Задание практикума №2.

<Цель задания – изучение методов сортировки, модульное программирование, проектирование и реализация экранного пользовательского интерфейса >

Содержание задания.

Требуется написать программу, которая упорядочивает пользовательские данные. Входные данные для упорядочения (длина последовательности и сами элементы) могут находиться в заранее подготовленном файле либо вводиться с клавиатуры. В процессе сортировки следует вычислять количество сравнений и перемещений элементов последовательности. Результаты работы программы (упорядоченная последовательность и количество операций) выводятся в файл либо на экран. Метод сортировки, тип элементов последовательности и пользовательский интерфейс определяются вариантом задания. Программа должна быть устойчива к ошибкам: ввод некорректных данных не должен приводить к аварийной остановке программы.

Процедуру сортировки следует отладить на массиве целых чисел. Для этого надо написать программу, которая вводит с клавиатуры массив, вызывает процедуру сортировки и печатает на экране упорядоченный массив, количества сравнений и перемещений элементов. Во время отладки процедура сортировки должна печатать массивы, получающиеся на каждом шаге сортировки. Содержание терминов *перемещение элементов массива* и *шаг сортировки* зависит от метода сортировки.

Выполняя задание, нужно использовать средства модульного программирования языка Турбо Паскаль.

Первый этап выполнения задания состоит в детальном проектировании пользовательского интерфейса и разработке модульной структуры программы.

Задание сдаётся по частям – программа отладки сортировки и программа упорядочивания пользовательских данных ("интерфейс") сдаются отдельно. Отчёт по заданию включает в себя:

1. Описание модульной структуры системы. Перечислить, из каких модулей состоит программа; для каждого модуля указать основные константы, типы, переменные и процедуры; для основных процедур описать параметры и суть процедуры; описать связи между модулями.
2. Отчёт по методу сортировки. Описание метода; минимально возможное количество операций и массив, на котором оно достигается; максимально возможное количество операций и соответствующий ему массив.
3. Отчёт по "интерфейсу". Вид экрана на каждом этапе работы программы; управляющие клавиши; примерный сценарий работы пользователя с программой.

Замечание. Операторы перехода любого вида, включая halt, exit, break, не использовать.

Варианты задания.

1. Методы сортировки.

метод сортировки	перемещение *move(d1,d2)*	шаг сортировки
1. простые вставки	$d1 := d2$	установка на место очередного элемента
2. бинарные вставки	$d1 := d2$	установка на место очередного элемента
3. пузырек	$d1 \leftrightarrow d2$	один просмотр массива с перестановками
4. челночная	$d1 \leftrightarrow d2$	перестановка на место элемента, нарушающего порядок
5. простой выбор	$d1 \leftrightarrow d2$	установка на место очередного элемента
6. Шелла	в зависимости от метода сортировки подмассива	упорядочен подмассив x_i , x_{i+k}, x_{i+2k},…
7. быстрая	$d1 \leftrightarrow d2$	установка на место очередного элемента
8. простое слияние	$d1 := d2$	описан в методе
9. естественное слияние	$d1 := d2$	описан в методе

2. Варианты интерфейсной части.

1. Входные данные для сортировки содержатся в файле, упорядоченная последовательность записывается в файл; количество операций выводится на экран. Имена файлов задаются пользователем, редактирование имени файла при вводе не предусматривается.
2. Входные данные для сортировки содержатся в файле, упорядоченная последовательность записывается в файл; количество операций выводится на экран. Имена файлов задаются пользователем, при вводе возможно редактирование имени файла.
3. Входные данные вводятся пользователем с клавиатуры. Пользователь не имеет возможности исправлять введённые элементы. Упорядоченная последовательность записывается в файл; количество операций выводится на экран. Пользователь может просмотреть упорядоченную последовательность.
4. Входные данные вводятся с клавиатуры. Пользователь может исправлять элемент последовательности при его вводе. Результаты выводятся на экран.
5. Входные данные вводятся с клавиатуры. Предусмотрено редактирование элемента последовательности при его вводе. Упорядоченная последовательность записывается в файл; количество операций выводится на экран. Пользователь может просмотреть упорядоченную последовательность.
6. Пользователь указывает список файлов. Требуется упорядочить элементы всех файлов, объединив их в одну последовательность. Упорядоченная последовательность записывается в файл; количество операций выводится на экран. Предусмотрено редактирование имён файлов при вводе.
7. Входные данные вводятся с клавиатуры или берутся из файла. Упорядоченная последовательность выводится на экран или записывается в файл, количество операций выводится на экран. Имена файлов вводятся пользователем. Редактирование в процессе ввода не предусмотрено.
8. Входные данные вводятся пользователем с клавиатуры. Пользователь может исправлять введённые ранее элементы последовательности. Результаты выводятся на экран.
9. Входные данные вводятся с клавиатуры или берутся из файла. Упорядоченная последовательность записывается в файл, количество операций выводится на экран. Имена файлов вводятся пользователем. Имеется возможность редактирования имени файла и элемента последовательности при вводе.

3. Типы элементов массива

1. Длинные целые числа, превосходящие размер стандартных типов Турбо Паскаля.
2. Рациональные числа вида m/n, $m \in \boldsymbol{Z}$, $n \in \boldsymbol{N}$.
3. Числа, записанные по правилам ассемблера (как последовательность цифр и спецификатор основания системы счисления).
4. Английские слова из строчных и прописных букв.
5. Русские слова из строчных и прописных букв.
6. Английские имена и фамилии.
7. Русские имена и фамилии.
8. Римские числа.
9. Температура, может быть задана по Цельсию или по Фаренгейту -12C, 45F, -0.5C
10. Даты в формате dd.mm.yyyy.
11. Адреса: улица/дом/квартира. При вводе следует учесть различные варианты записи адреса и виды адресов: ул. Масловка, д. 5, кв. 3 – улица Масловка, дом 5, кв.3; Сиреневый бульвар, д.11 - Сиреневый б-р, д.11.

Методические указания.

<<< В процессе разработки. >>>
<<< Смотрите записи семинарского занятия. Задавайте вопросы. >>>

<<<

<параллельная работа над частями задания>
<отладка сортировки – специальный main>
<отладка интерфейса – заглушка вместо сортировки>
⇒
<модули>

Перед написанием программы требуется доопределить интерфейс – описать сценарий взаимодействия пользователя с программой, продумать расположение окон на экране, основные цвета, действующие клавиши, подсказки. В вариантах 1 – 3 надо также определить формат файлов. Описание интерфейса является частью задания и сдаётся как часть отчёта.

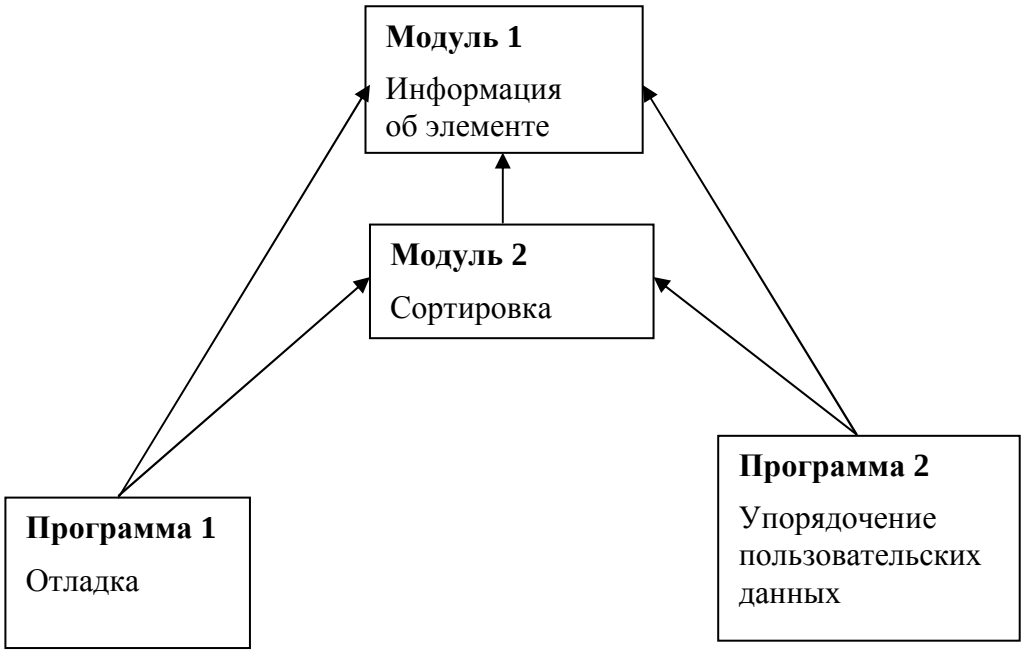
Для настройки процедуры сортировки на обработку данных разных типов (целые числа и тип, определяемый вариантом задания), тип элемента и операции с элементами последовательности надо также описать в отдельном модуле.

Метод сортировки, интерфейс взаимодействия пользователя с программой и типы элементов последовательностей определяются вариантом задания.

Таким образом, требуется написать

1. два модуля, содержащих описание типов элементов последовательности и описание процедур обработки элемента последовательности;
2. модуль с процедурой сортировки;
3. модуль с интерфейсными процедурами;
4. программу для отладки процедуры сортировки;
5. программу, упорядочивающую пользовательские данные.

Замечание. Процедура сортировки должна уметь печатать или не печатать состояние массива после каждого шага сортировки в зависимости от того, какая программа её вызвала. Можно реализовать это, введя глобальную переменную (печатать/не печатать), значение которой устанавливается вызывающей программой и проверяется процедурой сортировки.


Модуль 1
Информация
об элементе
Модуль 2
Сортировка
Программа 1
Отладка
Программа 2
Упорядочение
пользовательских
данных

>>>

Краткое описание методов сортировки.

Упорядочивается по возрастанию массив $X_1, \ldots, X_n$.

Сортировки вставками.

Общая идея методов сортировки вставками: в ранее упорядоченную подпоследовательность $X_1, X_2, \ldots, X_{k-1}$ вставляется элемент X_k так, чтобы упорядоченными оказались k первых элементов исходной последовательности. Для этого

1. в подмассиве $X_1, X_2, \ldots, X_{k-1}$ отыскивается место для элемента X_k, т.е. находится номер i, такой что $X_j < X_k$ для $j \in [1, i]$ и $X_k < X_j$ для $j \in [i + 1, n]$;
2. элементы $X_{i+1}, \ldots, X_{k-1}$ сдвигаются на одну позицию вправо и X_k записывается на место элемента X_{i+1}.

Процесс заканчивается, когда поставлен на свое место элемент X_n.

В зависимости от способа поиска места для элемента X_k различаются следующие методы.

а) ***Метод простых вставок.*** Величина X_k по очереди сравнивается с $X_1, X_2, \ldots, X_{k-1}$ до тех пор, пока не будет найдено место для элемента X_k .

б) ***Метод бинарных вставок.*** Величина X_k сравнивается со средним элементом подпоследовательности $X_1, X_2, \ldots, X_{k-1}$. Если X_k меньше этого элемента, то место для него ищется тем же способом в левой половине подпоследовательности, а если больше – в правой половине.

Метод пузырька.

По очереди просматриваются пары соседних элементов X_1 и X_2, X_2 и X_3, X_3 и X_4 и т.д.; в неупорядоченных парах ($X_i > X_{i+1}$) переставляются элементы. В результате максимальный элемент окажется на своем окончательном месте – в конце последовательности. Далее аналогичная процедура применяется ко всем элементам, кроме последнего. Если при очередном просмотре последовательности не было произведено ни одной перестановки, то последовательность уже упорядочена и следует прекратить сортировку.

Челночная сортировка.

Сравниваются последовательные пары X_1 и X_2, X_2 и X_3, X_3 и X_4 и т.д. до обнаружения неупорядоченной пары: $X_k > X_{k+1}$. "Неупорядоченный" элемент сравнивается и меняется местами с предыдущими элементами, пока не окажется на своём месте. Таким образом получается упорядоченный подмассив $X_1, X_2, \ldots, X_{k+1}$. Затем с $(k+1)$-ой позиции возобновляется просмотр массива X и поиск неупорядоченной пары. Если неупорядоченная пара не найдётся, процесс заканчивается.

Сортировка посредством простого выбора.

В массиве $X_1, \ldots, X_n$ отыскивается минимальный элемент. Найденный элемент меняется местами с первым элементом массива. Затем так же обрабатывается подмассив $X_2, \ldots, X_n$. Когда обработан подмассив X_{n-1}, X_n, сортировка оканчивается.

Метод Шелла.

Пусть k - целое от 1 до $n/2$. Независимо друг от друга упорядочиваются (одним из описанных выше методов) подпоследовательности из элементов, отстоящих друг от друга

на k позиций: X_i, X_{i+k}, X_{i+2k}, ... (i = 1, 2, ..., k).

Затем k уменьшается и процесс повторяется заново. Последний шаг должен выполняться при k = 1.

Значение k можно менять по разным законам. В отчете следует объяснить выбранный способ изменения k.

Быстрая сортировка (метод Хоара).

Суть метода заключается в следующем. Выбирается произвольный элемент q массива X_1,..., X_n. Элементы массива переставляются таким образом, что бы в начале массива оказались элементы, меньшие или равные q, а в конце – большие или равные q. Таким образом массив делится на две части.

Требуемую перестановку элементов можно выполнить следующим образом. Последовательность просматривается слева направо, пока не встретится элемент, больший или равный q; затем массив просматривается справа налево до элемента, меньшего или равного q. Найденные элементы меняются местами, после чего просмотры с обоих концов последовательности возобновляются со следующих элементов.

После разбиения массива на две части описанный выше процесс рекурсивно применяется к полученным подмассивам.

Когда длина обрабатываемого подмассива станет меньше трех, его надо упорядочить более простым методом.

Сортировка слиянием.

Основная идея такой сортировки - разделить последовательность на уже упорядоченные подпоследовательности (назовем их "отрезками") и затем объединять эти отрезки во все более длинные упорядоченные отрезки, пока не получится единая упорядоченная последовательность. Отметим, что при этом необходима дополнительная память (массив Y[1..n]).

Различаются следующие варианты сортировки слиянием.

а) ***Простое слияние.*** Считается, что вначале отрезки состоят только из одного элемента, они сливаются в отрезки из двух элементов (из X_1 и X_2, из X_3 и X_4, ...), которые переносятся в массив Y. На втором этапе соседние двухэлементные отрезки (Y_1, Y_2 и Y_3, Y_4; Y_5, Y_6 и Y_7, Y_8; ...) объединяются в отрезки из 4 элементов, которые записываются в массив X. На третьем этапе строятся отрезки из 8 элементов, которые заносятся в массив Y, и т.д.

б) ***Естественное слияние.*** Берутся наиболее длинный (по неубыванию) отрезок в начале массива X и наиболее длинный (также по неубыванию, но при просмотре справа налево) отрезок в конце массива, и они сливаются в один отрезок, который записывается в *начало* массива Y. Затем сливаются следующие максимально длинные отрезки с обоих концов, и полученный отрезок записывается (справа налево) в *конец* массива Y. Третьи по порядку отрезки после слияния записываются снова в *начало* Y (вслед за первым объединенным отрезком), четвертые - в *конец* Y (перед вторым объединенным отрезком) и т.д. Первый этап сортировки оканчивается, когда все элементы из X будут перенесены в массив Y. На втором этапе применяется та же процедура, только массивы X и Y меняются ролями. И так далее.

Замечание: в обоих вариантах следует учитывать, что в конце концов упорядоченные элементы должны оказаться в массиве X.

Модули в Турбо Паскале.

<<< добавить про инициализирующую часть – она нужна для того, чтобы присвоить значение переменной dbg , хранящейся в модуле SortUnit. Идея: "использование недокументированный особенностей"☺ Main отладки присваивает переменной DBG

значение true (для установки режима работы процедуры sort) и вызывает процедуру sort. Интерфесная часть не знает вообще ничего о разных режимах работы sort. Начальное значение DBG= false присваивается в иниц. части модуля SortUnit. Пример – см. в десктоп → BP → My Test Programs → Unit Initialize Part>>>

Модуль – логически замкнутая часть программы, которую можно компилировать отдельно от остальных частей. Модуль может содержать описания констант, типов, переменных, процедур. Информацию, описанную в одном модуле, может использовать основная программа и другие модули.

Модуль описывается следующим образом:

```
unit {имя модуля};
interface
   { интерфейсная часть }
implementation
   { реализация }
end.
```

В интерфейсной части описываются имена, видимые вне модуля. Для процедур и функций приводятся только заголовки. В части реализации даются полные описания для всех процедур и функций, перечисленных в интерфейсной части, и описываются локальные для модуля имена (константы, типы, переменные, процедуры), недоступные другим частям программы, но необходимые для работы модуля.

Модуль сохраняется в файл с расширением PAS. Перед использованием модуль следует откомпилировать с помощью команды compile. Откомпилированный модуль записывается в файл с расширением TPU (Turbo Pascal Unit).

Для использования модуля следует в начале текста программы сразу за ее заголовком поместить директиву uses:

```
program ...;
uses { имя модуля1, имя модуля 2, … };
begin
. . .
end.
```

Один модуль может использовать информацию из другого модуля. Если модулю нужны определения другого модуля для своей интерфейсной части (например, там определен тип параметра процедуры), можно поместить директиву uses в интерфейсную часть:

```
unit Unit1;
interface
uses Unit2;
   { используются имена из Unit2 }
. . .
implementation
    { реализация }
end.
```

В противном случае ее следует указать в части реализации:

```
unit Unit1;
interface
   { интерфейс }
implementation
uses Unit2;
   { используются имена из Unit2 }
   { реализация }
end.
```

При внесении изменений в модуль его следует откомпилировать ещё раз, так же как и модули, использующие информацию, описанную в изменённом модуле.

/// Далее – фрагмент методички по практикуму ///

<<< работа с внешними файлами в Турбо Паскале >>>
<<< работа со стороками в Турбо Паскале >>>

Модули CRT и CRT1

Для реализации подсистемы интерфейса следует использовать возможности, которые предоставляют стандартный модуль CRT системы программирования Турбо Паскаль 7.0 и модуль CRT1, специально созданный для данного задания (ниже процедуры и функции из модуля CRT1 помечены звездочкой). Для того чтобы можно было воспользоваться процедурами, константами и т.п. из этих модулей, следует в начале текста программы сразу за ее заголовком поместить строку

```
uses crt, crt1;
```

1 Работа с окнами и цветом

Окном называется прямоугольный участок экрана. Установка окна (с помощью процедуры *window*) не вызывает никаких изменений на экране, но означает, что все последующие действия с экраном ведутся только в рамках окна, как будто бы и нет иной части экрана. В частности, все координаты позиций отсчитываются от левого верхнего угла окна; исключение составляет обращение к процедуре *window*, координаты для которой всегда задаются как абсолютные, т.е. отсчитываются от верхнего левого угла экрана. В начале работы программы текущим окном является весь экран.

Координаты позиций на экране (в текущем окне) задаются парой *(x,y)*, где *x* означает номер колонки экрана (окна), а *y* - номер строки. Отсчет координат ведется от левого верхнего угла экрана (окна), который имеет координаты (1,1). Правый нижний угол всего экрана имеет координаты (80,25).

При высвечивании любого символа на экране используются два цвета: фоновый цвет, которым закрашивается тот участок экрана, где показывается символ, и передний цвет, которым высвечивается сам символ. Все допустимые цвета (их всего 16) нумеруются целыми числами от 0 до 15. При этом передний цвет может быть любым, а фоновый цвет должен иметь номер от 0 до 7. Для более наглядного обозначения цветов в модуле CRT описаны следующие константы:

```
black=0;      {черный}           darkgray=8;      {темно-серый}
blue=1;       {синий}            lightblue=9;     {светло-синий}
green=2;      {зеленый}          lightgreen=10;   {салатовый}
cyan=3;       {сине-зеленый}     lightcyan=11;    {голубой}
red=4;        {красный}          lightred=12;     {светло-красный}
magenta=5;    {малиновый}        lightmagenta=13; {светло-малиновый}
brown=6;      {коричневый}       yellow=14;       {желтый}
lightgray=7;  {серый}            white=15;        {белый}
```

Рекомендуется использовать эти названия вместо числовых кодов цветов; например, вместо *textcolor(14)* лучше писать *textcolor(yellow)*.

Для работы с окнами и цветом в модулях CRT и CRT1 имеются следующие процедуры (тип *byte* обозначает целые числа от 0 до 255):

textmode(3) - процедура при фактическом параметре 3 устанавливает цветной текстовый режим работы с экраном из 80 колонок и 25 строк; с обращения к этой процедуре надо начинать работу подсистемы интерфейса

window(x1,y1,x2,y2:byte) - процедура объявляет прямоугольную часть экрана с левым верхним углом в точке *(x1,y1)* и правым нижним углом в точке *(x2,y2)* текущим окном; все координаты здесь - абсолютные, т.е. отсчитываются от левого верхнего угла всего экрана

* *windcoord(var x1,y1,x2,y2:byte)* - процедура присваивает своим параметрам (абсолют-ные) координаты текущего окна

textcolor(fc:byte) - процедура делает цвет с номером *fc* (от 0 до 15) цветом символов (передним цветом) для всех последующих выдач на экран, осуществляемых стандартной процедурой вывода *write*

textbackground(bc:byte) - процедура делает цвет с номером *bc* (от 0 до 7) фоновым цветом для всех последующих выдач на экран, осуществляемых процедурой вывода *write*

* *getcolors(var fc,bc:byte)* - процедура присваивает параметрам *fc* и *bc* номера текущих переднего и фонового цветов, соответственно

clrscr - процедура очищает текущее окно (записывает во все его позиции пробелы), закрашивая его текущим фоновым цветом

2 Работа с курсором

Курсор всегда показывает позицию на экране, в которую будет помещен очередной символ, вводимый с клавиатуры стандартной процедурой ввода *read* или выводимый стандартной процедурой вывода *write*.

Для работы с курсором полезны следующие процедуры из модулей CRT и CRT1:

gotoxy(x,y:byte) - процедура перемещает курсор в позицию *(x,y)* текущего окна

* *wherexy(var x,y:byte)* - процедура присваивает своим параметрам координаты текущей позиции курсора
* *crsoff* - процедура делает курсор невидимым на экране
* *crson* - процедура восстанавливает видимость курсора

3 Непосредственный доступ к экрану

Вывод на экран можно осуществлять без обращения к стандартной процедуре вывода *write*. В этом случае используется тот факт, что информация, отображаемая на экране, хранится в специальном месте оперативной памяти, называемом видеопамятью. Каждую секунду видеопамять многократно просматривается и ее содержимое отображается на экране, поэтому любая запись в нее означает вывод на экран. Чтение же из ячеек видеопамяти позволяет узнать текущее содержимое экрана. (*Замечание*: в этих операциях положение курсора не учитывается и не меняется.)

Каждый элемент видеопамяти состоит из двух байтов: один - это кода символа, который сейчас высвечивается в соответствующей позиции экрана, а другой - так называемый цветовой атрибут, указывающий фоновый и передний цвета, которые используются при высвечивании символа в этой позиции экрана, и признак "мерцания" символа. Более точно, цветовой атрибут - это величина *blink* + 16**bc*+ *fc*, где *fc* - номер переднего цвета, *bc* – номер фонового цвета, а *blink* равен 128, если символ должен мерцать на экране, и равен 0 в противном случае. Например, атрибут "белый символ на синем фоне, без мерцания" задается так: 16**blue* + *white*.

Для работы с видеопамятью в модуле CRT1 имеются следующие процедуры и функции:

* *putch(x,y:byte; c:char)* - процедура записывает символ *c* в позицию *(x,y)* текущего окна, не меняя цветовой атрибут в этой позиции
* *putattr(x,y:byte; a:byte)* - процедура меняет цветовой атрибут позиции *(x,y)* текущего окна на новое значение *a*, не меняя сам символ в этой позиции
* *getch(x,y:byte):char* - значением функции является символ, высвечиваемый в данный момент в позиции *(x,y)* текущего окна
* *getattr(x,y:byte):byte* - значением функции является цветовой атрибут позиции *(x,y)* текущего окна

4 Ввод с клавиатуры "без эха"

Использовать в задании для ввода длин и элементов (дат) сортируемых последовательностей стандартную паскалевскую процедуру *read* нельзя: она не учитывает границы поля ввода, трактует даты как неправильные числа и т.д. Поэтому ввод длины и дат

(с возможностью редактирования вводимой информации) необходимо реализовать иначе, используя так называемый ввод "без эха".

Когда на клавиатуре нажимается клавиша (вводится символ), то данный символ автоматически не высвечивается на экране - это дополнительное действие, которое может быть выполнено, а может быть и не выполнено. В связи с этим различаются два вида ввода - "с эхом", когда введенный символ тут же высвечивается на экране (именно так работает процедура *read*, которая сама и высвечивает символ), и "без эха", когда символ не высвечивается. Во втором случае вопрос о том, высвечивать символ или нет, а если высвечивать, то в какой позиции экрана, должен решаться дополнительно. Например, если нажата управляющая клавиша (типа Enter или ←), то на экране обычно ничего не высвечивается, а если нажата клавиша с обычным символом, то этот символ, как правило, надо выводить на экран. Такое отделение операции ввода от операции вывода на экран позволяет запрограммировать свой редактор ввода, который "в темную" вводит символ за символом и определяет, что делать с каждым из них.

Для ввода "без эха" можно использовать следующую процедуру из модуля CRT1:

* *inkey(var c:char; var spec:boolean)* - процедура вводит "без эха" символ первой из клавиш, нажатых на клавиатуре, и присваивает его параметру *c* (если никакая клавиша еще не была нажата, процедура ждет нажатия первой же клавиши); при нажатии клавиши с обычным символом параметру *spec* присваивается значение *false*, при нажатии управляющей клавиши – значение *true*

Замечание. Количество символов, которые можно ввести с клавиатуры (с учетом управляющих клавиш и комбинаций типа Ctrl+Shift), столь велико, что для них не хватает имеющихся 256 кодов. Поэтому все эти символы разделены на две группы, в одну из которых включены обычные символы (буквы, цифры, знаки операций и т.п.), а в другую - управляющие символы (и их комбинации). При этом в каждой группе символы имеют одни и те же коды - от 0 до 255 (например, код 83 имеет и буква S, и клавиша Del), поэтому знание только кода еще не определяет символ. Именно из-за этого процедура *inkey* и сообщает через свой параметр *spec*, символ из какой группы был введен.

Ниже приведены (десятичные) коды некоторых управляющих символов, выдаваемые процедурой *inkey*:

```
↑ -     72        ↓ -   80        ← -         75        → -    7
Enter - 13        Del – 83        Backspace -  8        Ins -  82
Esc -   27        Tab -  9        Home –      71        End –  79
F1 -    59        F2 -  60        . . .                 F10 -  68
```

Рекомендуется ввести в программе константы с подходящими именами, например:

```
const Enter=#13; Left=#75; EndKey=#79; F3=#61; ...
```

и уже ими пользоваться.

Замечание:

Для рисования рамок вокруг окон можно с помощью процедуры *putchar* записывать в граничные позиции окон символы так называемой псевдографики. Они имеют следующие (десятичные) коды:

```
218: ┌  196: ─  191: ┐          201: ╔  205: ═  187: ╗

179: │  197: ┼  179: │          186: ║  206: ╬  186: ║

192: └  196: ─  217: ┘          200: ╚  205: ═  188: ╝
```

Примечание.

Для студентов, использующих Free Pascal.

(По непроверенным данным ☺.) В стандартном модуле CRT имеются процедуры работы с курсором crsoff и crson. Ниже дано описание процедур putchar и putattr

```
unit crt1;
interface
procedure putAttr(x, y: shortint; attribute: byte);
procedure putChar(x, y: shortint; character: char);
implementation
uses crt, windows;
var
	stdout: HANDLE;
procedure putAttr(x, y: shortint; attribute: byte);
var
	c: COORD;
	dummy: longword;
begin
	attrW := attribute;
	c.x := x;
	c.y := y;
	WriteConsoleOutputAttribute(
		stdout, @attribute, 1, c, dummy);

end;
procedure putChar(x, y: shortint; character: char);
var
	c: COORD;
	dummy: longword;
begin
	c.x := x;
	c.y := y;
	WriteConsoleOutputCharacter(
		stdout, @character, 1, c, dummy);

end;
begin
	stdout := GetStdHandle(STD_OUTPUT_HANDLE);
end.
```