапе подготовки к отладке?

4.7. Содержание отчета

Отчет должен содержать:
- титульный лист с указанием темы работы;
- цель работы;
- текст программы, снабженный содержательными комментариями;
- протокол работы, содержащий скриншоты для основных этапов работ;
- ответы на контрольные вопросы.

5. Лабораторная работа 3 "Ввод/вывод данных в С"

5.1. Цель работы

Приобретение практических навыков проектирования простых программ, содержащих процедуры ввода/вывода на консоль.

5.2. Подготовка к работе

Изучить функционирование основных библиотечных функций ввода/вывода [1], [2]:
Printf()
Puts()
Scanf()
Getch()
Putch()
Разработать алгоритм решения поставленной преподавателем задачи
Написать программу в соответствии с заданием и создать файл для отладки.

5.3. Вопросы для самоконтроля

- приведите примеры других функций ввода/вывода, кроме предложенных для изучения
- в чем основные отличия между функциями printf() и puts()
- можно ли вводить значения для нескольких переменных в рамках одного вызова scanf()
- можно ли выводить значения для нескольких переменных в рамках одного вызова printf()

5.4. Программа работ

Загрузить интегрированную среду

Загрузить проект с указанным преподавателем заданием для отладки.
Пользуясь отладочными средствами, добиться работоспособности программы

5.5. Методические указания

Предполагается решение сравнительно простых задач, требующих информационного обмена с консолью. Кроме функций ввода/вывода требуется использовать языковые средства для организации ветвления, циклов и прочих стандартных действий.

Типичное задание выглядит следующим образом:

- написать и отладить программу, запрашивающую в бесконечном диалоге у пользователя операнд, возводящую полученное значение в квадрат и выводящую результат на экран. Требуется обеспечить вывод их диалога.

При проектировании программы необходимо обратить внимание на то, чтобы диалог с пользователем был интуитивно понятным, а пользование программой было простым и естественным.

В приводимом тексте программы обязательно наличие содержательных комментариев.

В процессе защиты нужно быть готовым к корректировке программы на предмет изменения выполняемой функции (например, извлечение корня вместо возведения в степень), изменения типа данных (целочисленные или с плавающей точкой), изменения формата выводимого результата (десятичное целое, шестнадцатеричное целое, плавающая точка во всех допустимых форматах и.т.п.)

5.6. Содержание отчета

Отчет должен содержать:
- титульный лист с указанием темы работы;
- цель работы;

- текст программы, снабженный содержательными комментариями;
- протокол работы, содержащий скриншоты для основных этапов работ;
- ответы на контрольные вопросы.

6. Лабораторная работа 4 "Использование управляющих ESC – последовательностей ANSI в текстовом режиме вывода на экран"

6.1. Цель работы

Приобретение практических навыков проектирования простых программ, поддерживающих полноценный интерфейс с пользователем в текстовом режиме экрана.

6.2. Подготовка к работе

Изучить описание возможностей драйвера ANSI для вывода текстовых сообщений :
Разработать алгоритм решения поставленной преподавателем задачи
Написать программу в соответствии с заданием и создать файл для отладки.

6.3. Вопросы для самоконтроля

- в каких случаях целесообразно использовать текстовый режим вывода на экран
- возможен ли перенос программы, использующей ESC-последовательности, в альтернативные операционные системы
- описать альтернативные возможности среды Турбо С для организации пользовательского интерфейса (в текстовом и графическом режимах)

6.4. Программа работ

Загрузить интегрированную среду
Загрузить проект с указанным преподавателем заданием для отладки.
Пользуясь отладочными средствами, добиться работоспособности программы

6.5. Методические указания

При написании программы возможно ограничиться двумя примитивами:
- очистка экрана;
- позиционирование курсора на экране.

Пример реализации примитивов и их использования приведен ниже.

```
/*****************************************************************/
/* This prog demonsrates scren-controle via ANSI.SYS driver      */
/* Don't forget to add line "device=ansi.sys" into config file   */
/* Works under WinXP as well  ( modify config.nt }               */
/*****************************************************************/

void E_clear(void)
{
 printf("\x1b[2J");
}

void E_pos(int row, int col)
{
 char ss[20];
 char s2[30];

 strcpy(s2,"");
/* sprintf(ss,"[");*/
 sprintf(ss,"\x1b[");
 strcat(s2,ss);
 sprintf(ss,"%d",row);
 strcat(s2,ss);
 sprintf(ss,";");
 strcat(s2,ss);
 sprintf(ss,"%d",col);
 strcat(s2,ss);
 sprintf(ss,"H");
 strcat(s2,ss);
 strcat(s2,"\x0");
 printf("%s",s2);
}

main()
{
 /* ESC[2J CLS */
 E_clear();
 printf("12345678901234567890123456789O\n");
 /* ESC[row;colH set cursor position */
 printf("\x1b[2;12H2-12\n");
 E_pos(5,23);
/* puts("5-23");*/
 getch();
}
```

Задание для выполнения лабораторной работы выдается преподавателем. Типичный пример задания:

- написать и отладить программу, перемещающую в бесконечном диалоге с пользователем символьную строку на экране. Направление перемещения определяется нажатием клавиш:
- 1 – влево;
- 2 - вправо;
- 3 -вверх;
- 4 - вниз.

. Требуется обеспечить выход их диалога.

При проектировании программы необходимо обратить внимание на то, чтобы диалог с пользователем был интуитивно понятным, а пользование программой было простым и естественным.

В приводимом тексте программы обязательно наличие содержательных комментариев.

В процессе защиты нужно быть готовым к корректировке программы с целью изменения выполняемой функции.

6.6. Содержание отчета

Отчет должен содержать:
- титульный лист с указанием темы работы;
- цель работы;
- текст программы, снабженный содержательными комментариями;
- протокол работы, содержащий скриншоты для основных этапов работ;
- ответы на контрольные вопросы.

7. Лабораторная работа 5 "Работа с последовательными файлами в языке С".

7.1. Цель работы

Приобретение практических навыков проектирования простых программ, содержащих процедуры файлового ввода/вывода.

7.2. Подготовка к работе

Изучить функционирование основных библиотечных функций ввода/вывода [1], [2]:
fopen();
fclose();
feof();
fprintf()
fputs()
fscanf()
fgetc()
fputc()
Разработать алгоритм решения поставленной преподавателем задачи
Написать программу в соответствии с заданием и создать файл для отладки.

7.3. Вопросы для самоконтроля

- как определить конец файла при последовательном чтении;
- в чем основные отличия между функциями printf() и fprintf();
- как гарантированно записать данные на диск до завершения программы;
- чем отличаются режимы открытия файла (опции открытия r,w,…)
- опишите особенности текстовых файлов

7.4. Программа работ

Загрузить интегрированную среду
Загрузить проект с указанным преподавателем заданием для отладки.
Пользуясь отладочными средствами, добиться работоспособности программы

7.5. Методические указания

Предполагается решение сравнительно простых задач, требующих сохранения и считывания данных на жесткий диск. Данные, с которыми предстоит работать, считаются текстовыми (т.е., все символы соответствуют текстовой части кодовой таблицы).

Для закрепления знаний по ранее выполненным работам, повторно используются ранее изученные средства языка С.

7.6. Порядок выполнения

- написать и отладить программу, запрашивающую в бесконечном диалоге у пользователя операнды и типы выполняемых операций. Результаты расчетов выводятся на экран, текстовые данные записываются в файл или считываются из файла. Длина текстовых данных ограничена одной строкой длиной не более 250 символов.

Перечень операций:
- арифметические функции: сложение, вычитание, умножение, деление;
- тригонометрические функции: sin, cos;
- ввод с консоли текстовой строки и сохранение ее в файле с предопределенным именем;
- считывание текстовой строки из файла с предопределенным именем и отображение ее на экране;
- выход из программы.

При проектировании программы необходимо обратить внимание на то, чтобы диалог с пользователем был интуитивно понятным, а пользование программой было простым и естественным.

В приводимом тексте программы обязательно наличие содержательных комментариев.

В процессе защиты нужно быть готовым к корректировке программы на предмет изменения выполняемой функции (например, извлечение корня вместо тригонометрических функций), изменения типа данных (целочисленные или с плавающей точкой), изменения формата выводимого результата (десятичное целое, шестнадцатеричное целое, плавающая точка во всех допустимых форматах и.т.п.)

7.7. Содержание отчета

Отчет должен содержать:
- титульный лист с указанием темы работы;
- цель работы;
- текст программы, снабженный содержательными комментариями;
- протокол работы, содержащий скриншоты для основных этапов работ;
- ответы на контрольные вопросы.

8. Лабораторная работа 6 " Работа с текстовыми файлами в языке С. Редактирование и обработка текстовых файлов".

8.1. Цель работы

Приобретение практических навыков проектирования простых программ для обработки и редактирования текстовых файлов.

8.2. Подготовка к работе

Изучить функционирование основных библиотечных функций для работы с файлами и текстовыми объектами [1], [2]:
fopen(), fclose(), feof(), fprintf(), fscanf(), fgetc(), fputc(),…;
strcat(), strcmp(), strcpy(),…
sprintf(), sscanf(),…

Разработать алгоритм решения поставленной преподавателем задачи
Написать программу в соответствии с заданием и создать файл для отладки.

8.3. Вопросы для самоконтроля

- какие дополнительные фукнции работы со строками, кроме упомянутых ранее, имеются в библиотеках С;
- для достижения каких целей применяют функции sprintf(), sscanf();
- как преобразовать символьную содержащую цифры строку в числовой вид;
- как преобразовать число в символную строку;
- как выделить динамический буфер для редактирования текстовой строки;
- опишите особенности текстовых файлов.

8.4. Программа работ

Загрузить интегрированную среду
Загрузить проект с указанным преподавателем заданием для отладки.
Пользуясь отладочными средствами, добиться работоспособности программы

8.5. Методические указания

Предполагается решение усложненных задач по отношению к ранее разработанным, требующих сохранения и считывания данных на жесткий диск. Данные, с которыми предстоит работать, считаются текстовыми (т.е., все символы соответствуют текстовой части кодовой таблицы).

Для закрепления знаний по ранее выполненным работам, повторно используются ранее изученные средства языка С. Разрабатываемая программа должна быть развитием предыдущих работ.

8.6. Порядок выполнения лабораторной работы

- написать и отладить программу, выполняющую типичные функции органайзера, содержащего калькулятор и записную книжку. Ограничения и требования:
- арифметические функции: сложение, вычитание, умножение, деление;
- тригонометрические функции: sin, cos;
- редактирование текстовых файлов, содержащих не более 8 строк длиной 250 символов каждая;
- предусмотреть средства задания и изменения имени файла ;
- выход из программы.

При проектировании программы необходимо обратить внимание на то, чтобы диалог с пользователем был интуитивно понятным, а пользование программой было простым и естественным.

В приводимом тексте программы обязательно наличие содержательных комментариев.

В процессе защиты нужно быть готовым к корректировке программы на предмет изменения выполняемой функции, количества функций и параметров текстовых файлов.

В программе необходимо применить оператор switch и разновидности циклов.

Кроме того, программа должна состоять из нескольких пользовательских функций.

8.7. Содержание отчета

Отчет должен содержать:

- титульный лист с указанием темы работы;
- цель работы;
- текст программы, снабженный содержательными комментариями;
- протокол работы, содержащий скриншоты для основных этапов работ;
- ответы на контрольные вопросы.

9. Лабораторная работа 7 " Работа с пользовательскими объектами, построенными с использованием структур и объединений"

9.1. Цель работы

Приобретение практических навыков проектирования программ, содержащих более сложные виды данных с использованием для этого таких средств языка С, как структуры или объединения.

9.2. Подготовка к работе

Изучить особенности построения пользовательских объектов [1], [2]:
struct;
union;
enum;
typedef.

Разработать алгоритм решения поставленной преподавателем задачи
Написать программу в соответствии с заданием и создать файл для отладки.

9.3. Вопросы для самоконтроля

- в каких случаях используется средство языка С struct;
- в каких случаях используется средство языка С union;
- в чем удобство использования typedef;
- как передать пользовательские данные для обработки в пользовательскую функцию;
- можно ли применить динамически выделяемый пул памяти, с элементами пользовательской структуры данных.

9.4. Программа работ

Загрузить интегрированную среду
Загрузить проект с указанным преподавателем заданием для отладки.
Пользуясь отладочными средствами, добиться работоспособности программы

9.5. Методические указания

Предполагается решение усложненных задач по отношению к ранее разработанным, требующих сохранения и считывания данных на жесткий диск. Данные, с которыми предстоит работать, должны содержать разнородные компоненты, как числовые, так и символьные.

Для закрепления знаний по ранее выполненным работам, повторно используются ранее изученные средства языка С. Разрабатываемая программа должна быть развитием предыдущих работ.

9.6. Порядок выполнения лабораторной работы

- написать и отладить программу, выполняющую типичные функции органайзера, содержащего калькулятор, записную книжку, базу телефонных данных. Ограничения и требования:
- арифметические функции: сложение, вычитание, умножение, деление;
- тригонометрические функции: sin, cos;
- редактирование текстовых файлов, содержащих не более 8 строк длиной не более 250 символов каждая;
- предусмотреть средства задания и изменения имени файла ;
- считывание и корректировка записей телефонной книги, содержащей не более 20 элементов, каждый из которых содержит ФИО, номер телефона, заметки, рейтинг(некоторая числовая величина);
- выход из программы.

При проектировании программы необходимо обратить внимание на то, чтобы диалог с пользователем был интуитивно понятным, а пользование программой было простым и естественным.

В приводимом тексте программы обязательно наличие содержательных комментариев.

В процессе защиты нужно быть готовым к корректировке программы на предмет изменения выполняемой функции, количества функций и параметров текстовых файлов.

9.7. Содержание отчета

Отчет должен содержать:
- титульный лист с указанием темы работы;
- цель работы;
- текст программы, снабженный содержательными комментариями;
- протокол работы, содержащий скриншоты для основных этапов работ;
- ответы на контрольные вопросы.

10. Лабораторная работа 8 " Работа с графическими средствами среды Турбо С "

10.1. Цель работы

Приобретение практических навыков проектирования программ, содержащих графические растровые объекты.

10.2. Подготовка к работе

Изучить библиотечные функции для работы в графическом режиме [1], [2]:
Разработать алгоритм решения поставленной преподавателем задачи
Написать программу в соответствии с заданием и создать файл для отладки.

10.3. Вопросы для самоконтроля

- какие удобства предоставляет графический интерфейс;
- какие графические режимы персонального компьютера доступны в среде проектирования Турбо С;
- как переключаться между текстовым и графическим режимами;
- как построить график математических зависимостей в графическом окне.

10.4. Программа работ

Загрузить интегрированную среду
Загрузить проект с указанным преподавателем заданием для отладки.
Пользуясь отладочными средствами, добиться работоспособности программы

10.5. Методические указания

При выполнении данной работы допускается как написание отдельной программы, так и доработка ранее спроектированной. Во втором варианте необходимо предусмотреть переключение между текстовым и графическим режимами работы. Графический режим выбирается разработчиком.

10.6. Порядок выполнения лабораторной работы

- написать и отладить программу, выводящую график некоторой математической функции на экран в графическом режиме.
- выбор из функции: sin, cos, степень, логарифм;
- запрос интервала горизонтальной оси;
- прорисовка рамки окна вывода и делений осей;
- вывод сообщений в окне (например, запроса на завершение)
- выход из программы.

При проектировании программы необходимо обратить внимание на то, чтобы диалог с пользователем был интуитивно понятным, а пользование программой было простым и естественным.

В приводимом тексте программы обязательно наличие содержательных комментариев.

В процессе защиты нужно быть готовым к корректировке программы на предмет изменения выполняемой функции.

10.7. Содержание отчета

Отчет должен содержать:
- титульный лист с указанием темы работы;
- цель работы;
- текст программы, снабженный содержательными комментариями;
- протокол работы, содержащий скриншоты для основных этапов работ;
- ответы на контрольные вопросы.

11. Литература

1. Уинер, Р. Язык турбо си [Текст] / Пер. с англ.; Под ред. В. В. Мартынюка. - Москва. : Мир, 1991. - 380 с.

2. Белецкий, Я. Энциклопедия языка СИ [Текст] : пер. с польского / А. Д. Плитман, М. Ю. Рачков, А. В. Стрельцова; Под ред. Ф. Ф. Пащенко. - Москва. : Мир, 1992. - 687 с. : ил.

Приложение А - Кодировка символов для IBM PC

Альтернативная кодировка – это кодовая страница IBM, где все специфические европейские символы в верхней половине замененяются на кириллицу, оставляя псевдографические символы нетронутыми. Следовательно, это не портит вид программ, использующих для работы текстовые окна, а также обеспечивает использование в них символов кириллицы. Альтернативная кодировка все ещё жива и чрезвычайно популярна в среде MS-DOS и OS/2. Кроме того, в этой кодировке записываются имена в файловой системе FAT (и короткие имена в VFAT).

Исторически существовало много вариантов альтернативной кодировки, но все различия касаются только области 0xF0 — 0xFF (240—255). Окончательным стандартом стала кодировка 866, поддержка которой была добавлена в MS-DOS версии 6.22 (до этого использовались всевозможные «самодельные» русификаторы).

CP866

Нижняя часть таблицы кодировки (латиница) полностью соответствует кодировке ASCII. В приведённых таблицах числа под буквами обозначают шестнадцатеричный код буквы в Уникоде.

	.0	.1	.2	.3	.4	.5	.6	.7	.8	.9	.A	.B	.C	.D	.E	.F
8.	А 410	Б 411	В 412	Г 413	Д 414	Е 415	Ж 416	З 417	И 418	Й 419	К 41A	Л 41B	М 41C	Н 41D	О 41E	П 41F
9.	Р 420	С 421	Т 422	У 423	Ф 424	Х 425	Ц 426	Ч 427	Ш 428	Щ 429	Ъ 42A	Ы 42B	Ь 42C	Э 42D	Ю 42E	Я 42F
A.	а 430	б 431	в 432	г 433	д 434	е 435	ж 436	з 437	и 438	й 439	к 43A	л 43B	м 43C	н 43D	о 43E	п 43F
B.	▒ 2591	▒ 2592	▓ 2593	┃ 2502	┤ 2524	╡ 2561	╢ 2562	╖ 2556	╕ 2555	╣ 2563	║ 2551	╗ 2557	╝ 255D	╜ 255C	╛ 255B	┐ 2510
C.	└ 2514	┴ 2534	┬ 252C	├ 251C	─ 2500	┼ 253C	╞ 255E	╟ 255F	╚ 255A	╔ 2554	╩ 2569	╦ 2566	╠ 2560	═ 2550	╬ 256C	╧ 2567
D.	╨ 2568	╤ 2564	╥ 2565	╙ 2559	╘ 2558	╒ 2552	╓ 2553	╫ 256B	╪ 256A	┘ 2518	┌ 250C	▉ 2588	▄ 2584	▌ 258C	▐ 2590	▀ 2580
E.	р 440	с 441	т 442	у 443	ф 444	х 445	ц 446	ч 447	ш 448	щ 449	ъ 44A	ы 44B	ь 44C	э 44D	ю 44E	я 44F
F.	Ё 401	ё 451	Є 404	є 454	Ї 407	ї 457	Ў 40E	ў 45E	° B0	∙ 2219	· B7	√ 221A	№ 2116	¤ A4	■ 25A0	A0

Другие варианты

(Показаны только последние строки таблиц, поскольку всё остальное совпадает.)

Наиболее распространённый вариант до появления CP866, называемый также «модифицированной альтернативной кодировкой» (в KOI8-R используется тот же набор символов, но в другом порядке):

	.0	.1	.2	.3	.4	.5	.6	.7	.8	.9	.A	.B	.C	.D	.E	.F
F.	Ё 401	ё 451	≥ 2265	≤ 2264	⌠ 2320	⌡ 2321	÷ F7	≈ 2248	° B0	∙ 2219	· B7	√ 221A	n 207F	² B2	■ 25A0	A0

То же самое, но без буквы Ё (все символы 0xF0—0xFF совпадают с соответствующими символами CP437):

	.0	.1	.2	.3	.4	.5	.6	.7	.8	.9	.A	.B	.C	.D	.E	.F
F.	≡ 2261	± B1	≥ 2265	≤ 2264	⌠ 2320	⌡ 2321	÷ F7	≈ 2248	° B0	∙ 2219	· B7	√ 221A	n 207F	² B2	■ 25A0	A0

RUSCII, она же CP1125 (используется на Украине):

	.0	.1	.2	.3	.4	.5	.6	.7	.8	.9	.A	.B	.C	.D	.E	.F
F.	Ё 401	ё 451	Ґ 490	ґ 491	Є 404	є 454	І 406	і 456	Ї 407	ї 457	· B7	√ 221A	№ 2116	¤ A4	■ 25A0	A0

Альтернативная кодировка согласно ГОСТ 19768-87 (по набору символов совпадает с основной кодировкой; в позициях 0xF2—0xF5 должны быть прямые диагональные линии):

	.0	.1	.2	.3	.4	.5	.6	.7	.8	.9	.A	.B	.C	.D	.E	.F
F.	Ё 401	ё 451	□ 256D	□ 256E	□ 256F	□ 2570	→ 2192	← 2190	↓ 2193	↑ 2191	÷ F7	± B1	№ 2116	¤ A4	■ 25A0	A0

CP866.chuv — кодировка, использовавшаяся для отображения знаков чувашского алфавита

	.0	.1	.2	.3	.4	.5	.6	.7	.8	.9	.A	.B	.C	.D	.E	.F
F.	Ё 401	ё 451	Ӑ 4D0	ӑ 4D1	Ӗ 4D6	ӗ 4D7	Ҫ 4AA	ҫ 4AB	Ӳ 4F2	ӳ 4F3	· B7	√ 221A	№ 2116	¤ A4	■ 25A0	A0

Приложение Б - Некоторые библиотечные функции языка С

GETC - GETCHAR

#include <stdio.h>

int getc (stream); читает символ из потока stream.

FILE *stream; указатель на структуру FILE.

int getchar(); читает символ из stdin.

Описание:

Макро getc читает отдельный символ из текущей позиции потока stream и изменяет соответствующий указатель на файл для указания следующего символа. Макро getchar идентична getc(stdin).

Возвращаемое значение:

Макро getc и getchar возвращают прочитанный символ. Возвращаемое значение EOF свидетельствует об ошибке или об условии достижения конца файла. Для определения категории ошибки используются функции ferror или feof.

См. также fgetc, fgetchar, getch, getche, putc, putchar, ungetc.

Замечание! Процедуры getc, getchar идентичны fgetc, fgetchar, но они являются макро, а не функциями.

Пример

```
                #include <stdio.h>

                FILE *stream;
                char buffer[81];
                int i, ch;
                .
                .
                .
                /* следующие операторы позволяют
выбирать требуемую  вводи-
          мую строку из stdin */

                for (i = 0; (i < 80) && ((ch = getchar
()) != EOF) &&
                    (ch != '\n'); i++)
                   buffer [i] = ch;

                buffer[i] = '\0';

                /*  для ввода строк из потока stdin в
операторах, приведен-
          ных выше, можно вместо "getchar ()"
использовать  "getc(stdin)".
      */

                GETCH

                #include <conio.h> требуется только для
объявления функции.

                int getch();

                Описание:
```

Функция getch читает без эхо-отображения отдельный символ прямо с консоли. Напечатанные (введенные) символы не имеют эхо-отображения. Если введен CONTROL-C, система выполняет INT 23H.

Возвращаемое значение:

Функция getch возвращает прочитанный символ. Возвращаемого значения в случае ошибки нет. См.также cgets, getche, getchar.

Пример

```
#include <conio.h>
#include <ctype.h>


int ch;

/* в этом цикле берутся символы с
клавиатуры, пока не встретится "не пробельный" символ.
Предшествующие символы пробела отбрасываются. */
```

PRINTF

```
#include <stdio.h>

int printf (format-string [, argument...]);
char *format-string;  строка управления
форматом.
```

Описание

Функция printf форматирует и печатает наборы символов и значений в выходной стандартный поток stdout. Строка формата состоит из обычных символов, escape-последовательностей и, если за строкой формата следуют аргументы, еще и спецификации формата.

Обычные символы и escape-последовательности просто копируются в stdout в порядке их появления.

Например, строка

printf ("Line one\n\t\tLine two\n");

выработает на выводе

Line one

Line two.

Более подробно escape-последовательности описываются в главе 2.2.4 руководства MSC Compiler Language Reference.

Если за строкой формата следуют аргументы arguments, то эта строка также должна содержать спецификации формата, определяющие формат вывода этих аргументов. Спецификация формата всегда начинается с символа знака процента (%). Ниже о нем описывается подробнее.

Строка формата читается слева направо. Когда встречается первая спецификация формата (если она есть), то значение первого аргумента после строки формата преобразовывается и выводится согласно заданной спецификации. Вторая спецификация формата вызывает преобразование и вывод второго аргумента и так далее, до конца строки формата. Если аргументов больше, чем спецификаций формата, то эти дополнительные аргументы игнорируются. Результат является неопределенным, если аргументов недостаточно для всех спецификаций формата.
Спецификация формата имеет следующую форму:

%x[flags][wigth][.precision][{F:N:h:I}]type.

Каждое поле в спецификации формата является отдельным символом или числом, выражающим отдельную опцию формата. Символ type, появляющийся после последнего необязательного поля формата, определяет аргумент как символ, строку или число. (См. табл. R.1.).

Простейшая спецификация формата содержит только символ знака процента и символ типа (например, %S). Необязательные поля управляют другими аспектами форматирования, как описывается ниже.

ПОЛЕ	ОПИСАНИЕ
flags	Включение вывода и печати знаков, пробелов, десятичных точек, восьмеричных и шестнадцатеричных префиксов. (См. табл. R.2.).
width	Минимальное число выводимых символов.
precision	Максимальное число символов, печатаемых на всем или части поля вывода; или минимальное число цифр для печати целых значений. (См. табл. R.3.).
F, N	Префиксы, позволяющие пользователю override, по умолчанию, адресацию соглашений моделей памяти.
F	Используется для малой модели для печати

	значения, объявленного far.
N	Используется для средней, большой и huge-моделей для near-значений. F и N могут быть использованы только с типами символов s и p, поскольку они уместны только с аргументами, представляющими указатель.
h, l	Предполагаемый размер аргумента: h используется в качестве префикса с целыми типами d, i, o, x, X для определения, что аргумент является short int. l используется в качестве префикса с типами d, i, o, x, X для обозначения, что аргумент является long int. Символ l используется также как префикс с типами e, E, f, g, G для определения, что аргумент является скорее double, чем float.

Если за символом знака процента (%) следует символ, не обозначающий тип формата, то этот символ копируется в поток stdout. Например, для печати символа знака процента используется комбинация %%.

Таблица R.1.

Типы символов функции printf

СИМВОЛ	ТИП АРГУМЕНТА	ФОРМАТ ВЫВОДА
d	целый	целочисленный десятичный знаковый
i	целый	целочисленный десятичный знаковый
u	целый	беззнаковый целочисленный десятичный
o	целый	беззнаковый восьмеричный целый
x	целый	беззнаковый шестнадцатеричный целый, использующий "abcdef"
X	целый	беззнаковый шестнадцатеричный целый, использующий "ABCDEF".
f	с плавающей точкой	знаковое значение, имеющее форму [-]dddd.dddd, где dddd - одна или более десятичных цифр. Количество цифр перед десятичной точкой зависит

		от величины числа, а количество цифр после десятичной точки зависит от требуемой точности.
e	с плавающей точкой	знаковое значение, имеющее форму [-]d.ddde[sign]ddd, где d - десятичная цифра, dddd - одна или более десятичныхя цифр, ddd - ровно три десятичных цифры, и sign - либо "+", либо "-".
E	с плавающей точкой	идентично формату "e", за исключением того, что вместо "e" вводится "E".
g	с плавающей точкой	знаковое значение, распечатываемое в формате "f" или "e", и являющееся более компактным для выбранных значения и точности (как показано ниже). Формат "e" используется, только когда значение экспоненты меньше -4 или больше, чем precision. Ведущие нули отсекаются, и десятичная точка появляется тогда, когда за ней следует одна или несколько цифр.
G	с плавающей точкой	идентично формату "g", за исключением того, что вместо "e" вводится экспонента "E" (если она необходима).
c	символьный	отдельный символ.
s	строковый	символы печатаются до первого нулевого символа '\o' или до достижения precision.
n	указатель на целый	число символов успешно записывается в поток stream; это значение хранится в целом, адрес которого выбирается как аргумент.
p	far-указатель	печать адреса, указываемого аргументом, в форме xxxx:yyyy, где xxxx является сегментом, yyyy является разветвлением, а цифры x и y являются шестнадцатеричными цифрами верхнего регистра (upper-case). %Np печатает только адрес разветвления yyyy. Поскольку %p предполагает указатель на far-значение, аргументы p-указателя могут быть сброшены к far

в маленьких моделях программ.

Таблица R.2.

Символы flags функции printf

FLAG(*)	ЗНАЧЕНИЕ	ПО УМОЛЧАНИЮ
-	Смещение результата влево внутри поля wigth	Смещение вправо
+	Присоединение знака к выводимому значению, если оно имеет знаковый тип	Знак "-" появляется только для отрицательных знаковых значений
blank (' ')	К выводимому значению присоединяется ' ', если выводимое значение является знаковым и положительным; флаг "+" override флаг blank, если оба есть, и положительное знаковое значение выводится вместе со знаком	Без пробела
#	При использовании с форматами o, x, X, флаг # присоединяет к любому ненулевому выводимому значению, соответственно, 0, 0x, 0X	Без префикса
	Когда флаг # используется в формате e, E, f, он определяет наличие десятичной точки в выводимом значении	Десятичная точка появляется только тогда, когда за ней идут цифры
	Когда флаг # используется в формате g, G, он определяет наличие десятичной точки в выводимом значении и препятствует отсечению ведущих нулей	Десятичная точка появляется только тогда, когда за ней идут цифры Ведущие нули отсекаются
	Флаг # игнорируется, при его использовании в форматах c, d, i, u, s	

Примечание. В формате спецификации может содержаться более, чем один flag.

Width - неотрицательное десятичное целое, контролирующее минимальное число напечатанных символов. Если число символов в значении вывода меньше, чем в width, слева и справа добавляются пробелы (в зависимости от того, где

определен флаг "-"), пока минимальная ширина не будет достигнута. Если к width присоединяется 0, то 0 будут добавляться до тех пор, пока не будет достигнут минимум width. (Это не применяется для чисел, смещенных влево).

Спецификация width не требует отсечения значения; если число символов выводимого значения больше чем определено в width, или не задано в нем, все значения символов распечатываются (подлежат спецификации precision).

В спецификации width может быть звездочка (*), когда вместо значения подставляется соответствующий ему аргумент из списка аргументов. Аргумент width должен предшедствовать соответствующему значению.

Спецификация precision является неотрицательным десятичным целым, которому предшедствует точка (.), определяющая количество печатаемых символов или же число десятичных мест.

В отличии от спецификации width, спецификация precision требует отсечения выводимого значения или, в случае значения с плавающей точкой, его округления. В случае подстановки аргумента из списка аргументов в спецификации precision может быть звездочка (*). В списке аргументов аргумент precision предшедствует форматируемому значению. Объяснение значений precision, в зависимости от типа type и случая, когда precision пропущено, представлено в таблице R.3.

Таблица R.3.

Как тип type влияет на значение precision в функции printf

ТИП	ЗНАЧЕНИЕ	ПО УМОЛЧАНИЮ
d i u o x X	Precision определяет минимальное число печатаемых цифр. Если число цифр в аргументе меньше, чем размер precision, слева перед выводимым значением добавляются нули. Если число цифр не превосходит размер precision, значение не отсекается	Если precision равна 0 или пропущена, или если появляется точка (.) без идущих за ней цифр, то precision устанавливается равной 1
e E f	Precision определяет число цифр, печатаемых после десятичной точки. Последняя печатаемая цифра округляется	Precision по умолчанию равна 6; если она равна 0 или перед ней появляется точка (.) без следуемых за ней

		цифр, тогда десятичная точка не печатается
g G	Precision определяет максимальное число важных (многозначных) печатаемых символов	Печатаются все важные (многозначные) цифры
c	Не происходит никакого действия	Печать символа
s	Precision определяет максимальное число печатаемых символов Символы, превышающие размер precision, не печатаются	Печать символов, пока не встретится нулевой символ

Возвращаемое значение

Эта функция возвращает количество напечатанных символов. См. также fprintf, scanf, sprintf, vfprintf, vprintf, vsprintf.

Пример:

```
main ()

/* форматирование и печать различных данных */

{
char ch = 'h', *string = "computer";
int count = 234, *ptr, hex = 0x10, oct = 010, dec = 10;
double fp = 251.7366;

printf("%d  %+d  %06d  %X  %x  %o\n\n",
      count, count, count, count, count, count);

printf("1234567890123%n4567890123456789O\n\n", &count);
printf("Value of count should be 13; count = %d\n\n",
       count);

printf("%10c%5c\n\n", ch, ch);

printf("%25s\n%25.4s\n\n", string, string);

printf("%f  %.2f  %e  %E\n\n", fp, fp, fp, fp);

printf("%i  %i  %i\n\n", hex, oct, dec);

ptr = &count;
printf("%Np  %p  %Fp\n",
       ptr, (int far *) ptr, (int far *)ptr);
}

Тогда на выводе получится следующее:

234  +234  000234   EA  ea  352

12345678901234567890123456789O


Value of count should be 13; count = 13;
    h    h
          computer
                   comp
```

```
251.736600    251.74   2.517366e+002   2.517366E+002

16  8  10

127A  1328:127A   1328:127A.
```

PUTC-PUTCHAR

```
#include <stdio.h>

int putc (c, stream);   записывает символ в поток stream
int c;                  записываемый символ

FILE *stream;           указатель на структуру FILE

int putchar(c);         записывает символ в <stdout>
int c;                  записываемый символ
```

Описание

Процедура putc записывает отдельный символ "c" в текущую позицию выходного потока stream. Процедура putchar идентична процедуре putc(c, stdout).

Возвращаемое значение.

Эти обе процедуры возвращают записанный символ. В случае ошибки возвращается значение EOF. Так как значение EOF может быть воспринято как целая величина, поэтому для проверки места возникновения ошибки применяется функция ferror.
См. также fputc, fputchar, getc, getchar.
Замечание: Процедуры putc и putchar идентичны fputc и fputchar, но они являются макро, а не функциями.

Пример:

```
#include <stdio.h>

FILE *stream;
char buffer[81];
int i, ch;
.
.
.
/* следующий оператор позволяет записать буфер в поток */

for  (i = 0;  (i < 81)  &&  ((ch = putc(buffer[i],
                                stream)) != EOF) ;)
         ++i;

/* Замечание: Поскольку тело утверждения пусто, операция
записи происходит в выражении проверки. */
```

PUTCH

```
#include <conio.h>     требуется только для объявления
                       функции

void putch(c)
int c;                 выводимый символ
```

Описание.

Функция putch записывает символ "c" прямо на консоль.

Возвращаемое значение.

Возвращаемого значения нет.
См.также cprintf, getch, getche.

Пример:

```
#include <conio.h>

 /* в следующем примере показано, как может быть определена функ-
ция getche посредством использования функций putch и getch.*/

int getche()
{
    int ch;

    ch=getch();
    putch(ch);
    return(ch);
}
```

SCANF

```
#include <stdio.h>

int scanf(format-string[[, argument...]]);
char *format-string.       строка управления форматом.
```

Описание.

Функция scanf читает данные из стандартного потока stdin в место, определяемое аргументами arguments. Каждый аргумент должен быть указателем на значение с типом, который соответствует типу, заданному в строке формата. Строка формата управляет преобразаваниями полей ввода. Эта строка может содержать следующее:

"Пробельные" символы, т.е. символ пробела ' ', табуляции \t, новой строки '\n'. Для функции scanf символом пробела определяется считывание, но без запоминания, всех вводимых последующих символов пробела вплоть до первого символа, не являющегося пробелом. При вводе один символ пробела в строке формата соответствует любому числу, включая 0, или любой комбинации символов пробела.

Любой символ управления, не являющийся пробелом и символом знака процента %. Тогда по этому символу для функции scanf определяется считывание, но без запоминания соответствующих символов управления. Если следующий символ в <stdin> не соответствует символам управления, то scanf оканчивает свою работу.

Спецификацию формата, введенную со знаком %. В этом случае scanf читает и преобразовывает введенные символы к значениям заданного типа, причем значения определяются соответствующими аргументами из списка аргументов.

Строка формата читается слева направо. Символы вне спецификации формата предполагаются согласованными с последовательностью символов в потоке stdin; эти согласованные символы в stdin сканируются, но не запоминаются. Если символ в stdin противоречит строке формата, scanf оканчивает свою работу. Этот конфликтующий символ остается в stdin, так как он не может быть прочитан. Когда встречается первая спецификация формата, тогда значение первого поля ввода преобразовывается в соответствии со спецификацией формата и запоминается в месте, заданном первым аргументом. По второй спецификации формата выполняется преобразование второго поля

ввода и запоминание его по второму аргументу; и так до конца строки формата.

Поле ввода ограничивается первым "пробельным" символом или первым символом, который не может преобразоваться по заданному формату, или случаем достижения поля width, которое идет первым.

Если для выбранной спецификации формата задано больше аргументов, чем требуется, то лишние аргументы игнорируются.

Спецификация формата имеет следующую форму.

%<flags><wigth><.precision><{F:N:h:I}><type>.

Каждое поле в формате спецификаций является отдельным символом или числом, выражающим отдельную опцию формата. Символ type, появляющийся после последнего необязательного поля формата, определяет тип поля ввода как символьного, строкового или численного.

Простейший формат спецификации содержит только символ знака процента и символ типа (например, %S).

Каждое поле спецификации формата описывается ниже.

Если за знаком процента % следует символ, не являющийся символом управления форматом, то этот символ и идущие за ним символы, вплоть до следующего знака %, трактуются как обычная последовательность символов, т.е. последовательность, которая должна быть введена. Например, чтобы ввести символ знака %, используется комбинация %%.

Звездочка (*), идущая за знаком %, подавляет назначение следующего поля ввода, задающегося как поле, определяемое типом type. Это поле сканируется, но не запоминается.

Widht является положительным десятичным целым и управляет максимально возможным числом символов, считываемых из stdin. Преобразовываются и запоминаются по соответствующему аргументу только те символы, которые не превышают width. Если в width встречаются "пробельные" символы, т.е. символы пробела, табуляции или новой строки, то по выбранному формату они не преобразовываются, пока не будет достигнут размер width.

Необязательные префиксы F и N не учитывают принятое по умолчанию адресное соглашение используемых моделей памяти. F может быть префиксом к аргументу argument, указывающему на far-объект; а N - на near-объект.

Необязательный префикс l свидетельствует о том, что используется версия long; а префикс h - указывает на использование версии short. Соответствующий argument указывает на long или double-объект (при помощи префикса l) или на short-объект (при помощи префикса h). Модификаторы l и h могут использоваться вместе с типами символов d, i, o, x, u. Модификатор l также может использоваться с символами type e и f. Если определен любой другой type, модификаторы l и h игнорируются.

Символы type и их значения описаны в таблице R.4.

Таблица R.4

Типы символов функции scanf

СИМВОЛ	ПРЕДПОЛАГАЕМЫЙ ТИП ВВОДА	ТИП АРГУМЕНТА
d	десятичный целый	указатель на int.
D	десятичный целый	указатель на long.
o	восьмеричный целый	указатель на int.
O	восьмеричный целый	указатель на long.
x	шестнадцатеричный	указатель на int.

	целый	
X	шестнадцатеричный целый	указатель на long.
i	десятичный, восьмеричный или шестнадцатеричный целый	указатель на int.
I	десятичный, восьмеричный или шестнадцатеричный целый	указатель на long.
u	беззнаковый десятичный целый	указатель на unsigned int.
U	беззнаковый десятичный целый	указатель на unsigned long.
e f	значение с плавающей точкой, содержащее необязательный знак ("+", "-"), одну или больше десятичную цифру, обычно содержащую десятичную точку и экспоненту ("e", "E"), которая записывается за знаковым целым значением.	указатель на float
c	символьный. Символы пробела, табуляции или новой строки, так называемые "пробельные символы", которые обычно пропускаются, при задании этого типа считываются. Для считывания следующего символа, не являющегося "пробельным", используется комбинация %1s.	указатель на char
s	строковый.	указатель на символьный массив, достаточно большой для вводимого поля вместе с нулевым символом окончания '\0', появляющимся автоматически.
n	чтение при вводе из stream или буфера не происходит.	указатель на int, в котором записывается число успешно считанных символов из потока или буфера, вплоть до указанных в вызове scanf.

p	значение в форме xxxx : yyyy, где цифры x и y являются шестнадцатеричными цифрами верхнего регистра.	указатель на far-группу данных.

При чтении строк, не ограниченных символами пробела, множество символов в квадратных скобках [] должно заменяться строковым типом s. Соответствующее поле ввода читается вплоть до первого символа, не содержащегося в ограниченном квадратными скобками множестве символов. Если в этом множестве первым символом является caret (^), результат сохраняется: поле ввода считывается до первого символа, не входящего в это множество символов. Чтобы запомнить строку без нулевого символа '\0', применяется спецификация %nc, где n - десятичное целое. В этом случае символьный тип s определяет аргумент, который указывает на массив символов. Следующие n символов считываются из входного потока в определенное местоположение и нулевой символ не записывается.

Функция scanf для каждого поля ввода сканирует символ за символом. Она может окончить чтение отдельного поля при достижении символа пробела, если либо достигнуто поле width; либо следующий вводимый символ не может быть преобразован по заданному формату; либо следующий символ конфликтует с соответствующим ему символом в управляющей строке формата; либо же следующий символ отсутствует в выбранном множестве символов. Когда происходит вынужденный процесс окончания считывания, то следующее поле ввода рассматривается с самого первого конфликтующего символа. Этот символ, если он один, рассматривается как непрочитанный, либо как первый символ следующего поля ввода, либо как первый символ в последующих операциях чтения потока stdin.

Возвращаемое значение.

Эта функция возвращает число успешно преобразованных и назначенных полей. В возвращаемом значении не содержится число прочитанных но не назначенных полей. При попытке считывания конца файла возвращается значение EOF. Возвращаемое значение 0 указывает, что нет назначенных полей.

См. также fscanf, printf, sscanf, vfprintf, vprintf, vsprintf.

Пример 1.

```
#include <stdio.h>

int i;
float fp;
char c, s[81];

scanf("%d  %f  %c  %s", &i, &fp, &c, s);
/* ввод различных данных */.
```

Пример 2.

```
#include <stdio.h>

main ()          /* преобразование шестнадцатеричного
                 ** или восьмеричного  целого     к
                 ** десятичному целому  */
{
int numassigned, val;

pintf("Enter hexadecimal or octal #, or 00
       to guit:\n");
```

```
   do
      {  printf("# = ");
         numassigned = scanf("%i", &val);
         printf("Decimal # = %i\n", nal);
      }
   while (val && numassigned);

      /*  конец  цикла,  если  значение  ввода равно 00, или если
scanf не способна назначить поле */.

   }

      Тогда на выходе будет следующее.

   Enter hexadecimal or octal #, or 00 to guit:

   # = 0xf
   Decimal # = 15

   # = 0100

   Decimal # = 64

   # = 00

   Decimal # = 0.
```

Приложение B - Драйвер ANSI.SYS

Этот драйвер позволяет вам использовать ANSI-последовательности в реальном режиме. ANSI-последовательность - это последовательность символов, обычно начинающаяся с символа, имеющего код 27 (ESC-символ), разработанная Американским Национальным Институтом Стандартов (ANSI).

Для инсталяции драйвера ansi.sys необходимо включить в файл config.sys командную строку следующего формата:

device=[дисковод:][маршрут]ansi.sys[/x]

Ключ /x предназначен для независимого использования одноименных клавиш. Например, некоторые клавиатуры (101-клавишные) имеют две клавиши home. Обе эти клавиши считаются идентичными, пока в командной строке не будет указан ключ /x.

Далее приводится список ANSI-последовательностей для MS-DOS. При описании последовательностей используются следующие переменные:

Pn числовой параметр - десятичное число, указываемое вами в ASCII цифрах.
Ps параметр выбора - десятичное число, используемое для выбора подфункции. Вы можете указать более одной подфункции путем разделения параметров символом (;) (точка с запятой).
PL параметр строки - десятичное число, указываемое вами в ASCII цифрах.
Pc параметр колонки - десятичное число, указываемое вами в ASCII цифрах.

ANSI-последовательности

Последовательность	Функция
ESC[PL;PcH	Позиция курсора (CUP).
ESC[PL;PcF	Горизонтальная и вертикальная позиция (HVP). CUP и HVP перемещают курсор на позицию, указываемую параметрами. Если параметры не указаны, то курсор помещается в левый верхний угол экрана.
ESC[PnA	Курсор вверх (CUU). Курсор сдвигается на Pn строк. Если курсор уже находится на верхней строке, то эта последовательность игнорируется системой.
ESC[PnB	Курсор вниз (CUD). Курсор сдвигается на Pn строк. Если курсор уже находится на нижней строке, то эта последовательность игнорируется

	системой.
ESC[PnC	Курсор вперед (CUF). Курсор сдвигается на Pn колонок. Если курсор уже находится в самой правой колонке, то эта последовательность игнорируется системой.
ESC[PnD	Курсор назад (CUD). Курсор сдвигается на Pn колонок. Если курсор уже находится в самой левой колонке, то эта последовательность игнорируется системой.
ESC[6n	Доклад о состоянии устройства (DSR). Драйвер консоли при получении DSR выводит RCP последовательность.
ESC[s	Сохранить позицию курсора (SCP). Эта позиция может быть восстановлена с помощью RCP последовательности.
ESC[u	Восстановить позицию курсора (RCP).
ESC[2J	Очистить экран (ED). Курсор после очистки экрана помещается в левый верхний угол.
ESC[K	Удалить строку (EL). Удаление строки от позиции курсора (включая саму позицию) до конца строки.
ESC[Ps;...;Psm	Функции экрана (SGR). Вызванные функции действуют до появления следующей SGR последовательности.

Функции графики

0 - Отмена всех атрибутов
1 - Bold
2 - Underscore
3 - Blink
4 - Reverse video
5 - Concealed

Выбор цвета:
30 - Черный
31 - Красный
32 - Зеленый
33 - Желтый
34 - Синий
35 - Магента
36 - Циан
37 - Белый

Цвет фона:
40 - Черный
41 - Красный
42 - Зеленый
43 - Желтый
44 - Синий
45 - Магента

46 - Циан
47 - Белый
Параметры 30-47 соответствуют стандарту ISO 6429.

ESC[=Psh
ESC[=h
ESC[=0h
ESC[?7h

Установить режим (SM).
SM-последовательность выбирает ширину или тип экрана, в зависимости от перечисленных ниже значений параметра:

0 - 40 * 25 ЧБ
1 - 40 * 25 ЦВ
2 - 80 * 25 ЧБ
3 - 80 * 25 ЦВ
4 - 320 * 200 ЦВ
5 - 320 * 200 ЧБ
6 - 640 * 200 ЧБ
7 - Перенос в конце каждой строки
14 - 640 * 200 ЦВ
15 - 640 * 350 МОНО
16 - 640 * 350 ЦВ
17 - 640 * 480 ЦВ
18 - 640 * 480 ЦВ
19 - 320 * 200 ЦВ

ESC[=Psl
ESC[=l
ESC[=0l
ESC[?7l

Сброс режима (RM).
Параметры для RM такие же, что и для SM, за исключением параметра 7, сбрасывающего режим переноса в конце каждой строки.

ESC[код;строка;...p

Эта последовательность позволяет переопределить клавиши клавиатуры указанным строкам, где:

строка - это либо ASCII код одного символа, либо строка, заключенная в кавычки. Например, как 65, так и "A" могут быть использованы для представления заглавной буквы A.

код - величина, представляющая соответствующую клавишу. Таблица соответствия приведена ниже. Некоторые величины представляют собой пару значений, разделенных точкой с запятой. Этот символ (;) также должен быть включен в управляющую (ESC) строку при указании кода.

Клавиша	Код			
	Одна	+SHIFT	+Ctrl	+Alt
F1	0;59	0;84	0;94	0;104
F2	0;60	0;85	0;95	0;105

F3	0;61	0;86	0;96	0;106
F4	0;62	0;87	0;97	0;107
F5	0;63	0;88	0;98	0;108
F6	0;64	0;89	0;99	0;109
F7	0;65	0;90	0;100	0;110
F8	0;66	0;91	0;101	0;111
F9	0;67	0;92	0;102	0;112
F10	0;68	0;93	0;103	0;113
F11	0;133	0;135	0;137	0;139
F12	0;134	0;136	0;138	0;140
Home	0;71	55	0;119	...
Стрелка вверх	0;72	56	...	...
Страница вверх (PgUp)	0;73	57	0;132	...
Стрелка влево	0;75	52	0;115	...
Стрелка вниз	0;77	54	0;116	...
End	0;79	49	0;117	...
Страница вниз (PgDn)	0;81	51	0;118	...
Ins	0;82	48	...	...
Del	0;83	46	...	...
PrtSc	...	...	0;114	...
A	97	65	1	0;30
B	98	66	2	0;48
C	99	67	3	0;46
D	100	68	4	0;32
E	101	69	5	0;18
F	102	70	6	0;33
G	103	71	7	0;34
H	104	72	8	0;35
I	105	73	9	0;23
J	106	74	10	0;36
K	107	75	11	0;37
L	108	76	12	0;38
M	109	77	13	0;50
N	110	78	14	0;49
O	111	79	15	0;24
P	112	80	16	0;25
Q	113	81	17	0;16
R	114	82	18	0;19
S	115	83	19	0;31
T	116	84	20	0;20
U	117	85	21	0;22
V	118	86	22	0;47
W	119	87	23	0;17
X	120	88	24	0;45
Y	121	89	25	0;21
Z	122	90	26	0;44
1	49	33	...	0;120
2	50	64	...	0;121
3	51	35	...	0;122
4	52	36	...	0;123
5	53	37	...	0;124
6	54	94	...	0;125
7	55	38	...	0;126
8	56	42	...	0;127
9	57	40	...	0;128
0	48	41	...	0;129
-	45	95	...	0;130
=	61	43	...	0;131
Tab	9	0;15	...	...
Null	0;3	...	...	...

Пример:

Поменять клавиши обратной косой и знака вопроса, используя заковыченные строки, можно следующей последовательностью:

```
ESC["\";"?"pESC["?";"\"p
```