# Информационная система расчетов по договорам

Белорусский национальный технический университет

Международный институт дистанционного образования

Кафедра ПОВТ и АС

КУРСОВОЙ ПРОЕКТ

по курсу « Структуры и организация данных в ЭВМ »

На тему

« Информационная система расчетов по договорам »

Исполнитель ст. гр.417313 Я

Руководитель Романов А.В.

## Минск 2005г.

Введение

Без использования данных и структур, образованных элементами данных, не обходится ни одна программа для электронных вычислительных машин. Любые конструкции программы – нотации, операторы и т. д. – обязательно включают в себя идентификаторы некоторых данных и их совокупностей. Логическая схема структуры данных, представляющей собой совокупность взаимосвязанных данных, определяет не только внутреннее представление в памяти компьютера информационной модели некоторой предметной области (или ее составной части), но и, что самое главное, построение алгоритма, применяемого для обработки этой структуры. Так, например, алгоритм включения нового элемента в таблицу, организованную в памяти как вектор записей, не может быть использован для включения в иерархически организованный список. Как указывает Н. Вирт, определяющую роль в программировании играют не алгоритмы, а именно структуры данных, логическое и физическое построение которых являются главными факторами, влияющими на реализацию программируемой процедуры обработки.

Значимость той роли, которую играют структуры данных в процессе конструирования и кодирования программ для ЭВМ, послужила причиной появления в специальной «компьютерной» литературе множества работ, посвященных структурному подходу к организации данных. Было разработано значительное количество программных компонентов (многие из них получили название «стандартных»), применение которых существенно облегчило работу по обработке тех взаимосвязанных совокупностей данных, которые выбирают программисты для использования в своих программах. В учебных программах высших учебных заведений появилась специальная дисциплина «Структуры и организация данных в ЭВМ», в рамках которой изучаются не только логические схемы различных структур данных и способы их физической организации, но и построение алгоритмов обработки таких структур (формирования, просмотр и т. д.). Важность такой дисциплины для специалистов по программному обеспечению трудно переоценить.

Тема данного курсового проекта – «Информационная система расчётов по договорам». При этом по заданию к курсовому проекту необходимо использовать структуру данных типа вектор и пирамидальную сортировку данных.

Базовым была взята ИСР Delphi, так как он позволяет с большой гибкостью оперировать различными данными, а также предоставляет практически неограниченные возможности по созданию пользовательских интерфейсов.

Ниже я приведу некоторые обоснования использования среды Delphi для разработки данного программного продукта.

Delphi - это комбинация нескольких важнейших технологий:

·          Высокопроизводительный компилятор в машинный код.

·          Объектно-ориентированная модель компонент.

·          Визуальное (а, следовательно, и скоростное) построение приложений из программных прототипов.

Проект данной курсовой работы представляет собой инструмент для управления информационной системой расчетов по договорам для коммерческой научно-производственной организации.

# 1.         Состав DELPHI-проекта

1.1.     Состав проекта

Данный проект состоит из одной формы Form1. На форме расположены следующие компоненты (см. рис1):

- компонент MainMenu1 – осуществляет общее управление программой, в частности сохранение файлов с данными, обновление данных из файлов, выход из программы.

- компонент BtnDel – кнопка в нижней части формы для удаления записей данных.

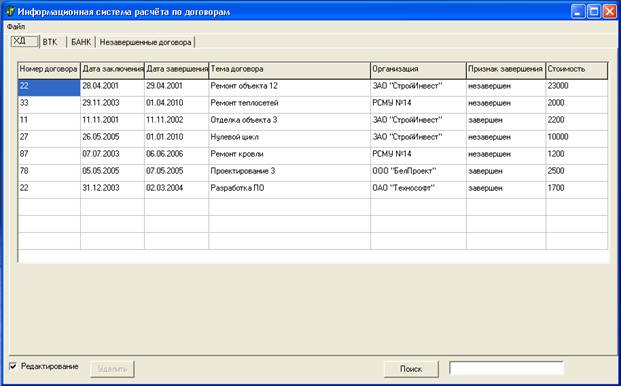
- компонент txtSearch – поле ввода искомых данных.

- компонент btnSearch – кнопка для начала поиска введенных данных в поле txtSearch.

- компонент CheckBox1 – соответственно для разрешения редактирования данных.

- компонент PageControl1 – содержит вкладки TabSheet 1÷4 на которых отражены данные (соответственно “ХД”, “ВТК”, “БАНК” и “Незавершенные договора”).

Компоненты TabSheet 1÷4 содержат в себе элементы таблицы (соответственно “XDgrid”, “WTKgrid”, “BANKgrid” и “NDgrid”). Кроме того, TabSheet 4 содержит ещё компонент GroupBox1 c кнопками btnSort1 и btnSort2 для сортировки списка незавершенных договоров по возрастанию и по убыванию количества членов ВТК.



# Рис.1 – Главная форма программы.

1.2 Основные модули и процедуры, входящие в состав программного комплекса

Список модулей:

Программа содержит модуль Unit1 – модуль интерфейсной формы проекта.

Список основных процедур, входящих в состав программного комплекса:

- procedure LoadFromFiles – процедура загрузки данных из файлов в одномерные массивы.

- procedure InitGrids – процедура инициализации таблиц и заполнения их в соответствии с массивами.

- procedure FillArrays – процедура заполнения массивов в соответствии с данными в таблицах.

- procedure SaveInFiles – процедура сохранения данных из массивов в файлы.

- procedure FillNDgrid – заполнение таблицы незавершенных договоров.

- procedure Sort – пирамидальная сортировка таблицы незавершенных договоров NDgrid по возрастанию.

- procedure Sort2– пирамидальная сортировка таблицы незавершенных договоров NDgrid по убыванию.

- procedure SweepRows(r1,r2:word) - замена местами строк в таблице незавершенных договоров NDgrid при сортировке.

- procedure SaveRow(var sr:SRow;r:word) – сохранение замененной строки.

2. Статические данные и структуры

В программе для хранения данных объявлено 5 одномерных строковых массива типа String[N] , где N≤255.

XDar: array [1..70] of String[30];

WTKar: array [1..150] of String[30];

BANKar: array [1..50] of String[30];

SRow=array [0..5] of String[30];

s: array [0..5] of String[30];

|  |  |  |
| --- | --- | --- |
| Имя массива | Тип | Размер в байтах |
| XDar | String[N] | (30+1)\*70=2170 |
| WTKar | String[N] | (30+1)\*150=4650 |
| BANKar | String[N] | (30+1)\*50=1550 |
| SRow | String[N] | (30+1)\*6=186 |
| S | String[N] | (30+1)\*6=186 |

Кроме того, в программе для временных нужд объявляются переменные:

nCol, i, j, y, x, n, n1, n2, c типа integer (каждая по 4 байта);

l, r типа word (каждая по 2 байта);

st, code, s импа string[30] (каждая по 30+1=31 байт).

# 3.         Логические структуры данных

Главным элементом и базовой структурой данного проекта являются обычные одномерные строковые массивы XDar, WTKar и BANKar размерностью 70, 150 и 50 соответственно.

Объявление массива выглядит следующим образом:

XDar: array [1..70] of String[30];

WTKar: array [1..150] of String[30];

BANKar: array [1..50] of String[30];

Массив (array) – это структура данных. Общим признаком всех массивов всех типов является возможность прямого доступа к их элементам со стороны программы. Эта возможность обеспечивается нумерацией элементов с помощью индекса, который обычно имеет целый тип.

Для логического определения массива ему необходимо происвоить имя, указать пару ограниченых значений индекса (или несколько пар граничных значений индексов), а также указать тип элементов.

Логическая схема структуры массива XDar:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | 0 | 1 | 2 | … | 30 |
| 1 |  |  |  |  |  |
| 2 |  |  |  |  |  |
| 3 |  |  |  |  |  |
| … |  |  |  |  |  |
| 70 |  |  |  |  |  |

Каждый элемент массива занимает 1 байт памяти. Соответственно массив XDar будет занимать (30+1)\*70=2170 байт.

Логическая схема структуры массива WTKar:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | 0 | 1 | 2 | … | 30 |
| 1 |  |  |  |  |  |
| 2 |  |  |  |  |  |
| 3 |  |  |  |  |  |
| … |  |  |  |  |  |
| 150 |  |  |  |  |  |

Каждый элемент массива занимает 1 байт памяти. Соответственно массив WTKar будет занимать (30+1)\*150=4650 байт.

Логическая схема структуры массива BANKar:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | 0 | 1 | 2 | … | 30 |
| 1 |  |  |  |  |  |
| 2 |  |  |  |  |  |
| 3 |  |  |  |  |  |
| … |  |  |  |  |  |
| 50 |  |  |  |  |  |

Каждый элемент массива занимает 1 байт памяти. Соответственно массив BANKar будет занимать (30+1)\*50=1550 байт.

# 4.         Алгоритмы обработки основных структур

Основной операцией обработки структуры в данном программном обеспечении является пирамидальная сортировка (по заданию на курсовое проектирование).

Данный вид сортировки не рекомендуется для небольшого числа элементов, как, скажем, в нашем программном обеспечении. Однако для большого количества элементов пирамидальная сортировка оказывается очень эффективной, и чем больше число элементов, тем эффективнее.

Пирамидальная сортировка требует N∙Log2N шагов даже в худшем случае. Такие отлиные характеристики для худшего случая – одно из самых выгодных качеств пирамидальной сортировки.

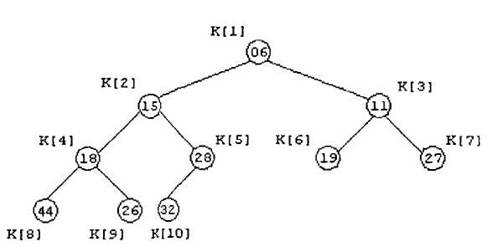
Но в принципе для данного вида сортиовки, видимо, больше всего подходят случаи, когда элементы более или менее рассортированы в обратном порядке, т.е. для нее характерно неестественное поведение. Очевидно, что при обратном порядке фаза построения пирамиды не требует никаких пересылок.

Пирамида определяется как некоторая последовательность ключей

K[L], ..., K[R], такая, что

K[i] ≤ K[2i] & K[i] ≤ K[2i + 1],                                                      (1)

для всякого i = L, ..., R/2. Если имеется массив К[1], К[2], ..., К[R], который индексируется от 1, то этот массив можно представить в виде двоичного дерева. Пример такого представления при R=10 показан на рисунке 2.

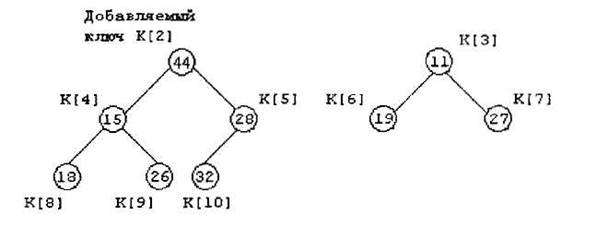


# Рис.2 – Массив ключей, представленный в виде двоичного дерева

Дерево, изображенное на рисунке 2, представляет собой пирамиду, поскольку для каждого i = 1, 2, ..., R/2 выполняется условие (1). Очевидно, последовательность элементов с индексами i = R/2+1, R/2+2, ...., R (листьев двоичного дерева), является пирамидой, поскольку для этих индексов в пирамиде нет сыновей.

Способ построения пирамиды «на том же месте» был предложен Р. Флойдом. В нем используется процедура просеивания (sift), которую рас-смотрим на следующем примере.

Предположим, что дана пирамида с элементами К[3], К[4], ..., К[10] нужно добавить новый элемент К[2] для того, чтобы сформировать расши-ренную пирамиду К[2], К[3], К[4], ..., К[10]. Возьмем, например, исходную пирамиду К[3], ..., К[10], покачанную на рисунке 3, и расширим эту пирамиду «влево», добавив элемент К[2] =44.

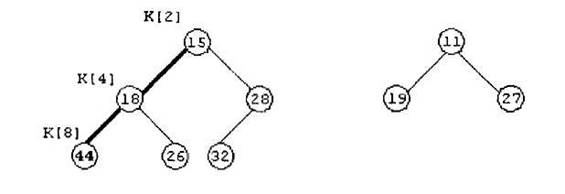


# Рис.3 – Пирамида, в которую добавляется ключ К[2]=44

Добавляемый ключ К[2] просеивается в пирамиду: его значение сравнивается с ключами узлов-сыновей, т.е. со значениями 15 и 28. Если бы оба слюча-сына были больше, чем просеиваемый ключ, то последний остался бы на месте, и просеивание было бы завершено. В нашем случае оба ключа-сына меньше, чем 44, следовательно, вставляемый ключ меняется местами с наименьшим ключом в этой паре, т.е. с ключом 15. Ключ 44 записывается в элемент К[4], а ключ 15 - в элемент К[2]. Просеивание продолжается, поскольку ключи-сыновья нового элемента К[4] оказываются меньше его - происходит обмен ключей 44 и 18. В результате получаем новую пирамиду, показанную на рисунке 4.

В нашем примере получалось так, что оба ключа-сына просеиваемого  
элемента оказывались меньше его. Это не обязательно: для инициализации  
обмена достаточно того, чтобы оказался меньше хотя бы один сыновей  
ключ, с которым и производится обмен.

Просеивание элемента завершается при выполнении любого из двух условий: либо у него не оказывается потомков в пирамиде, либо значение его ключа не превышает значений ключей обоих сыновей.



# Рис.4 – Просеивание ключа 44 в пирамиду.

Алгоритм просеивания в пирамиду допускает рекурсивную формулировку:

1)  просеивание элемента с индексом temp,

2)  при выполнении условий остановки - выход,

3)  определение индекса q элемента, с которым выполняется обмен,

4)  обмен элементов с индексами temp и q,

5)  temp:= q,

6)  перейти к п. 1.

Этот алгоритм в применении к нашему массиву а реализован в подпрограмме Sift, которая выполняет просеивания в пирамиду с максимальным индексом R:

Procedure Sift (temp, R: Integer);

Var q: integer;

x: TElement;

Begin

q:==2\*t;

If q > R Then Exit;

If q < R Then

If a[q-l].Key > a[q].Key Then q:= q + 1;

If a[temp-1].Key <= a[q-l].Key Then Exit;

x:= a[temp-1];

a [temp-1] := a[q-l];

a[q-l]:= x;

temp:= q;

Shift (temp, R);

End;

Процедура Shift учитывает индексацию вектора а от нуля.

Теперь рассмотрим процесс создания пирамиды из массива а[0], а[1], a[Highlndex]. Элементы этого массива индексируются от 0, а пирамида от 1. Ясно, что элементы a[N/2], a[N/2+1], ..., a[Highlndex] уже образуют пирамиду, поскольку не существует двух индексов i (i= N/2+1, N/2+2, …) и j, таких, что, j=2i (или j=2i+l). Эти элементы составляют последовательность, которую можно рассматривать как листья соответствующего двоичного дерева. Теперь пирамида расширяется влево: на каждом шаге добавляется новый элемент и при помощи просеивания помещается на соответствующее место. Этот процесс иллюстрируется следующим примером.

Процесс построения пирамиды

44 55 12 42 94 18 06 67

44 55 12 42 94 18 06 67

44 55 06 42 94 18 12 67

44 42 06 55 94 18 12 67

06 42 12 55 94 18 44 67

Примечание - жирным шрифтом отмечены ключи, образующие пирамиду на текущем шаге ее построения

Следовательно, процесс построения пирамиды из N элементов «на том же месте» можно описать следующим образом:

R:= N;

For i:= N Div 2 Downto 1 Do

Sift(i, R);

Для того, чтобы получить не только частичную, но и полную упорядоченность элементов нужно проделать N сдвигающих шагов, причем после каждого шага на вершину дерева выталкивается очередной (наименьший элемент). Возникает вопрос, где хранить «всплывающие» верхние элементы? Существует такой выход: каждый раз брать последнюю компоненту пирамиды (скажем, это будет х), прятать верхний элемент на место х, а х посылать в начало пирамиды в качестве элемента а[0] и просеивать его в нужное место. В следующей таблице приводятся необходимые в этом случае шаги:

Пример преобразования пирамиды в упорядоченную последовательность

06  42 12 55 94 18 44 67

12  42 18 55 94 67 44 06

18  42 44 55 94 67 12 06

42  55 44 67 94 18 12 06

44 55 94 67 42 18 12 06

55 67 94 44 42 18 12 06

67 94 55 44 42 18 12 0б

94 67 55 44 42 18 12 06 - Результат

Этот процесс описывается с помощью процедуры Sift следующим образом:

For R:= Highlndex Downto 1 Do Begin

x:=a[0]; a[0]:=a[R]; a[R]:=x;

Sift(1, R);

End;

Из примера сортировки видно, что на самом деле в результате мы получаем последовательность в обратном порядке. Но это легко можно исправить, изменив направление отношения порядка в процедуре Sift (в третьем и четвертом операторах If текста процедуры Sift, приведенного выше). В результате получаем следующую процедуру PyramidSort, учитывающую специфику индексации вектора а:

Procedure PyramidSort;

Var R, i: integer;

x: TElement;

Begin

R:= N;

For i:= N Div 2 Downto 1 Do

Sift(i, R);

For R:= Highlndex Downto 1 Do Begin

x:=a[0]; a[0]:= a[R]; a[R]:= x;

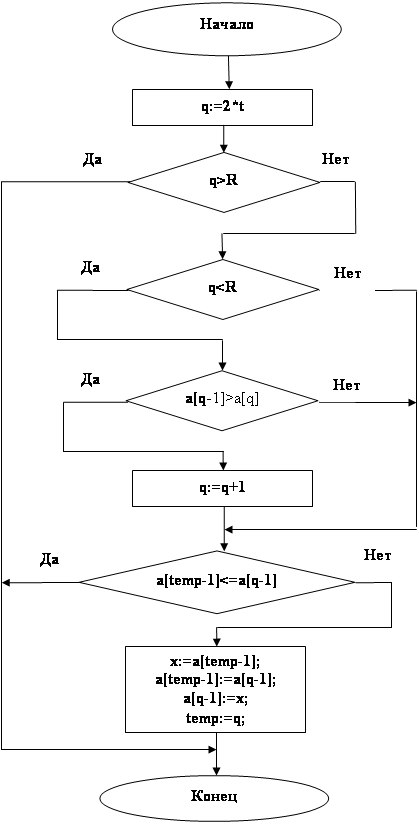
Sift(l, R);

End;

# Алгоритм просеивания для некоторого массива а можно представить в следующей блок-схеме:

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
|  |  | |  | | --- | | **Да** | |  |
|  |  |  |  |
|  |  | | |

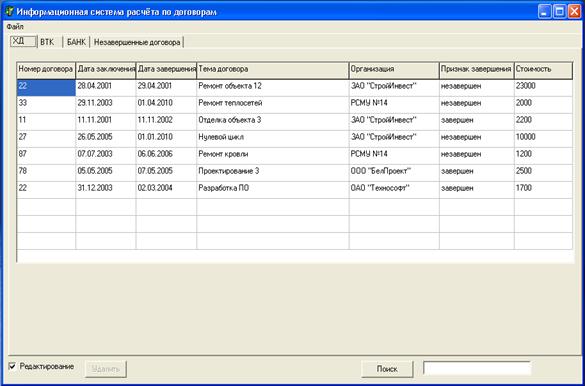
# 



# 5. Руководство пользователя

Данное програмное обеспечение имеет интуитивно понятный интерфейс и использует все богатейшие возможности пакета Borland Delphi.

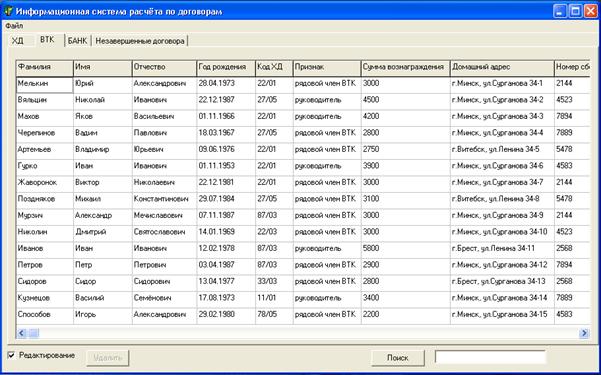
Программа имеет четыре вкладки. При первоначальном запуске активируется первая – вкладка хозяйственных договоров “ХД” (см. рис.5).



# Рис.5 – Вкладка ходяйственных договоров

# Здесь же можно внести изменения (предварительно отметив чекбокс «редактирование»), добавить или удалить новые записи.

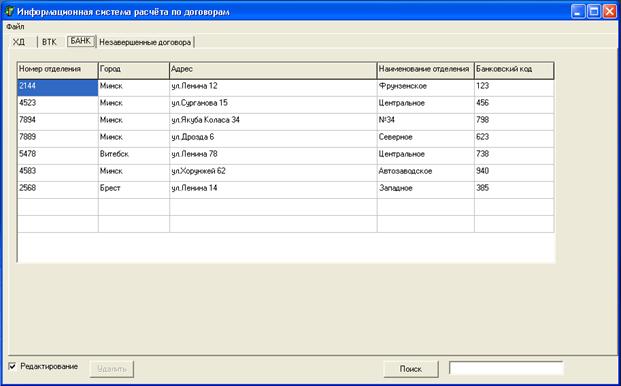
# Перейдя на вкладку временных трудовых коллективов “ВТК” (см. рис.6), мы попадем на список атрибутов исполнителей всех хоздоговоров, являющиеся членами соответствующих ВТК.



# Рис.6 – Вкладка списков атрибутов исполнителей всех хоздоговоров

# Здесь также можно редактировать списки, удалять их и добавлять новые. Нужно отметить, что “Код ХД” составляется из номера договора и года заключения договора.

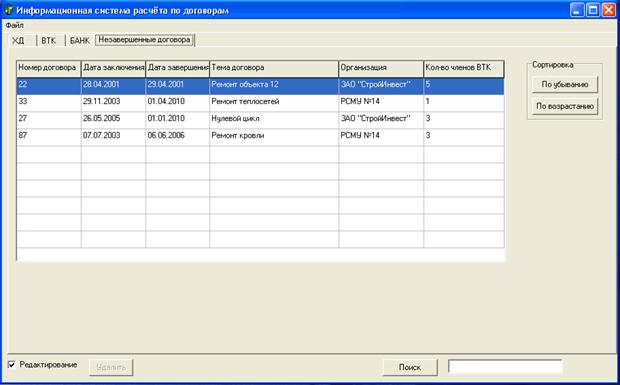
# Третья вкладка “Банк” содержит атрибуты отделений банков сбербанка, где имеются счета исполнителя договоров (см. рис.7).



# Рис.7 – Вкладка атрибутов отделений банков сбербанка.

# В данной вкладке как и в прдедыдущих можно радактировать атрибуты, удалять их, добавлять новые.

# И в четвертой вкладке находится упорядоченный по количеству членов ВТК (по возрастанию или по убыванию) всех незавершенных договоров с указанием части их атрибутов (согласно заданию к курсовому проектированию). Скриншот данной вкладки представлен на рис. 8.



# Рис.8 – Вкладка атрибутов незавершенных договоров

# Сортировка незавершенных договоров происходит или по убыванию, или по возрастанию количества членов ВТК пирамидальным методом (согласно заданию к курсовому проектированию).

# Из всех вкладок доступно меню “Файл”, в которм можно сохранить файлы с записями всей информации, обновить данную информацию или выйти из программы. Также доступен поиск отдельных элементов атрибутов.

# Заключение

В процессе разработки данного курсового проекта были изучены и закреплены знания по физическим размещениям структур данных и методам их обработки (сортировки). В ИСР DELPHI была разработана инормационная система расчётов по договорам. При создании программы не использовались компоненты баз данных данной ИСР.

Тестирование данного продукта показало хорошую работоспособность и отсутствие ошибок и недочётов в программе, а также в выполнении требований задания курсового проекта. Были изучены базовая структура данных типа вектор и пирамидальный метод сортировки.

Литература

1     Структуры и организация данных в компьютере. Учебное пособие / Лакин В.И., Романов А.В. – Мн.: БНТУ, 2004 – 176 с.

2     Архангельский А.Я. Delphi 6. Справочное пособие. - М.: ЗАО «Издательсво БИНОМ», 2001. - 1024 с.

3     Вирт Н. Алгоритмы и структуры данных. - СПб: Невский диалект, 2001. 352 с.

4     Гук М. Аппаратные средства IBM PC. Энциклопедия. - СПб: Питер, 2003. - 928 с.

5     Кнут Д.Э. Искусство программирования, том 1. Основные алгоритмы. - М.: Издательский дом «Вильямс», 2002. -720 с.

6     Кнут Д.Э. Искусство программирования, том 3. Сортировка и поиск. - М.: Издательский дом «Вильямс», 2001. - 832 с.

7     Лэнгсам Й., Огенстайн М., Тененбаум А. Структура данных для персональных ЭВМ. – М.: Мир, 1989. – 475 с.

8     Фаронов В. Система программирования DELPHI. -СПб.: «БВХ-Петербург», 2004. - 887 с.

# Приложение 1

# Листинг программы

unit Unit1;

interface

uses

Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,

ComCtrls, Grids, Menus, StdCtrls, ExtCtrls;

type

// тип для хранения строки таблицы NDgrid

SRow=array [0..5] of String[30];

TForm1 = class(TForm)

PageControl1: TPageControl;

TabSheet1: TTabSheet;

XDgrid: TStringGrid;

TabSheet2: TTabSheet;

WTKgrid: TStringGrid;

TabSheet3: TTabSheet;

BANKgrid: TStringGrid;

MainMenu1: TMainMenu;

N1: TMenuItem;

N2: TMenuItem;

N3: TMenuItem;

btnSearch: TButton;

txtSearch: TEdit;

TabSheet4: TTabSheet;

NDgrid: TStringGrid;

N4: TMenuItem;

CheckBox1: TCheckBox;

btnDel: TButton;

GroupBox1: TGroupBox;

btnSort1: TButton;

btnSort2: TButton;

procedure FormCreate(Sender: TObject);

procedure N2Click(Sender: TObject);

procedure N3Click(Sender: TObject);

procedure btnSearchClick(Sender: TObject);

procedure N4Click(Sender: TObject);

procedure CheckBox1Click(Sender: TObject);

procedure btnDelClick(Sender: TObject);

procedure btnSort2Click(Sender: TObject);

procedure btnSort1Click(Sender: TObject);

private

{ Private declarations }

XDar: array [1..70] of String[30]; {массив для ХД}

WTKar: array [1..150] of String[30]; {массив для ВТК}

BANKar: array [1..50] of String[30]; {массив для БА}

public

{ Public declarations }

procedure LoadFromFiles;

procedure InitGrids;

procedure FillArrays;

procedure SaveInFiles;

procedure FillNDgrid;

procedure Sort;

procedure Sort2;

procedure SweepRows(r1,r2:word);

procedure SaveRow(var sr:SRow;r:word);

end;

var

Form1: TForm1;

implementation

{$R \*.DFM}

{Процедура загрузки файлоы в массивы}

procedure TForm1.LoadFromFiles;

var

F:TextFile;

i:integer;

begin

{Файл - 'XD.txt'}

// открываем файл для чтения

AssignFile(F,'XD.txt');

Reset(F);

if IOResult <> 0 then

// ошибка открытия файла!

begin

{$I+}

MessageBox(0,'Ошибка!','Не возможно открыть файл XD.txt!',MB\_OK);

exit;

end;

{$I+}

// считываем файл построчно до конца и заполняем массив XDar

i:=1;

while not(SeekEof(F))do

begin

ReadLn(F,XDar[i]);

inc(i);

end;

CloseFile(F); // закрыть файл

{Файл - 'WTK.txt'}

{$I-}

// открываем файл для чтения

AssignFile(F,'WTK.txt');

Reset(F);

if IOResult <> 0 then

// ошибка открытия файла!

begin

{$I+}

MessageBox(0,'Ошибка!','Не возможно открыть файл WTK.txt!',MB\_OK);

exit;

end;

{$I+}

// считываем файл построчно до конца и заполняем массив XDar

i:=1;

while not(SeekEof(F))do

begin

ReadLn(F,WTKar[i]);

inc(i);

end;

CloseFile(F); // закрыть файл

{Файл - 'BANK.txt'}

{$I-}

// открываем файл для чтения

AssignFile(F,'BANK.txt');

Reset(F);

if IOResult <> 0 then

// ошибка открытия файла!

begin

{$I+}

MessageBox(0,'Ошибка!','Не возможно открыть файл BANK.txt!',MB\_OK);

exit;

end;

{$I+}

// считываем файл построчно до конца и заполняем массив XDar

i:=1;

while not(SeekEof(F))do

begin

ReadLn(F,BANKar[i]);

inc(i);

end;

CloseFile(F); // закрыть файл

end;

{Процедура инициализации таблицы и заполнение ее в соответсвии с массивами}

procedure TForm1.InitGrids;

var i,j:integer;

begin

XDgrid.Cells[0,0]:='Номер договора';

XDgrid.Cells[1,0]:='Дата заключения';

XDgrid.Cells[2,0]:='Дата завершения';

XDgrid.Cells[3,0]:='Тема договора';

XDgrid.Cells[4,0]:='Организация';

XDgrid.Cells[5,0]:='Признак завершения';

XDgrid.Cells[6,0]:='Cтоимость';

WTKgrid.Cells[0,0]:='Фамилия';

WTKgrid.Cells[1,0]:='Имя';

WTKgrid.Cells[2,0]:='Отчество';

WTKgrid.Cells[3,0]:='Год рождения ';

WTKgrid.Cells[4,0]:='Код ХД ';

WTKgrid.Cells[5,0]:='Признак';

WTKgrid.Cells[6,0]:='Сумма вознаграждения ';

WTKgrid.Cells[7,0]:='Домашний адрес ';

WTKgrid.Cells[8,0]:='Номер сбербанка';

WTKgrid.Cells[9,0]:='Расчетный счет ';

BANKgrid.Cells[0,0]:='Номер отделения';

BANKgrid.Cells[1,0]:='Город';

BANKgrid.Cells[2,0]:='Адрес ';

BANKgrid.Cells[3,0]:='Наименование отделения ';

BANKgrid.Cells[4,0]:='Банковский код ';

NDgrid.Cells[0,0]:='Номер договора';

NDgrid.Cells[1,0]:='Дата заключения';

NDgrid.Cells[2,0]:='Дата завершения';

NDgrid.Cells[3,0]:='Тема договора';

NDgrid.Cells[4,0]:='Организация';

NDgrid.Cells[5,0]:= 'Кол-во членов ВТК';

for i:=1 to 10 do

begin

for j:=1 to 7 do

XDgrid.Cells[j-1,i]:=XDar[(i-1)\*7+j];

end;

for i:=1 to 15 do

begin

for j:=1 to 10 do

WTKgrid.Cells[j-1,i]:=WTKar[(i-1)\*10+j];

end;

for i:=1 to 10 do

begin

for j:=1 to 5 do

BANKgrid.Cells[j-1,i]:=BANKar[(i-1)\*5+j];

end;

end;

{Заполним массивы в соответсвии с данными в таблице}

procedure TForm1.FillArrays;

var i:integer;

begin

for i:=0 to 69 do

begin

XDar[i+1]:=XDgrid.Cells[(i mod 7),(i div 7)+1];

end;

for i:=0 to 149 do

begin

WTKar[i+1]:=WTKgrid.Cells[(i mod 10),(i div 10)+1];

end;

for i:=0 to 49 do

begin

BANKar[i+1]:=BANKgrid.Cells[(i mod 5),(i div 5)+1];

end;

end;

{Сохраним данные из массивов в файл}

procedure TForm1.SaveInFiles;

var

F:TextFile; // текстовый файл

i:integer;

begin

{XD.txt}

// открываем файл для записи

AssignFile(F,'XD.txt');

Rewrite(F);

// построчно записываем из массива в файл

for i:=1 to 70 do

WriteLn(F,XDar[i]);

CloseFile(F); // закрыть файл

{WTK.txt}

// открываем файл для записи

AssignFile(F,'WTK.txt');

Rewrite(F);

// построчно записываем из массива в файл

for i:=1 to 150 do

WriteLn(F,WTKar[i]);

CloseFile(F); // закрыть файл

{BANK.txt}

// открываем файл для записи

AssignFile(F,'BANK.txt');

Rewrite(F);

// построчно записываем из массива в файл

for i:=1 to 50 do

WriteLn(F,BANKar[i]);

CloseFile(F); // закрыть файл

end;

procedure TForm1.FormCreate(Sender: TObject);

begin

LoadFromFiles; // загрузка данных из файла в массивы

InitGrids; // инициализация таблицы

FillNDgrid; // заполнить таблицу незавершенных договоров

end;

{Выход}

procedure TForm1.N2Click(Sender: TObject);

begin

Halt;

end;

{Сохранить}

procedure TForm1.N3Click(Sender: TObject);

begin

FillArrays; // сначала заполним массивы в соответсвии с таблицами

SaveInFiles; // теперь сохраним в файл

end;

{Поиск}

procedure TForm1.btnSearchClick(Sender: TObject);

var

myRect: TGridRect;

Grid: TStringGrid;

nCol,i,j:integer;

st:String;

begin

st:=txtSearch.Text; // строка для поиска

// определяем активную закладку

case PageControl1.ActivePageIndex of

0: begin Grid:= XDgrid; nCol:=7; end;

1: begin Grid:=WTKgrid; nCol:=10; end;

2: begin Grid:=BANKgrid; nCol:=5; end;

end;

myRect.Left := 11;

myRect.Top := 11;

myRect.Right := 11;

myRect.Bottom := 11;

Grid.Selection:= myRect;

if(st=' ') or (st='') then exit;

// поиск

for i:=1 to 10 do

for j:=0 to nCol-1 do

if Grid.Cells[j,i]=st then

begin

myRect.Left := j;

myRect.Top := i;

myRect.Right := j;

myRect.Bottom := i;

Grid.Selection := myRect;

exit;

end;

end;

{Заполнить таблицу незавершенных договоров в соответсвии с

XDgrid, WTKgrid, BANKgrid}

procedure TForm1.FillNDgrid;

var i,j,y,n:integer;

code:string;

st:string;

begin

j:=1;

for i:=1 to 10 do

if(XDGrid.Cells[5,i]='незавершен') then

begin

NDgrid.Cells[0,j]:=XDGrid.Cells[0,i];

NDgrid.Cells[1,j]:=XDGrid.Cells[1,i];

NDgrid.Cells[2,j]:=XDGrid.Cells[2,i];

NDgrid.Cells[3,j]:=XDGrid.Cells[3,i];

NDgrid.Cells[4,j]:=XDGrid.Cells[4,i];

// составим код договора

code:= NDgrid.Cells[0,j]+'/'+ NDgrid.Cells[1,j][9]+ NDgrid.Cells[1,j][10];

// найдем сколько человек состоит именно в этом ВТК

n:=0;

for y:=1 to 15 do

if(WTKgrid.Cells[4,y]=code) then inc(n);

str(n,st);

NDgrid.Cells[5,j]:=st;

inc(j);

end;

end;

{Обновить - обновить таблицу незавершенных договоров}

procedure TForm1.N4Click(Sender: TObject);

var i,j:integer;

begin

for i:=1 to 10 do

for j:=0 to 5 do

NDgrid.Cells[j,i]:='';

FillNDgrid;

end;

{Переключение флага редактирования}

procedure TForm1.CheckBox1Click(Sender: TObject);

var opt:TGridOptions;

begin

opt:=XDgrid.Options;

if CheckBox1.Checked=false then

begin

Include(opt,goRowSelect);

Exclude(opt,goEditing);

btnDel.Enabled := true;

end

else

begin

Exclude(opt,goRowSelect);

Include(opt,goEditing);

btnDel.Enabled := false;

end;

XDgrid.Options := opt;

WTKgrid.Options := opt;

BANKgrid.Options := opt;

end;

{Удалить}

procedure TForm1.btnDelClick(Sender: TObject);

var

myRect: TGridRect;

Grid: TStringGrid;

nCol,i,j:integer;

begin

// определяем активную закладку

case PageControl1.ActivePageIndex of

0: begin Grid:= XDgrid; nCol:=7; end;

1: begin Grid:=WTKgrid; nCol:=10; end;

2: begin Grid:=BANKgrid; nCol:=5; end;

end;

if(Grid.Row>0) and (Grid.Row<10) then

for i:=Grid.Row to 10 do

begin

for j:=0 to nCol-1 do

Grid.Cells[j,i]:=Grid.Cells[j,i+1];

end;

end;

{пирамидальная сортировка таблицы NDgrid по возростанию}

procedure TForm1.Sort;

var

l,r:word;

x,n,n1,n2,c,y:integer;

s:string;

sr:SRow;

procedure Sift;

label l3;

var i,j,y:word;

begin

i:=l;j:=2\*i;s:=NDgrid.Cells[5,i];SaveRow(sr,i);

while j<=r do

begin

if j<r then

begin

Val(NDgrid.Cells[5,j],n1,c);

Val(NDgrid.Cells[5,j+1],n2,c);

if n1<n2 then j:=j+1;

end;

Val(s,n1,c);

Val(NDgrid.Cells[5,j],n2,c);

if n1>=n2 then goto l3;

for y:=0 to 5 do

NDgrid.Cells[y,i]:=NDgrid.Cells[y,j];

i:=j; j:=2\*i;

end;

l3:

for y:=0 to 5 do

begin

NDgrid.Cells[y,i]:=sr[y];

end;

end; // Sift

begin

n:=0;

for y:=1 to 10 do

if (NDgrid.Cells[5,y]<>'') and (NDgrid.Cells[5,y]<>' ') then

inc(n);

l:=(n div 2)+1;r:=n;

while l>1 do

begin

l:=l-1; Sift;

end;

while r>1 do

begin

SaveRow(sr,1);

s:=NDgrid.Cells[5,1];

SweepRows(1,r);

r:=r-1; Sift;

end;

end; // Sort

{пирамидальная сортировка таблицы NDgrid по убыванию}

procedure TForm1.Sort2;

var

l,r:word;

x,n,n1,n2,c,y:integer;

s:string;

sr:SRow;

procedure Sift;

label l3;

var i,j,y:word;

begin

i:=l;j:=2\*i;s:=NDgrid.Cells[5,i];SaveRow(sr,i);

while j<=r do

begin

if j<r then

begin

Val(NDgrid.Cells[5,j],n1,c);

Val(NDgrid.Cells[5,j+1],n2,c);

if n1>n2 then j:=j+1;

end;

Val(s,n1,c);

Val(NDgrid.Cells[5,j],n2,c);

if n1<=n2 then goto l3;

for y:=0 to 5 do

NDgrid.Cells[y,i]:=NDgrid.Cells[y,j];

i:=j; j:=2\*i;

end;

l3:

for y:=0 to 5 do

begin

NDgrid.Cells[y,i]:=sr[y];

end;

end; // Sift

begin

n:=0;

for y:=1 to 10 do

if (NDgrid.Cells[5,y]<>'') and (NDgrid.Cells[5,y]<>' ') then

inc(n);

l:=(n div 2)+1;r:=n;

while l>1 do

begin

l:=l-1; Sift;

end;

while r>1 do

begin

SaveRow(sr,1);

s:=NDgrid.Cells[5,1];

SweepRows(1,r);

r:=r-1; Sift;

end;

end; // Sort2

{поменять местами строки r1 и r2 в таблице NDgrid}

procedure TForm1.SweepRows(r1,r2:word);

var s: array [0..5] of String[30];

i:integer;

begin

for i:=0 to 5 do

s[i]:=NDgrid.Cells[i,r1];

for i:=0 to 5 do

NDgrid.Cells[i,r1]:=NDgrid.Cells[i,r2];

for i:=0 to 5 do

NDgrid.Cells[i,r2]:=s[i];

end;

{сохранить строку номер r таблицы NDgrid в sr}

procedure TForm1.SaveRow(var sr:SRow;r:word);

var i:integer;

begin

for i:=0 to 5 do

sr[i]:=NDgrid.Cells[i,r];

end;

procedure TForm1.btnSort2Click(Sender: TObject);

begin

Sort;

end;

procedure TForm1.btnSort1Click(Sender: TObject);

begin

Sort2;

end;

end.