Министерство образования и науки Украины
Севастопольский национальный технический университет

Методические указания

к выполнению лабораторных работ по дисциплине «Операционные системы» для студентов дневной формы обучения направления 0.915 – «Компьютерная инженерия»
Часть 1.

Севастополь 2008

УДК 519.6

Методические указания к выполнению лабораторных работ по дисциплине «Операционные системы» для студентов дневной формы обучения направления 0.915 – «Компьютерная инженерия» / Сост. Г.Г.Сергеев. – Севастополь: Изд-во СевНТУ, 2008. –Ч.1. - 20с.

Целью методических указаний является оказание помощи студентам при выполнении лабораторных работ по дисциплине «Операционные системы».

Методические указания рассмотрены и утверждены на заседании кафедры КиВТ (протокол № 8 от 29 июня 2008 г.).

Допущено учебно-методическим центром СевНТУ в качестве методических указаний.

Рецензент Овчинников А.Л. – старший преподаватель кафедры информационных систем СевНТУ.

СОДЕРЖАНИЕ

ЛАБОРАТОРНАЯ РАБОТА №1. РАЗРАБОТКА ИНТЕРФЕЙСНЫХ ПРИЛОЖЕНИЙ В ОС WINDOWS

1.1 Цель работы

Изучить методы разработки и отладки программного обеспечения с использованием стандартных интерфейсных компонентов ОС Windows. Научиться строить приложения, управляемые событиями.

1.2 Методика выполнения работы

1. Изучить методические указания к работе.
2. Разработать в соответствии с вариантом задания приложение, состоящее из двух окон: «родительского» и «дочернего». Во всех вариантах в «родительском» окне выполняется ввод и коррекция исходных данных, а в «дочернем» – вывод результатов.
3. Обеспечить возможность чтения и сохранения исходных данных в текстовом или типизированном файле. Команда открыть (сохранить) реализуется с помощью «главного» меню с использованием диалоговых компонентов TOpenDialog, TSaveDialog.

1.3 Варианты заданий

Вариант задания (таблица 1.1) выбирается как остаток от деления номера последних двух цифр номера зачетной книжки на 20.

Таблица 1.1 - Варианты заданий

№ вар.	Язык програм.	Задание	Способ организации файла
0	Delphi	1	Текстовый
1	C++ Builder	1	Текстовый
2	Delphi	2	Текстовый
3	C++ Builder	2	Текстовый
4	Delphi	3	Текстовый
5	C++ Builder	3	Текстовый

Продолжение таблицы 1.1.

№ вар.	Язык програм.	Задание	Способ организации файла
6	Delphi	4	Текстовый
7	C++ Builder	4	Текстовый
8	Delphi	5	Текстовый
9	C++ Builder	5	Текстовый
10	Delphi	1	Типизированный
11	C++ Builder	1	Типизированный
12	Delphi	2	Типизированный
13	C++ Builder	2	Типизированный
14	Delphi	3	Типизированный
15	C++ Builder	3	Типизированный
16	Delphi	4	Типизированный
17	C++ Builder	4	Типизированный
18	Delphi	5	Типизированный
19	C++ Builder	5	Типизированный

Задания:

1. Дана матрица MxN. Найти сумму элементов каждой строки; найти минимальный элемент в каждой строке; найти сумму первого и максимального элемента.
2. Дан текст. Найти сколько раз в каждой строке встречается заданное слово; подсчитать количество слов в строке; сколько слов в строке содержит гласных букв больше, чем согласных.
3. Даны 2 квадратных матрицы одинаковой размерности. Найти разность между суммами строк первой и второй матрицы; считая строки матрицы векторами найти разность между длинами векторов.
4. Дан текст. Определить длину каждой строки; длину максимального слова в строке; количество гласных букв в строке.
5. Дана матрица. Найти число с наибольшей дробной частью в каждой строке; подсчитать сколько различных чисел в каждой строке; считая строки матрицы векторами найти длину каждого вектора.

Примеры реализации интерфейса для задач с матрицами и текстом представлены на рисунках 1.1 и 1.2.

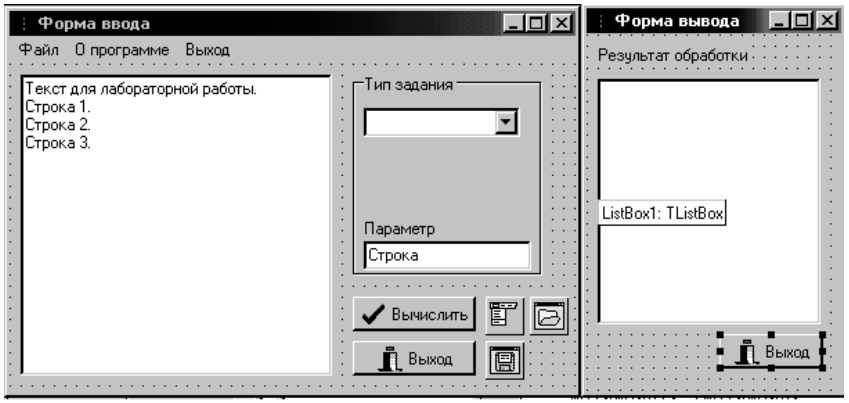

Рисунок 1.1 – Пример реализации интерфейса для задач обработки текста

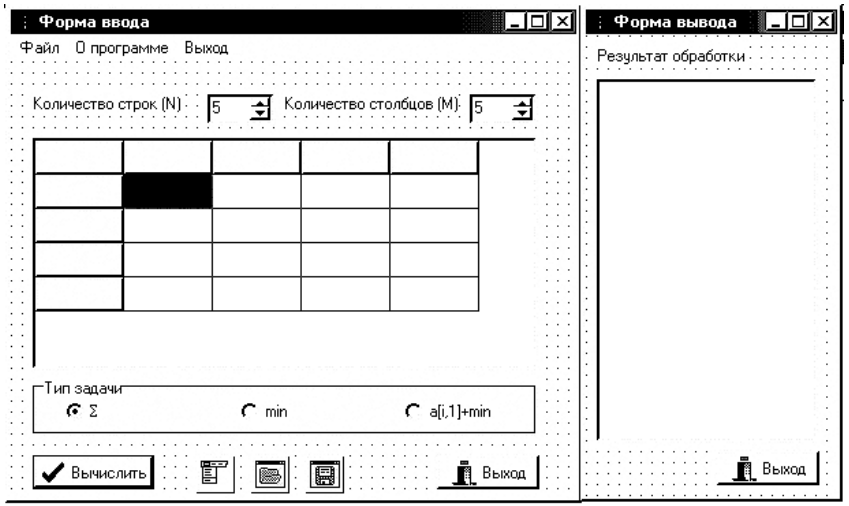

Рисунок 1.2 – Пример реализации интерфейса для задач обработки матрицы

1.4. Содержание отчета

Постановка задачи, описание метода решения задачи, текст программы, методика отладки и тестирования, выводы.

1.5. Контрольные вопросы

1. Сформулируйте базовые принципы объектно-ориентированного программирования.
2. В чем заключается принцип «визуального» программирования.
3. Что такое класс, объект, компонент. В чем общее и в чем различие между этими понятиями?
4. Какие интерфейсные компоненты используются в вашей программе. Приведите пример использования этих компонентов в интерфейсе Windows.
5. Что такое поток ввода-вывода. Какие свойства и методы потоков Вам известны?

ЛАБОРАТОРНАЯ РАБОТА №2. СИСТЕМНЫЕ ФУНКЦИИ API WINDOWS ДЛЯ ЗАПУСКА ПРОЦЕССОВ

2.1. Цель работы:

Изучить методы управления процессами в ОС Windows. Научиться использовать функции API для запуска и завершения процессов.

2.2. Методика выполнения работы

1. Изучить методические указания к работе.
2. Разработать средствами среды программирования, заданной вариантом задания программу для запуска и завершения дочернего процесса.
3. В качестве дочернего процесса использовать программу, разработанную в лабораторной работе №1. При запуске процесса использовать вызов функций WinExec и CreateProcess по выбору оператора. Для завершения процесса, созданного CreateProcess использовать функцию TerminateProcess.

2.3. Содержание отчета

Постановка задачи. Описание программы. Методика и результаты

испытаний. Выводы.

2.4. Варианты заданий

Вариант задания (таблица 2.1) выбирается как остаток от деления номера последних двух цифр номера зачетной книжки на 20.

Таблица 2.1 - Варианты заданий

№ вар.	Язык програм.	Способ отображения главного окна дочернего процесса	Приоритет процесса
0	Delphi	Свернутое	Обычный
1	C++ Builder	Свернутое	Обычный
2	Delphi	Максимизированное	Обычный
3	C++ Builder	Максимизированное	Обычный
4	Delphi	Минимизированное	Обычный
5	C++ Builder	Минимизированное	Обычный
6	Delphi	Норма	Обычный
7	C++ Builder	Норма	Обычный
8	Delphi	Свернутое	Низкий
9	C++ Builder	Свернутое	Низкий
10	Delphi	Максимизированное	Низкий
11	C++ Builder	Максимизированное	Низкий
12	Delphi	Минимизированное	Низкий
13	C++ Builder	Минимизированное	Низкий
14	Delphi	Норма	Низкий
15	C++ Builder	Норма	Низкий
16	Delphi	Максимизированное	Высокий
17	C++ Builder	Максимизированное	Высокий
18	Delphi	Норма	Высокий
19	C++ Builder	Норма	Высокий

2.5. Контрольные вопросы

1. Сформулируйте понятия: процесс, дескриптор процесса, контекст процесса.
2. Сформулируйте понятие потока. В чем отличие потока от процесса?

3. Какие функции для управления процессами API Windows Вы знаете?
4. Какую информацию содержат структуры startupinfo, processinformation?

2.6. Теоретическй раздел

Существует несколько способов запуска одной программы из другой.

WinExec - устаревшая функция, используется для совместимости с 16-битной Windows. Не рекомендуется к использованию в Win32-приложениях.

```
UINT WinExec(  LPCSTR lpCmdLine,  // address of command line
               UINT uCmdShow  // window style for new application
            );
```

Параметры:

- lpCmdLine – указатель на 0-ограниченную строку символов с именем приложения и параметрами командной строки;
- uCmdShow – определяет способ отображения окна программы (SW_HIDE, SW_MAXIMIZE, SW_MINIMIZE, SW_RESTORE, SW_SHOW, SW_SHOWDEFAULT и др).

В случае, если произошла ошибка, функция возвращает значение меньшее 32.

Для 32-х разрядных приложений Windows рекомендуется использовать функцию CreateProcess.

```
BOOL CreateProcess(
      LPCTSTR lpApplicationName,
      LPTSTR lpCommandLine,
      LPSECURITY_ATTRIBUTES lpProcessAttributes,
      LPSECURITY_ATTRIBUTES lpThreadAttributes,
      BOOL bInheritHandles,
      DWORD dwCreationFlags,
      LPVOID lpEnvironment,
      LPCTSTR lpCurrentDirectory,
      LPSTARTUPINFO lpStartupInfo,
      LPPROCESS_INFORMATION lpProcessInformation
   );
```

Параметры:

- lpApplicationName - имя программы;
- lpCommandLine - параметры командной строки;
- lpProcessAttributes - атрибуты безопасности процесса (имеет смысл

только в NT/2000);
- lpThreadAttributes - атрибуты безопасности главного потока (имеет смысл только в NT/2000);
- bInheritHandles - если bInheritHandles == TRUE, то созданный процесс (запущенная программа), наследует дескрипторы (handles) запускающей программы;
- dwCreationFlags параметры создания. Здесь можно указать класс приоритета создаваемого процесса и некоторые дополнительные параметры;
- lpEnvironment указатель на блок окружения или NULL, тогда используется блок окружения родителя;
- lpCurrentDirectory - текущая директория или NULL, тогда используется текущая директория родителя;
- lpStartupInfo - указатель на структуру STARTUPINFO, которая определяет положение главного окна;
- lpProcessInformation - информация о созданном процессе.

Поле dwCreationFlags определяет приоритет и параметры порождаемого дочернего процесса. В частности, предусмотрены следующие классы приоритетов:

- HIGH_PRIORITY_CLASS – используется для наиболее важных процессов с минимумом времени реакции;
- IDLE_PRIORITY_CLASS – процесс получает процессорное время только когда система свободна;
- NORMAL_PRIORITY_CLASS – средний (обычный) приоритет;
- REALTIME_PRIORITY_CLASS - процесс реального времени.

Структура StrartupInfo содержит информацию о параметрах главного окна нового приложения. Наиболее простой способ ее заполнения – вызов функции GetStartupInfo(LPSTARTUPINFO si). Наиболее часто программисты используют следующие поля::

- Cb – Определяет размер структуры (в байтах);
- dwX, dwY – опредляют смещение X и Y при наличии флага STARTF_USEPOSITION.
- dwXSize, dwYSize – ширина и высота окна при наличии флага
- STARTF_USESIZE.
- dwFlags – битовое поле флагов значимости параметров. Если флаг установлен, то соответствующий ему параметр применяется при создании окна. К этим флагам относятся упомянутые выше STARTF_USEPOSITION и STARTF_USESIZE – имеет тот же смысл, что и параметр uCmdShow
- cbReserved2 – резерв, всегда должен иметь значение 0.
- lpReserved2 – резерв, должен быть NULL.

Структура lpProcessInformation предназначена для хранения данных о дочернем процессе:
- HANDLE hProcess – дескриптор процесса;
- HANDLE hThread – дескриптор «главного» потока процесса;
- DWORD dwProcessId – глобальный идентификатор процесса;
- DWORD dwThreadId – глобальный идентификатор «главного» потока.

Ниже приводятся пример вызова приложения project1.exe из вложенного каталога testprocess и его завершения через 1 секунду после старта.

```
var
   ppath  : pchar;
   path   : shortstring;
   sinfo  : startupinfo;
   pinfo  : process_information;
begin
  path:=ExtractFilePath(paramstr(0))+'testprocess\project1.exe'+#0;
  ppath:=@path[1];
//  GetStartupInfo(sinfo); - Возможно получение sinfo от родительского процесса
  sinfo.cb:=sizeof(sinfo);
  sinfo.lpReserved:=Nil;
  sinfo.lpDesktop:=Nil;
  sinfo.lpTitle:=Nil;
  sinfo.cbReserved2:=0;
  sinfo.lpReserved2:=Nil;
  if not CreateProcess( Nil,  // lpApplicationName
       ppath,     //   lpCommandLine
        Nil,        // lpProcessAttributes,
        Nil,        //   lpThreadAttributes,
        false,       //  bInheritHandles,
        NORMAL_PRIORITY_CLASS,       //  dwCreationFlags,
        NIL,    // lpEnvironment
        NIL,    // lpCurrentDirectory
        sinfo,  //  lpStartupInfo,
         pinfo       //    lpProcessInformation
    ) then showmessage('Error');
  sleep(1000);
  TerminateProcess(pinfo.hProcess,1);
end;
```

ЛАБОРАТОРНАЯ РАБОТА №3. ОБРАБОТКА СООБЩЕНИЙ WINDOWS

3.1. Цель работы:

Изучить методы обработки сообщений в ОС Windows.

3.2. Методика выполнения работы

1. Изучить методические указания к работе.
2. Разработать программу для генерации и визуализации массива из 100 целых чисел.
3. Написать процедуру для обработки массива в соответствии с вариантом задания.
4. Разработать форму для вывода результатов обработки.
5. Передать в форму вывода результат обработки в виде параметра сообщения с кодом WM_USER+N+100, где N – номер варианта задания.

3.3. Варианты заданий

Вариант задания (таблица 3.1) выбирается как остаток от деления номера последних двух цифр номера зачетной книжки на 10.

Таблица 3.1 - Варианты заданий

№	Среда программирования	Функция обработки
0	Delphi	Минимум
1	C++ Builder	Минимум
2	Delphi	Максимум
3	C++ Builder	Максимум
4	Delphi	Среднее арифметическое
5	C++ Builder	Среднее арифметическое
6	Delphi	Сумма
7	C++ Builder	Сумма
8	Delphi	Сумма модулей элементов
9	C++ Builder	Сумма модулей элементов

3.4. Содержание отчета

Постановка задачи. Описание программы. Методика и результаты

испытаний. Выводы.

3.5. Контрольные вопросы

1. Для чего необходим передачи и обработки сообщений?
2. Какие действия надо выполнить для генерации сообщения?
3. Какие действия надо выполнить для приема сообщения?
4. Какие параметры сопровождают сообщения?
5. Почему в программах рекомендуется нумеровать собственные сообщения, начиная с кода WM_USER?

3.6 Теоретический раздел

Изменения в состоянии аппаратных средств, самой операционной системе, других исполняющихся в данный момент приложениях, которые могут оказать воздействие на ход выполнения задачи, называются событиями. Приложения извещаются о событиях через сообщения, которые им посылает Windows.

Отправку сообщений можно выполнить с помощью API-функции Windows:

```
LRESULT SendMessage(
   HWND hWnd,  // Дескриптор  окна назначения
   UINT Msg,  // Код сообщения
   WPARAM wParam,     // Первый парметр
   LPARAM lParam   //Второй параметр  );
```

Общий синтаксис для декларации обработчика сообщений Windows в среде визуального программирования Delphi:

procedure Handler_Name(var Msg : MessageType); message WM_XXXXX;

Handler_Name обозначает имя метода; Msg - имя передаваемого параметра; MessageType - какой либо тип записи, подходящий для данного сообщения; директива message указывает, что данный метод является обработчиком сообщения; WM_XXXXX - константа или выражение, которое определяет номер обрабатываемого сообщения Windows.

Предлагаемый фрагмент кода иллюстрирует объявление обработчика событий в разделе описания класса:

```
Const WM_MY=WM_USER+10;
 type
  TForm1 = class(TForm)
  private
```

```
    { Private declarations }
      procedure WMWork(var Msg : TMessage); message  WM_MY;
  public
    { Public declarations }
  end;
```

Константа WM_USER определяет код сообщения, начиная с которого можно безопасно производить их обработку без риска коллизий с системными сообщениями.

Далее, в секции implementation нужно определить обработчик:

```
procedure TForm1.WMWork(var Msg : TMessage);
begin
  …
  ResEdit.Text:=IntToStr(Msg.wParam)
end;
```

В данном случае при получении формой сообщения с кодом WM_MY в сроке редактирования ResEdit будет выведен первый параметр.

Альтернативным способом определения реакции приложения на прием сообщения, является переопределение обработчика событий OnMessage объекта Application, который автоматически создается при запуске программы. Следующий код выполняет те же функции, что и приведенный ранее

```
procedure TForm1.FormCreate(Sender: TObject);
begin
 Application.OnMessage:=AOM;
end;

procedure TForm1.AOM(var Msg: TMsg; var Handled: Boolean);
begin
 Handled:=False;
 if Msg.Message = WM_MY then
   begin
     ResEdit.Text:=IntToStr(Msg.wParam)
    Handled:=True;
 end;
end;
```

В обработчике сообщений нельзя выполнять операции, требующие длительного времени, поскольку это приведет к замедлению выполнения

всего приложения.

ЛАБОРАТОРНАЯ РАБОТА №4. ОБМЕН ДАННЫМИ МЕЖДУ ПРИЛОЖЕНИЯМИ ЧЕРЕЗ ФАЙЛ, ОТОБРАЖАЕМЫЙ В ПАМЯТЬ

4.1. Цель работы:

Изучить методы обмена данными между приложениями, в частности через файлы, отображаемые в память.

4.2. Методика выполнения работы

1. Изучить методические указания к работе.
2. Разработать средствами среды программирования, заданной вариантом задания, программу для подготовки данных. Программа должна иметь средства ввода и коррекции одномерного массива из 10 чисел.
3. Разработать программу для обработки данных одномерного массива и визуализации результата в соответствии с вариантом задания.
4. Разработать механизм передачи данных из первой программы во вторую посредством файла, отображаемого в память. Процедура обработки инициируется приемом сообщения с кодом WM_USER+N+100, где N – номер варианта задания.

4.3. Содержание отчета

Постановка задачи. Описание программы. Методика и результаты испытаний. Выводы.

4.4. Варианты заданий

Вариант задания (таблица 4.1) выбирается как остаток от деления номера последних двух цифр номера зачетной книжки на 10.

Таблица 4.1 - Варианты заданий

№	Среда программирования	Функция обработки
0	Delphi	Минимум
1	C++ Builder	Минимум
2	Delphi	Максимум
3	C++ Builder	Максимум
4	Delphi	Среднее арифметическое
5	C++ Builder	Среднее арифметическое
6	Delphi	Сумма
7	C++ Builder	Сумма

№	Среда программирования	Функция обработки
8	Delphi	Сумма модулей элементов
9	C++ Builder	Сумма модулей элементов

4.5. Контрольные вопросы

1. Что такое файл, отображаемый в память? Чем он отличается от обычного?
2. Какие действия надо выполнить для создания файла?
3. Какие действия надо выполнить для открытия файла?
4. Каким образом осуществляется чтение и запись данных?
5. Что значит «отобразить файл на адресное пространство процесса»? Какая API-функцияя для этого используется?

4.6. Теоретический раздел

Отображение файла данных в адресное пространство процесса предоставляет мощный механизм работы с файлами - программа может работать с файлом, как с массивом ячеек памяти. Само проецирование файла в память выполняется в три этапа:

1) Создается объект ядра "файл". Для создания объекта "файл" используется функция CreateFile().
2) С помощью функции CreateFileMapping() создается объект ядра "отображаемый файл". При этом используется дескриптор файла (handle), возвращенный функцией CreateFile().
3) Функцией MapViewOfFile() производится отображение объекта "отображаемый файл" или его части в адресное пространство процесса.

Для откреплення файла от адресного пространства процесса используется функция UnmapViewOfFile(), а для уничтожения объектов "файл" и "отображаемый файл" - функция CloseHandle.

API-Функция создания отображаемого в память файла определяется как

```
HANDLE CreateFileMapping(
  HANDLE hFile,     // Дескриптор «реального»  файла на диске
                    //FFFFFFFFH, если файл не существует
  LPSECURITY_ATTRIBUTES lpFileMappingAttributes,
                    //Атрибут защиты ОС, обычно равен NIL
  DWORD flProtect,  // Вид защиты доступа
                    //PAGE_READONLY   - только для чтения
```



```
                    //PAGE_READWRITE -  для чтения и записи
   // Максимальный размер   файла(2 двойных слова)
   DWORD dwMaximumSizeHigh,
   DWORD dwMaximumSizeLow,
   //Имя файла для регистрации в ОС
   LPCTSTR lpName    );
```

API-Функция открытия существующего файла:

```
HANDLE OpenFileMapping(
  // Вид защиты доступа
   DWORD dwDesiredAccess,
  //Признак наследования, обычно равен true
   BOOL bInheritHandle,
   //Имя файла для регистрации в ОС
   LPCTSTR lpName
  );
```

API-Функция отображения файла на адресное пространство процесса:

```
LPVOID MapViewOfFile(

   HANDLE hFileMappingObject,   // Дескриптор файла
   DWORD dwDesiredAccess,    // Способ доступа
  //Смещение относительно начала файла
   DWORD dwFileOffsetHigh,
   DWORD dwFileOffsetLow,
   //Количество отображаемых байт
   DWORD dwNumberOfBytesToMap
  );
```

Два процесса могут совместно использовать объект "отображаемый файл". При этом с помощью функции MapViewOfFile() каждый процесс отображает этот объект в свое адресное пространство и применяет эту часть адресного пространства как совместно используемую область данных. Если один процесс изменяет совместно используемую область данных, то она изменяется и для другого разделяющего ее процесса. Операционная система обеспечивает когерентность совместно используемой области данных для всех процессов, но для этого процессы должны работать с объектом "отображаемый файл", а не с самим файлом.

Отображаемый в память файл может и не иметь «образ» на жестком диске. В этом случае в качестве дескриптора файла при вызове функции CreateFileMapping указывается значение FFFFFFFFH.

Приведенный ниже пример иллюстрирует процесс записи программой подготовки данных в файл отображаемый в память (процедура SendData) и чтения данных программой-обработчиком (процедура GetData).

```
procedure TFirstForm.SendData;
var
        FMH, handle   : integer;
        FVIA          : pointer;
begin
      handle:=FindWindow('TSecondForm','Обработчик');
      FMH:=CreateFileMapping($FFFFFFFF,NIL,PAGE_READWRITE,
    0,1000,‘MyMapFile');
      if FMH = 0 then   begin ShowMessage('Ошибка '); exit; end;
      FVIA:= MapViewOfFile(FMH,File_Map_All_Access,0,0,0);
      if FVIA= nil then   begin ShowMessage('Ошибка '); exit; end;
      //запись данных
      move(OutputArray,FViewInAddress^,Sizeof(OutputArray));
      SendMessage(handle,WM_READDATA,0,0);
      //Оповещение второго процесса, что данные отправлены
       UnmapViewOfFile(FViewInAddress);
      //Завершение работы с файлом
       CloseHandle(FMapHandle);
      End;

procedure TSecondForm.GetData;
var
        FMH   : integer;
        FVIA  : pointer;
begin
      FMH := OpenFileMapping(FILE_MAP_ALL_ACCESS,true, ‘MyMap-
File');
      if FMH= 0 then   begin ShowMessage('Ошибка '); exit; end;
      FVIA := MapViewOfFile(FMapInHandle,File_Map_All_Access,0,0,0);
      if FVIA = nil then begin ShowMessage('Ошибка '); exit; end;
      // Чтение данных
      move(FViewInAddress^, InputArray,sizeof(InputArray));
      UnmapViewOfFile(FViewInAddress);
      //Завершение работы с файлом
       CloseHandle(FMapInHandle);
 end;
```

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Олифер В.Г. Сетевые операционные системы / В.Г. Олифер, Н.А. Олифер. – СПб.:Питер, 2001. – 544 с.

2. Корняков В. Н. Программирование документов и приложений MS Office в Delphi / В.Н. Корняков . —СПб.: БХВ-Петербург, 2005. — 496 с.

3. Богачев К.Ю. Основы параллельного программирования / К.Ю. Богачев – М.:Бином, 2001. – 336 с.

4. Сухарев М.В. Основы Delphi. Профессиональный подход / М.В. Сухарев. – М.:Наука и техника, 2004. – 623 с.

Заказ № от Тираж экз.
Изд-во СевНТУ