Как видим, команда разработчиков MASM32 позаботилась не только о простыне прототипов, но и о "независимости" нашего исходника от выбранной кодировки. То есть, для того чтобы "перезаточить" программу под UNICODE, нам вовсе не нужно заменять окончание A на W в имени функции. Достаточно просто приинклюдить другой файл с прототипами и эквивалентами наподобие

```
WriteConsoleW PROTO :DWORD,:DWORD,:DWORD,:DWORD,:DWORD
WriteConsole equ <WriteConsoleW>
```

и не "париться" с переписыванием исходника.

Надо отметить, в MASM32 подобного "юникодного" инклюда нет, однако вы легко можете сделать его сами.

#6. И, наконец, третья, самая большая "халява" — это маленькая фенечка, использование которой сразу же превращает макроассемблер из *языка кодирования* в *язык программирования*!

С помощью этой "фенечки" целый блок инструкций:

```
push 0
push 0
push 16d
push offset sWriteText
push hStdout
call WriteConsoleA
```

мы с легкостью можем заменить одной-единственной строчкой:

```
invoke WriteConsoleA, hStdout, offset sWriteText, 16d, 0, 0
```

Обратите внимание, что при использовании этой команды параметры мы передаем слева направо, в той же очередности, что и вещает нам MSDN. В отличие от простыни "пушей" с "каллом" в конце.

#7. Теперь самый главный момент... Затаите дыхание!

В свете вышесказанного, вышерасписанного и вышерасжеванного наш исходник принимает весьма красивый "высокоуровневый" вид:

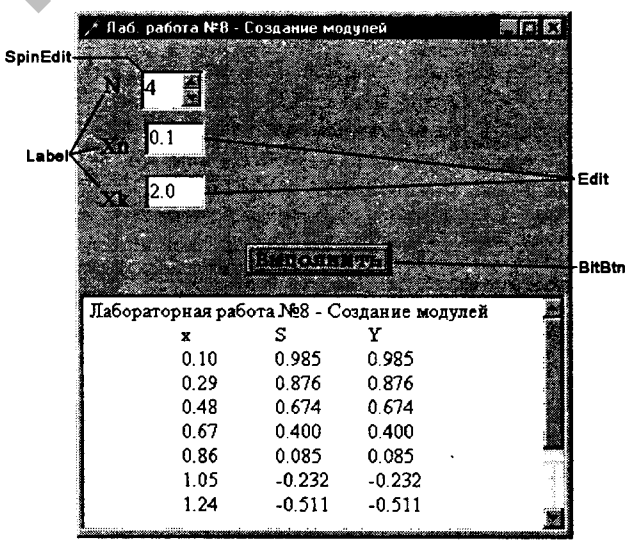
```
.386
.model flat,stdcall
option casemap:none

includelib kernel32.lib
include windows.inc
include kernel32.inc

.const

sConsoleTitle db 'My First Console Application',0
sWriteText db 'hEILo, Wo(R)LD!!'

.code

Main PROC
  LOCAL hStdout :DWORD

  invoke SetConsoleTitle, offset sConsoleTitle
  invoke GetStdHandle, STD_OUTPUT_HANDLE
  mov hStdout,EAX
  invoke WriteConsole, hStdout, offset sWriteText, 16d, NULL, NULL
  invoke Sleep, 2000d
  invoke ExitProcess, NULL
Main ENDP
end Main
```

А что? Самое время выпить бутылочку пива ;).

Программирование алгоритмов с использованием функций и процедур. Создание модулей

А.ШАКИРИН,
г.Минск, БАТУ, каф.ВТ.

Цель этой статьи — освоить методику создания модулей, содержащих процедуры и функции, и научиться использовать их в проекте.



1. Пример создания приложения

Необходимо создать Windows-приложение, которое выводит таблицу значений функции

$$Y(x) = \left(1 - \frac{x^2}{2}\right)\cos x - \frac{x}{2}\sin x$$

и ее разложения в ряд в виде суммы

$$S(x) = \sum_{n=0}^{n} (-1)^n \frac{2n^2+1}{(2n)!} x^{2n}$$

для значений x от x_n до x_k с шагом $h=(x_k - x_n)/10$. Для этого надо написать модуль, в котором вычисление значений $Y(x)$ следует оформить в виде функции, а вычисление $S(x)$ — в виде процедуры. Затем необходимо подключить модуль к проекту и выполнить созданное приложение.

Один из возможных вариантов панели интерфейса создаваемого приложения показан на **рисунке**.

1.1. Размещение компонентов на Форме

Разместим на Форме компоненты Label, Edit, SpinEdit, BitBtn и Memo.

Сохраним модуль под именем UnModul.

1.2. Создание модуля и подключение его к проекту

В соответствии с поставленной задачей создадим модуль, в котором вычисление значений $Y(x)$ оформим в виде функции, а вычисление $S(x)$ — в виде процедуры. Для создания модуля откроем в главном меню пункт File и щелкнем мышью на опции New...(Новый...). Delphi откроет панель New Items (Репозиторий), в которой сделаем активной пиктограмму **Unit** (Модуль) и нажмем кнопку OK. Откроется окно с "пустым" модулем Unit2. С помощью опции Save As... меню File сохраним модуль в папке с файлами проекта, присвоив ему, например, имя UnFuncProc.

В этом модуле операторы вычисления $Y(x)$ в виде подпрограммы-функции F и операторы вычисления $S(x)$ в виде подпрограммы-процедуры Sum оформим по правилам создания модулей.

Для подключения модуля UnFuncProc к проекту необходимо сделать активным окно с текстом модуля UnModul, затем

в меню File выбрать опцию Use Unit... и в открывшемся окне Use Unit указать имя используемого модуля — UnFuncProc. Убедитесь в том, что в разделе Implementation модуля UnModul появился оператор **Uses UnFuncProc;**, который Delphi вставила в текст модуля UnModul.

Откройте главный файл проекта и убедитесь в том, что проект не содержит посторонних модулей и файлов.

1.3. Текст модуля UnFuncProc

```
unit UnFuncProc;

 interface
 var
  n:integer;       // количество слагаемых в сумме S
 Function F(x:extended):extended;
 Procedure Sum(x:extended;Var s:extended);

 Implementation Function F(x:extended):extended;
  begin
    result:=(1-x*x*0.5)*cos(x)-0.5*x*sin(x);
  end;
 Procedure Sum(x:extended;Var s:extended);
  var
    c:extended;
    k:integer;
  begin
    c:=-x*x*0.5;
    S:=1;
   for k:=1 to n do
    begin
      s:=s+c*(2*k*k+1);
      c:=-c*x*x/((2*k+1)*(2*k+2));
    end;
  end;
end.
```

1.4. Текст модуля UnModul

```
Unit UnModul;

interface
uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls,
  Forms, Dialogs,StdCtrls, ExtCtrls, Spin, Buttons;

type
  TForm1 = class(TForm)
    Memo1: TMemo;
    Label1: TLabel;
    Label2: TLabel;
    Label3: TLabel;
    Edit1: TEdit;
    Edit2: TEdit;
    SpinEdit1: TSpinEdit;
    BitBtn1: TBitBtn;
    procedure FormCreate(Sender: TObject);
    procedure BitBtn1Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;
Type
 func=function(x:extended):extended; // функциональный тип
 proc=procedure(x:extended;Var s:extended); // процедурный тип
var
  Form1: TForm1;

implementation

uses UnFuncProc;        // Delphi подключает модуль UnFuncProc

{$R *.DFM}

procedure TForm1.FormCreate(Sender: TObject);
begin
  SpinEdit1.text:='4'; // начальное значение N
  Edit1.text:='0.1';   // начальное значение Xn
  Edit2.text:='2.0';   // начальное значение Xk
  Memo1.Clear;
  Memo1.Lines.Add('Лабораторная работа №8 - Создание модулей');
end;

{В процедуре Tablica вычисляется и выводится}
{таблица значений x, S(x) и Y(x)}
procedure Tablica(Sum:proc;F:func;n:integer;xn,xk,h:extended);
 var
  x,y,s:extended;
 begin
  Form1.Memo1.Lines.Add(#9+'x'+#9+'S'+#9+'Y');
                                        // заголовок таблицы
  x:=xn;
  repeat                                // цикл по x
   Sum(x,s);      // вызов процедуры Sum для вычисления S(x)
   y:=F(x);       // обращение к функции F для вычисления Y(x)
   Form1.Memo1.Lines.Add(#9+FloatToStrF(x,ffFixed,5,2)
                                        // вывод x
                +#9+FloatToStrF(s,ffFixed,6,3)
                                        // вывод S
                +#9+FloatToStrF(y,ffFixed,6,3));
                                        // вывод Y
   x:=x+h;
  until x>xk;
 end;
procedure TForm1.BitBtn1Click(Sender: TObject);
 var
  xn,xk,h:extended;
begin
 n:=StrToInt(SpinEdit1.Text); xn:=StrToFloat(Edit1.Text);
 xk:=StrToFloat(Edit2.Text); h:=(xk-xn)*0.1;          // шаг h
 Tablica(Sum,F,n,xn,xk,h);
          // вызов процедуры Tablica для вычисления таблицы
 end;
end.
```

2. Индивидуальные задания

В заданиях необходимо вывести на экран таблицу значений функции $Y(x)$ и ее разложения в ряд $S(x)$ для значений x от x_n до x_k с шагом $h=(x_k - x_n)/10$. Близость значений $S(x)$ и $Y(x)$ во всем диапазоне значений x указывает на правильность вычисления $S(x)$ и $Y(x)$.

№	x_n	x_k	S(x)	n	Y(x)
1	0,1	1	$1+\frac{\ln 3}{1!}x+\frac{\ln^2 3}{2!}x^2+\ldots+\frac{\ln^n 3}{n!}x^n$	8	3^x
2	$\frac{\pi}{5}$	$\frac{9\pi}{5}$	$\cos x+\frac{\cos 2x}{2}+\ldots+\frac{\cos nx}{n}$	18	$-\ln\left\|2\sin\frac{x}{2}\right\|$
3	$\frac{\pi}{5}$	$\frac{4\pi}{5}$	$\sin x-\frac{\sin 2x}{2}+\ldots+(-1)^{n+1}\frac{\sin nx}{n}$	6	$\frac{x}{2}$
4	0,1	0,8	$x\sin\frac{\pi}{4}+x^2\sin 2\frac{\pi}{4}+\cdots+x^n\sin n\frac{\pi}{4}$	12	$\frac{x\sin\frac{\pi}{4}}{1-2x\cos\frac{\pi}{4}+x^2}$
5	0,1	0,8	$x+\frac{x^5}{5}+\ldots+\frac{x^{4n+1}}{4n+1}$	16	$\frac{1}{4}\ln\frac{1+x}{1-x}+\frac{1}{2}arctg$
6	0,1	1	$1+\frac{\cos x}{1!}+\ldots+\frac{\cos nx}{n!}$	14	$e^{\cos x}\cos(\sin x)$

Выберите из таблицы два варианта индивидуальных заданий. Создайте модуль, в котором вычисление значений $S(x)$ реализуйте в виде подпрограммы-процедуры, а вычисление значений $Y(x)$ — в виде подпрограммы-функции. На панели интерфейса установите компонент, с помощью которого реализуйте возможность выбора соответствующего варианта задания и вывод таблицы значений x, $S_i(x)$, $Y_i(x)$, где i — номер варианта. Созданный модуль подключите к проекту и выполните приложение.

Литература

1. В.В.Феофанов. Delphi 3. Учебный курс. — М.: Нолидж, 2001.

2. Э.Возневич. Delphi. Освой самостоятельно. — М.: Восточная книжная компания, 1996.

3. Дж.Матчо, Д.Р.Фолкнер. Delphi. — М.: БИНОМ, 1995.

4. М.Канту. Delphi 2 для Windows 95/NT. — М.: ООО "Малип", 1997.