Программирование и программные среды

Коды, ассемблеры и языки высокого уровня

Алгоритм должен быть записан на языке, понятном процессору, т.е. **машинном языке** (*логические последовательности нулей и единиц*). Разные типы процессоров имеют разные наборы команд. Если язык программирования ориентирован на конкретный тип процессора и учитывает его особенности, то он называется языком программирования низкого уровня.

Ассемблер: операторы языка программирования близки к **машинному коду** и ориентированы на конкретные команды процессора.

Ассемблер

Представляет каждую команду машинного кода с помощью символьных условных обозначений (***мнемоник*).**

Преобразование одной машинной инструкции в одну команду ассемблера называется ***транслитерацией***.

Т.к. наборы инструкций для каждого модели процессора отличаются, конкретной компьютерной архитектуре соответствует свой язык ассемблера, и написанная на нем программа может быть использована только в этой среде.

- С помощью ассемблера создаются эффективные и компактные программы.
- Требуется хорошо знать устройство компьютера, его архитектуру (*методы адресации, типы данных и т.д.*).
- Затрудняется отладка больших приложений, а результирующая программа не может быть перенесена на компьютер с другим типом процессора.
- Применяют для написания небольших системных приложений: драйверов, модулей стыковки с нестандартным оборудованием.
- В машинной графике, на языке ассемблера пишутся библиотеки, эффективно реализующие алгоритмы обработки изображений, требующие интенсивных вычислений.

В качестве примера приведем программу на языке ассемблера для IBM PC. Программа вычисляет значение a = b + c для целых a, b и c:

```
.MODEL   SMALL     задает механизм распределения памяти под данные и команды
.DATA              определяет начало участка программы с данными
b        DW    5
c        DW   3
a        DW    ?
.CODE
begin   MOV   AX,@DATA   записывают адрес сегмента данных в
        MOV    DS,AX      регистр  данных DS
        MOV    AX,B
        ADD    AX,C
        MOV    A,AX
        MOV    AH,4CH     возврат управления ОС
        INT    21H        прерывание
 END    begin
```

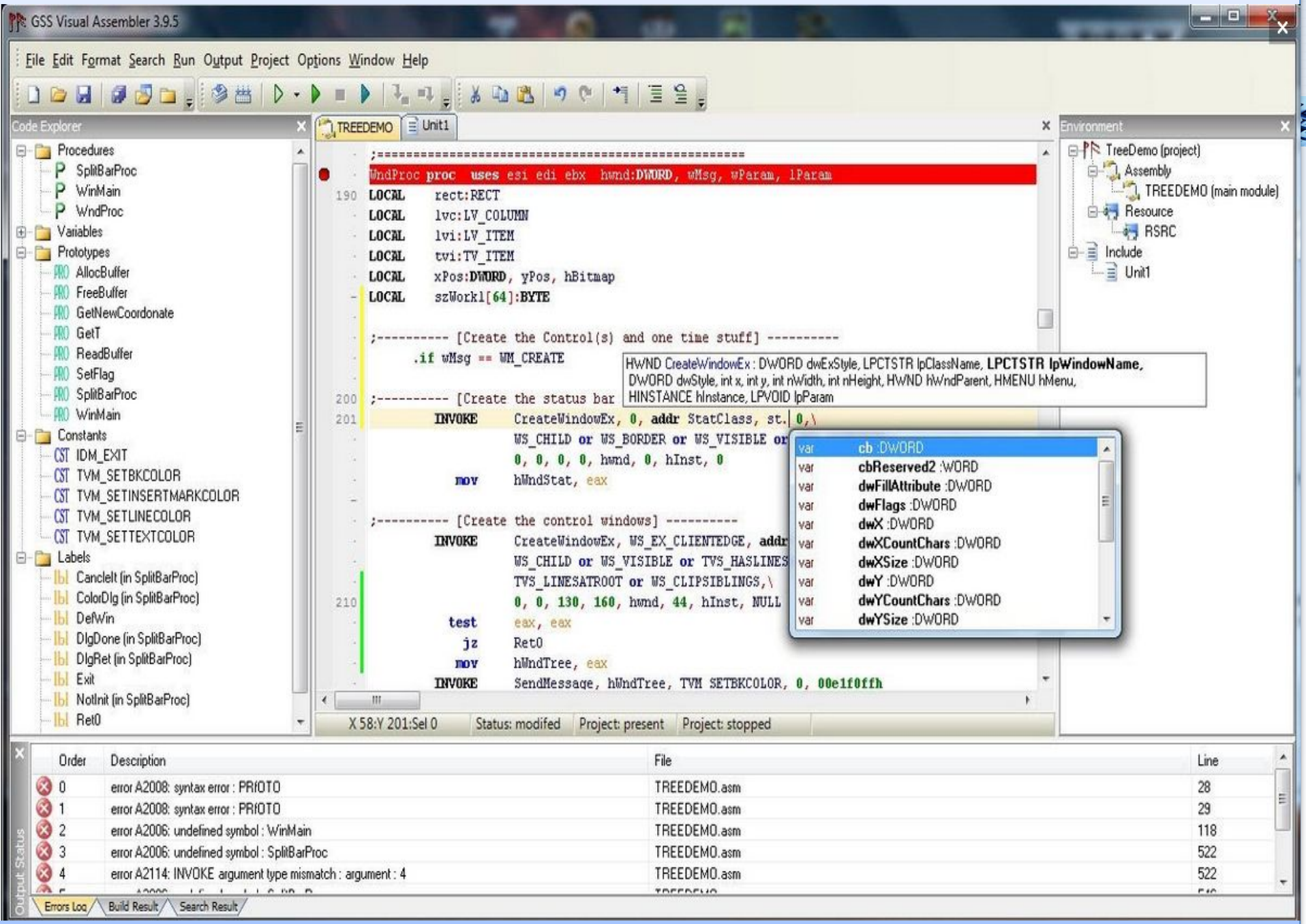
Emulator | Assembler

Intel 8080 (KP580BM80A) assembler

```
        ; i8080 assembler code
prompt  equ 0F86Ch
puts    equ 0F818h

        org 0

        lxi hl, msg
        call puts
        jmp prompt

msg:
        db 1fh,'radio-86rk snowa s nami!',0dh,0ah,0
```

```
                        ; i8080 assembler code
                prompt  equ 0F86Ch
                puts    equ 0F818h

                        org 0

0000    21 09 00        lxi hl, msg
0003    CD 18 F8        call puts
0006    C3 6C F8        jmp prompt

                msg:
0009    1F 72 61 64     db 1fh,'radio-86rk snowa s nami!',0dh,0ah,0
        . . .
0021    21 0D 0A 00

Labels:
 msg         0009  prompt      F86C  puts        F818

Memory dump:
0000:  21 09 00 CD 18 F8 C3 6C-F8 1F 72 61 64 69 6F 2D  !......l..radio-
0010:  38 36 72 6B 20 73 6E 6F-77 61 20 73 20 6E 61 6D  86rk.snowa.s.nam
0020:  69 21 0D 0A 00          -                       i!.............
```

Upload to emulator


GSS Visual Assembler 3.9.5
File Edit Format Search Run Output Project Options Window Help
Code Explorer
Procedures
SplitBarProc
WinMain
WndProc
Variables
Prototypes
AllocBuffer
FreeBuffer
GetNewCoordonate
GetT
ReadBuffer
SetFlag
SplitBarProc
WinMain
Constants
IDM_EXIT
TVM_SETBKCOLOR
TVM_SETINSERTMARKCOLOR
TVM_SETLINECOLOR
TVM_SETTEXTCOLOR
Labels
CancleIt (in SplitBarProc)
ColorDlg (in SplitBarProc)
DefWin
DlgDone (in SplitBarProc)
DlgRet (in SplitBarProc)
Exit
NotInit (in SplitBarProc)
Ret0
TREEDEMO
Unit1
;==
WndProc proc uses esi edi ebx hwnd:DWORD, wMsg, wParam, lParam
LOCAL rect:RECT
LOCAL lvc:LV_COLUMN
LOCAL lvi:LV_ITEM
LOCAL tvi:TV_ITEM
LOCAL xPos:DWORD, yPos, hBitmap
LOCAL szWork1[64]:BYTE
;---------- [Create the Control(s) and one time stuff] ----------
.if wMsg == WM_CREATE
;---------- [Create the status bar
INVOKE CreateWindowEx, 0, addr StatClass, st.
HWND CreateWindowEx : DWORD dwExStyle, LPCTSTR lpClassName, LPCTSTR lpWindowName,
DWORD dwStyle, int x, int y, int nWidth, int nHeight, HWND hWndParent, HMENU hMenu,
HINSTANCE hInstance, LPVOID lpParam
WS_CHILD or WS_BORDER or WS_VISIBLE or
0, 0, 0, 0, hwnd, 0, hInst, 0
mov hWndStat, eax
;---------- [Create the control windows] ----------
INVOKE CreateWindowEx, WS_EX_CLIENTEDGE, addr
WS_CHILD or WS_VISIBLE or TVS_HASLINES
TVS_LINESATROOT or WS_CLIPSIBLINGS,\
0, 0, 130, 160, hwnd, 44, hInst, NULL
test eax, eax
jz Ret0
mov hWndTree, eax
INVOKE SendMessage, hWndTree, TVM_SETBKCOLOR, 0, 00e1f0ffh
var cb :DWORD
var cbReserved2 :WORD
var dwFillAttribute :DWORD
var dwFlags :DWORD
var dwX :DWORD
var dwXCountChars :DWORD
var dwXSize :DWORD
var dwY :DWORD
var dwYCountChars :DWORD
var dwYSize :DWORD
X 58:Y 201:Sel 0
Status: modifed
Project: present
Project: stopped
Environment
TreeDemo (project)
Assembly
TREEDEMO (main module)
Resource
RSRC
Include
Unit1
Order
Description
File
Line
0 error A2008: syntax error : PRfOTO TREEDEMO.asm 28
1 error A2008: syntax error : PRfOTO TREEDEMO.asm 29
2 error A2006: undefined symbol : WinMain TREEDEMO.asm 118
3 error A2006: undefined symbol : SplitBarProc TREEDEMO.asm 522
4 error A2114: INVOKE argument type mismatch : argument : 4 TREEDEMO.asm 522
Output Status
Errors Log
Build Result
Search Result

Преимущества алгоритмических языков

- алфавит АЯ шире алфавита машинного языка, что повышает наглядность текста программы;
- набор операций не зависит от набора машинных операций, а выбирается из соображений удобства формулирования алгоритмов решения задач определенного класса;
- формат предложений достаточно гибок и удобен для использования, т.е. можно задать достаточно содержательный этап обработки данных;
- операции задаются с помощью общепринятых математических обозначений;
- данным в АЯ присваиваются индивидуальные имена;
- в языке может быть предусмотрен значительно более широкий набор типов данных по сравнению с набором машинных типов данных.

Языки программирования высокого уровня

- Язык Algol не дошел до нашего времени, но его идеи были перенесены в более современные языки программирования.
- Язык Cobol, разработан для коммерческих расчетов; существует до сих пор и на нем разрабатываются приложения, связанные с бизнесом.
- Язык FORTRAN, разрабатывался для сложных математических расчетов. И в настоящий момент этот язык не уступает пальмы первенства в этой области программирования, оставаясь самым удобным языком для расчетов в науке и технике, сложной графики и т.п.

Языки программирования высокого уровня

Basic, изначально был ориентирован на тех, кто делает первые шаги в программировании. Но для серьезных задач существует транслятор Visual Basic компании Microsoft, который фактически стал стандартом в разработке бизнес-решений и информационных систем. Ранее Basic был аппаратно встроен в базовое программное обеспечение IBM PC. Даже в комплект поставки операционной системы MS-DOS входит разновидность Basic — QBasic.

Языки программирования высокого уровня

Pascal, разработан швейцарским ученым, специалистом в области информатики, Никлаусом Виртом. Паскаль обеспечивает возможность создания больших программ, поддерживая их строгую логическую структуру. Паскаль считается важнейшим инструментом для обучения методам структурного программирования и с 1983 года введен во всех средних школах США в учебные курсы для учащихся, которые специализируются в области информатики

Языки программирования высокого уровня

C, 1972 г., Деннис Ричи специалист по системному программированию. Был использован для программирования новой операционной системы UNIX. Язык среднего уровня, в котором удобство, краткость и мобильность ЯВУ сочетаются с возможностью непосредственного доступа к аппаратуре, присущим ассемблерам.

C++, 1983г, Рик Масситти. Дает возможность структурировать большие программы, эффективно работать с аппаратными средствами, создавать элегантные и надежные интерфейсы.

Языки программирования высокого уровня

C, 1972 г., Деннис Ричи специалист по системному программированию. Был использован для программирования новой операционной системы UNIX. Язык среднего уровня, в котором удобство, краткость и мобильность ЯВУ сочетаются с возможностью непосредственного доступа к аппаратуре, присущим ассемблерам.

C++, 1983г, Рик Масситти. Дает возможность структурировать большие программы, эффективно работать с аппаратными средствами, создавать элегантные и надежные интерфейсы.

Понятие о системе программирования

Система программирования – это комплекс средств, предназначенный для создания и эксплуатации программ на конкретном языке программирования на ЭВМ определенного типа.

- **Средства для создания программ**. Традиционными средствами разработки программ являются ***алгоритмические (процедурные) языки программирования***. Для создания программы на выбранном языке программирования нужно иметь следующие компоненты:
- **Текстовый редактор** – это редактор, который позволяет набрать текст программы на языке программирования. Для этой цели можно использовать любые текстовые редакторы, но лучше пользоваться специализированным текстовым редактором.

Транслятор – это основа систем программирования. Трансляторы языков программирования, т.е. программы, обеспечивающие перевод исходного текста программы на машинный язык (объектный код), бывают двух типов: интерпретаторы и компиляторы.

Редактор связей (сборщик) – это программа, которая объединяет объектные модули отдельных частей программы и добавляет к ним стандартные модули подпрограмм стандартных функций (файлы с расширением .lib), которые содержатся в библиотеках, поставляемых вместе с компилятором, в единую программу, готовую к исполнению, т.е. создает исполнимый .exe файл. Этот файл имеет самостоятельное значение и может работать под управлением той (или такой же) операционной системы, в которой он создан.

Интегрированные системы программирования. В стандартную поставку, как правило, входят специализированный текстовый редактор, компилятор, редактор связей (сборщик), библиотеки стандартных функций, отладчик.

Системы визуального программирования — RAD-среды (Rapid Application Development), которые, не исключая возможности записи программы вручную, позволяют создавать текст программы автоматически, путем манипуляций со стандартными элементами управления, включенными в RAD-среду. Поэтому для RAD-среды понятие «программирование» часто заменяют понятием «проектирование».

В проектируемое окно готовые визуальные компоненты перетаскиваются с помощью мыши, затем свойства и поведение компонентов настраивается с помощью редактора. Исходный же текст программы, ответственный за работу этих элементов, генерируется автоматически с помощью среды быстрого проектирования, которая называется RAD-средой. Подобный подход называется ***визуальным программированием***.

Наиболее популярные языки программирования	*Соответствующие им визуальные среды быстрого проектирования программ для Windows*
Бейсик (Basic) – для освоения требует начальной подготовки (общеобразовательная школа)	Microsoft Visual Basic
Паскаль (Pascal) – требует специальной подготовки	Borland Delphi
Си++ (C++)- требует серьезной подготовки	Microsoft Visual C++
Ява (Java) требует серьезной подготовки	Java: Borland JBuilder

Компиляторы

Компилятор – это транслятор, который исходный текст программы переводит в машинный код. Если в тексте программы нет синтаксических ошибок, то машинный код будет создан. Но это, как правило, не работоспособный код, т.к. в этой программе не хватает подпрограмм стандартных функций, поэтому компилятор выдает промежуточный код, который называется ***объектным кодом*** и имеет расширение .obj.

Компилятор преобразует исходный код программы как одно целое, после чего сохраняет полный эквивалент исходного кода в виде исполняемого файла (в случае с ДОС, это com или exe файл).

Примеры:
1)Компилируемые Бейсики (Turbo Basic,...)
2) Паскаль, Си и многие другие ЯВУ
3) Ассемблер
Важно! **Компилятор**, в отличие от интерпретатора не выполняет программу, он ее создает, точнее, он создает исполняемый файл (в машинном коде) этой программы.

Фазы компиляции:

- **Лексический анализ** (разбор исходной прогаммы и выделении некоторых "крупных" единиц, лексем -) ключевые слова, идентификаторы, …)
- **Синтаксический анализ** (разбирает результат лексического анализа в соответствии с грамматикой)
- **Видозависимый анализ** (семантический, проверяет правильность типов данных)
- **Оптимизация кода** (преобразует промежуточное представление программы в целях повышения эффективности объектной программы (по скорости или объему памяти))
- **Генерация кода** (генерируется объектная программа)

Интерпретатор

Последовательно преобразует в машинный код каждую строку исходного кода программы. Интерпретатор сохраняет текущий результат не в виде выполняемого файла, а только в оперативной памяти компьютера. Интерпретаторы используются для разработки Web-страниц, такими как JavaScript. Поскольку Web-страницы просматриваются с помощью компьютеров различного типа, вы не сможете скомпилировать программу (заранее не известен тип компилятора). Вместо этого используется интерпретатор.

Интерпретатор

Примеры:

1) **Command.com** - интерпретатор командных файлов ДОС

2) **Интерпретируемые Бейсики** используемые в бытовых компьютерах ("Микроша", "БК","Спектрум") и ранних PC

3) **Perl** — высокоуровневый интерпретируемый динамический язык программирования общего назначения, созданный Ларри Уоллом, лингвистом по образованию. *Practical Extraction and Report Language* — «практический язык для извлечения данных и составления отчётов». С помощью Perl можно создать скрипт, который открывает один или несколько файлов, обрабатывает информацию и записывает результаты.

Бейсик

Язык Бейсик (BASIC — Beginners All-purpose Symbolic Instruction Code — универсальный символьный код для начинающих) был создан в 1965 г. Дж. Кемени и Т. Курцем как язык, облегчающий написание простых программ.

Первая программа 1964 год:

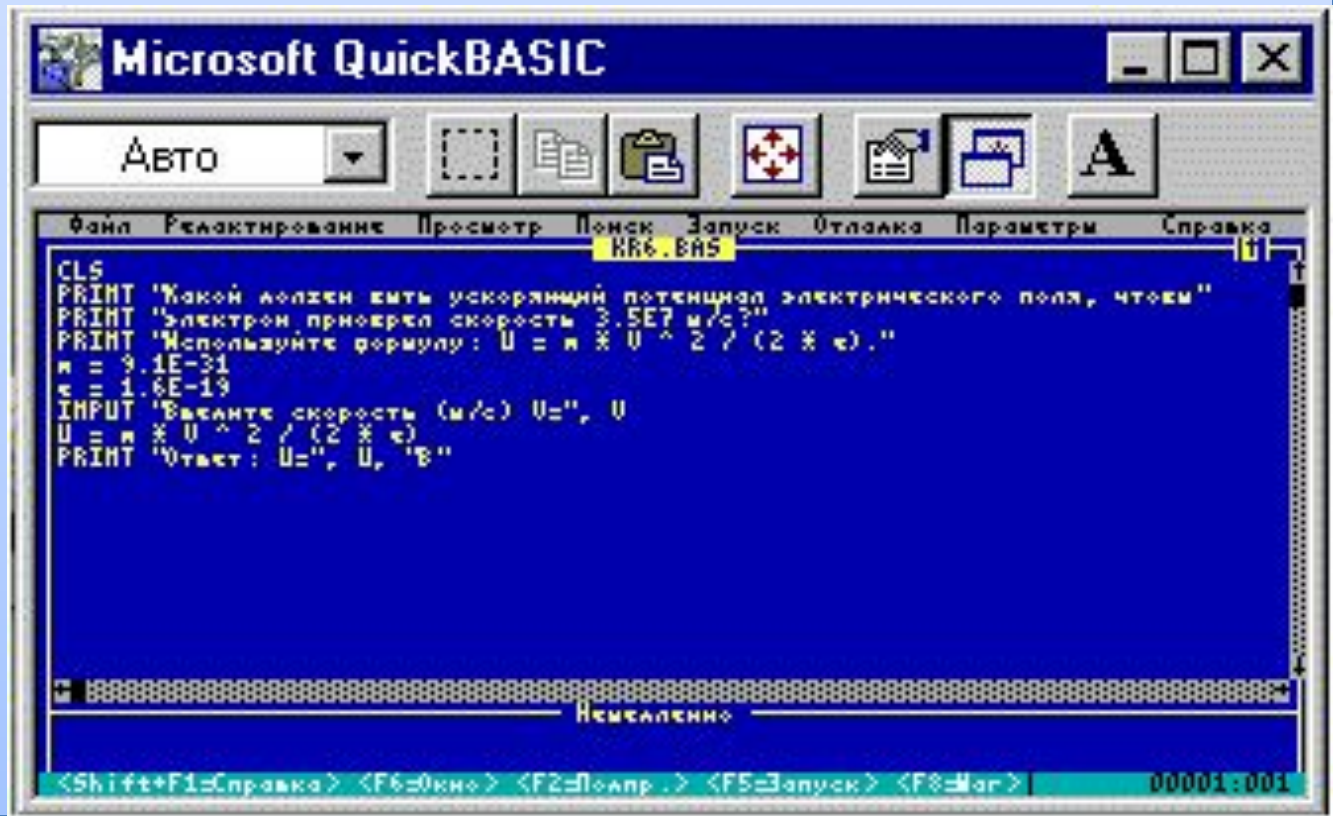
```
10 LET X=(7+8)/3
20 PRINT X
30 END
```

Сегодня: X=(7+8)/3PRINT X

Сейчас все чаще пишут Basic вместо BASIC, придавая другую трактовку названию: Basic — основной, базовый

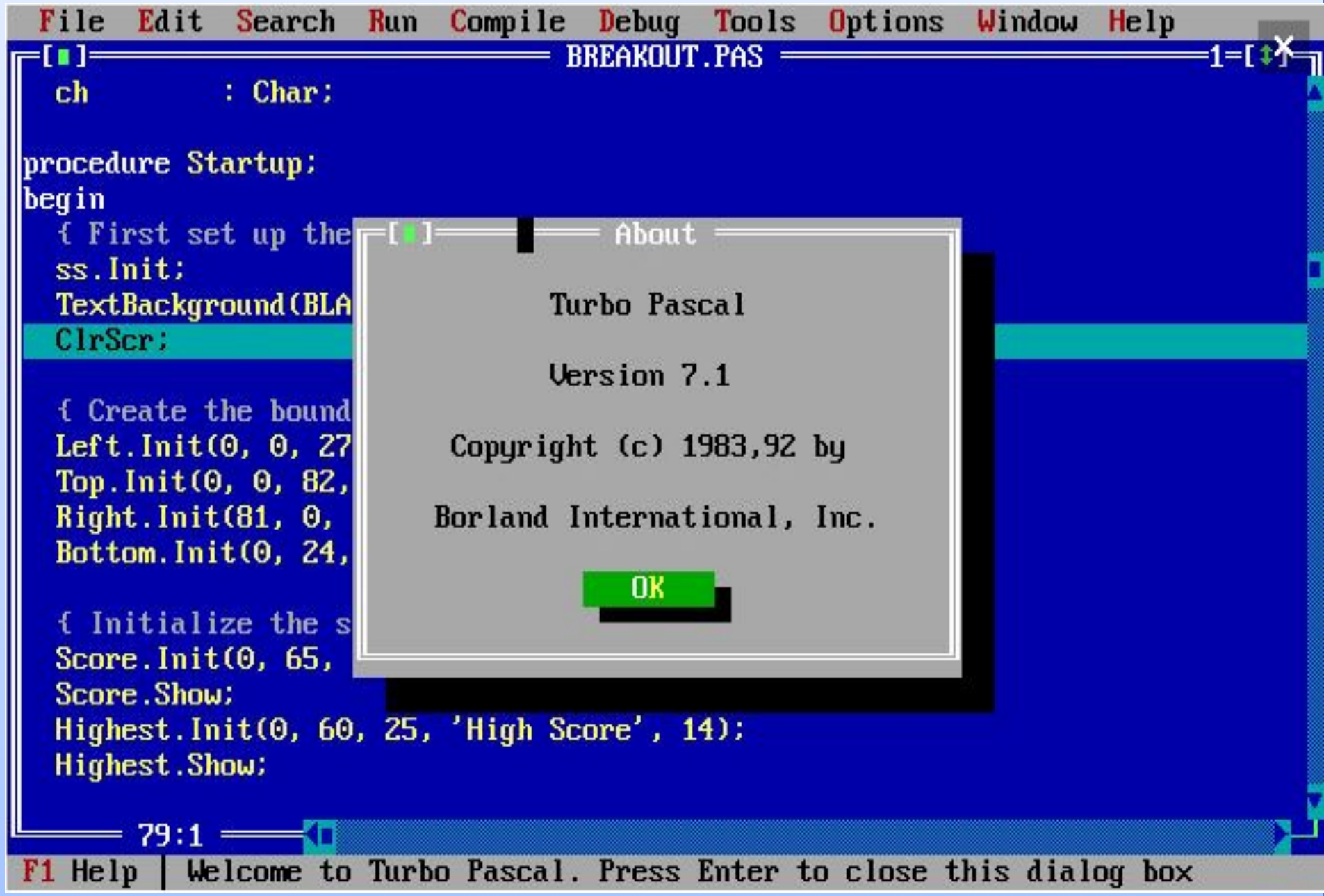
Программа на Бейсике

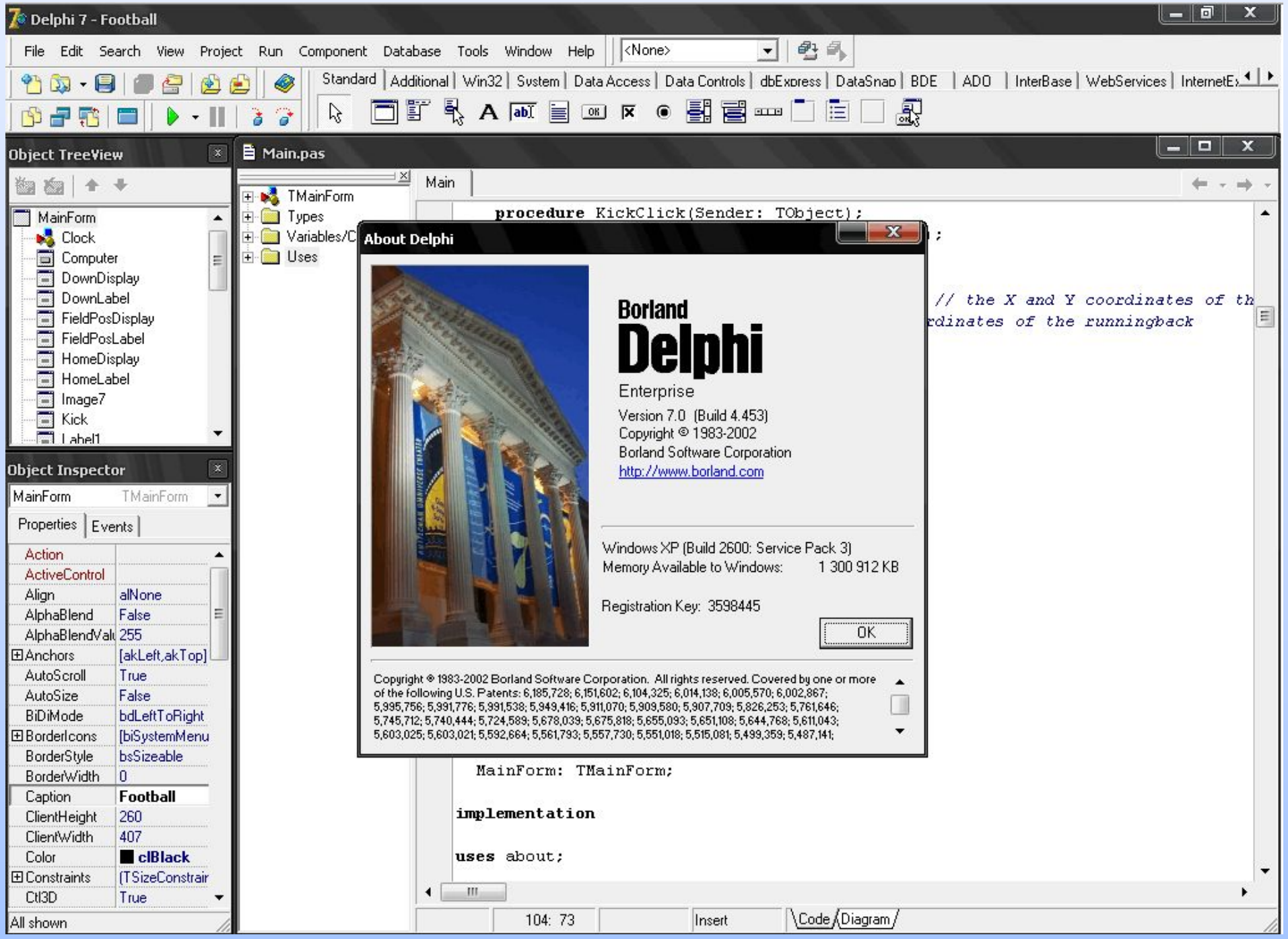
```
INPUT "N = "; N: DIM A(N)
FOR I = 1 TO N
   PRINT "A("; I; ") =";
   INPUT A(I)
NEXT I
S = 0
FOR I = 1 TO N
   S = S + A(I)
NEXT I
PRINT "Сумма ="; S
END
```

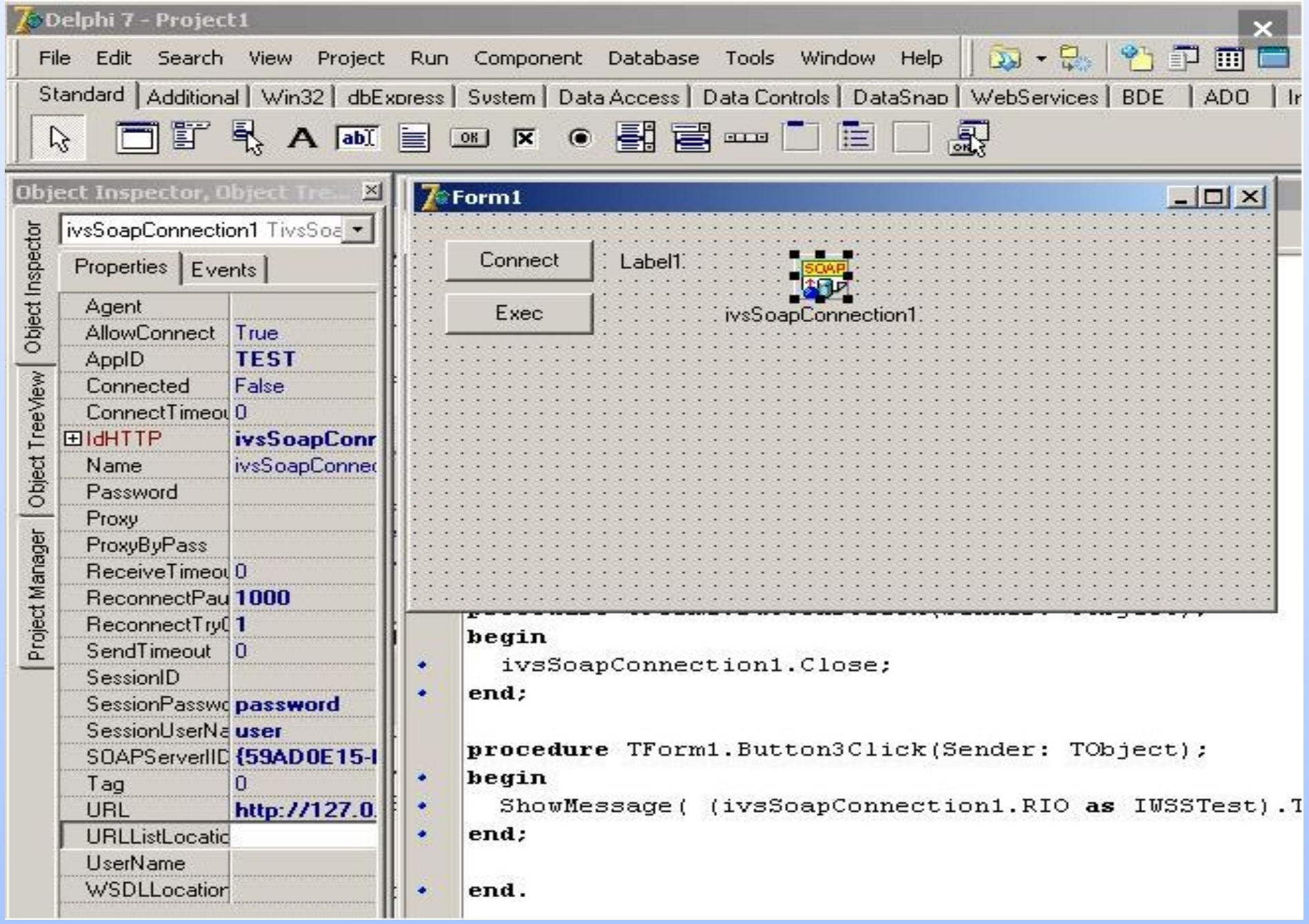


Программа на Паскале

```
Program Summa;
Type Mas = Array [1 .. 100] of Real;
Var A   : Mas;
       i, n: Integer;
       S   : Real;
BEGIN
Write('n = ');   ReadLn(n);
For  i:=1 to n do
Begin
     Write('A[', i, '] = ');
     ReadLn(A[i]);
end;
S : = 0;
For i : = 1 to n do
     S : = S + A[i];
     WriteLn('S = ', S:8:2);
END.
```

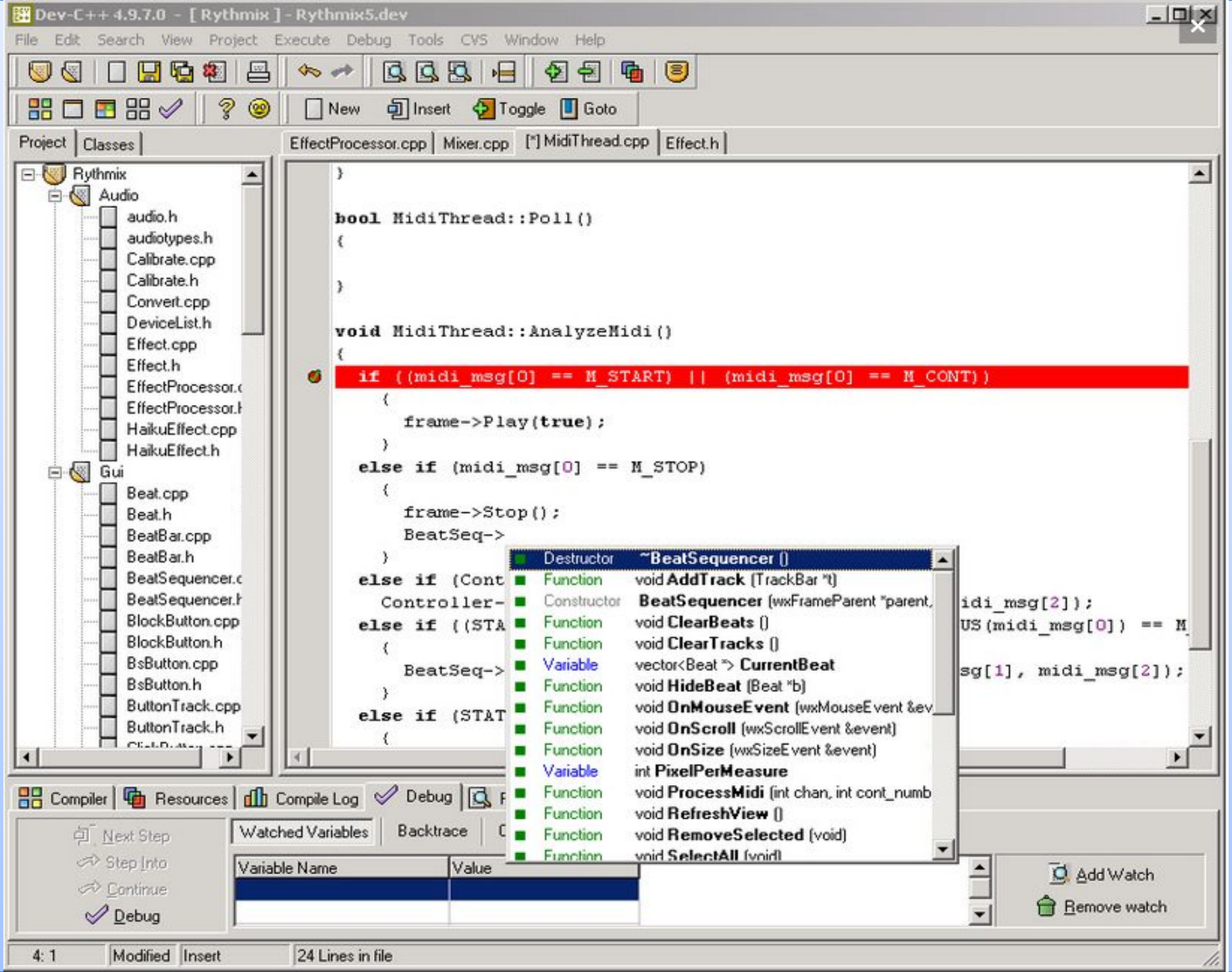

File Edit Search Run Compile Debug Tools Options Window Help
BREAKOUT.PAS
ch : Char;
procedure Startup;
begin
{ First set up the
ss.Init;
TextBackground(BLA
ClrScr;
{ Create the bound
Left.Init(0, 0, 27
Top.Init(0, 0, 82,
Right.Init(81, 0,
Bottom.Init(0, 24,
{ Initialize the s
Score.Init(0, 65,
Score.Show;
Highest.Init(0, 60, 25, 'High Score', 14);
Highest.Show;
About
Turbo Pascal
Version 7.1
Copyright (c) 1983,92 by
Borland International, Inc.
OK
79:1
F1 Help | Welcome to Turbo Pascal. Press Enter to close this dialog box


Delphi 7 - Football
File Edit Search View Project Run Component Database Tools Window Help
<None>
Standard Additional Win32 System Data Access Data Controls dbExpress DataSnap BDE ADO InterBase WebServices
Object TreeView
MainForm
Clock
Computer
DownDisplay
DownLabel
FieldPosDisplay
FieldPosLabel
HomeDisplay
HomeLabel
Image7
Kick
Label1
Object Inspector
MainForm TMainForm
Properties Events
Action
ActiveControl
Align alNone
AlphaBlend False
AlphaBlendVal 255
Anchors [akLeft,akTop]
AutoScroll True
AutoSize False
BiDiMode bdLeftToRight
BorderIcons [biSystemMenu
BorderStyle bsSizeable
BorderWidth 0
Caption Football
ClientHeight 260
ClientWidth 407
Color clBlack
Constraints (TSizeConstrair
Ctl3D True
All shown
Main.pas
TMainForm
Types
Variables/C
Uses
Main
procedure KickClick(Sender: TObject);
// the X and Y coordinates of th
rdinates of the runningback
MainForm: TMainForm;
implementation
uses about;
104: 73
Insert
Code Diagram
About Delphi
Borland
Delphi
Enterprise
Version 7.0 (Build 4.453)
Copyright © 1983-2002
Borland Software Corporation
http://www.borland.com
Windows XP (Build 2600: Service Pack 3)
Memory Available to Windows: 1 300 912 KB
Registration Key: 3598445
OK
Copyright © 1983-2002 Borland Software Corporation. All rights reserved. Covered by one or more of the following U.S. Patents: 6,185,728; 6,151,602; 6,104,325; 6,014,138; 6,005,570; 6,002,867; 5,995,756; 5,991,776; 5,991,538; 5,949,416; 5,911,070; 5,909,580; 5,907,709; 5,826,253; 5,761,646; 5,745,712; 5,740,444; 5,724,589; 5,678,039; 5,675,818; 5,655,093; 5,651,108; 5,644,768; 5,611,043; 5,603,025; 5,603,021; 5,592,664; 5,561,793; 5,557,730; 5,551,018; 5,515,081; 5,499,359; 5,487,141;


Delphi 7 - Project1
File Edit Search View Project Run Component Database Tools Window Help
Standard Additional Win32 dbExpress System Data Access Data Controls DataSnap WebServices BDE ADO
Object Inspector
ivsSoapConnection1 TivsSoa
Properties Events
Agent
AllowConnect True
AppID TEST
Connected False
ConnectTimeou 0
IdHTTP ivsSoapConr
Name ivsSoapConnec
Password
Proxy
ProxyByPass
ReceiveTimeou 0
ReconnectPau 1000
ReconnectTryC 1
SendTimeout 0
SessionID
SessionPasswc password
SessionUserNa user
SOAPServerIID {59AD0E15-I
Tag 0
URL http://127.0.
URLListLocatic
UserName
WSDLLocatior
Object Inspector
Object TreeView
Project Manager
Form1
Connect
Label1
Exec
SOAP
ivsSoapConnection1
begin
ivsSoapConnection1.Close;
end;
procedure TForm1.Button3Click(Sender: TObject);
begin
ShowMessage((ivsSoapConnection1.RIO as IWSSTest).T
end;
end.

Программа на СИ

```
# include <stdio.h>
# include <conio.h>
main()
{
float a[100], s; int i, n;
clrscr(); printf("n=");
scanf("%i", &n);
for (i = 1; i <= n; i++) {
    printf("a[%i]=", i);
    scanf("%f", &a[i]);}
s=0;
for (i = 1; i <= n; i++)
    s = s + a[i];
printf("s = % f \ n", s);
return 0;
}
```


Dev-C++ 4.9.7.0 - [Rythmix] - Rythmix5.dev
File Edit Search View Project Execute Debug Tools CVS Window Help
New Insert Toggle Goto
Project Classes
EffectProcessor.cpp Mixer.cpp [*] MidiThread.cpp Effect.h
Rythmix
Audio
audio.h
audiotypes.h
Calibrate.cpp
Calibrate.h
Convert.cpp
DeviceList.h
Effect.cpp
Effect.h
HaikuEffect.cpp
HaikuEffect.h
Gui
Beat.cpp
Beat.h
BeatBar.cpp
BeatBar.h
BlockButton.cpp
BlockButton.h
BsButton.cpp
BsButton.h
ButtonTrack.cpp
ButtonTrack.h
bool MidiThread::Poll()
void MidiThread::AnalyzeMidi()
if ((midi_msg[0] == M_START) || (midi_msg[0] == M_CONT))
frame->Play(true);
else if (midi_msg[0] == M_STOP)
frame->Stop();
BeatSeq->
Destructor ~BeatSequencer ()
Function void AddTrack (TrackBar *t)
Constructor BeatSequencer (wxFrameParent *parent,
Function void ClearBeats ()
Function void ClearTracks ()
Variable vector<Beat *> CurrentBeat
Function void HideBeat (Beat *b)
Function void OnMouseEvent (wxMouseEvent &ev
Function void OnScroll (wxScrollEvent &event)
Function void OnSize (wxSizeEvent &event)
Variable int PixelPerMeasure
Function void ProcessMidi (int chan, int cont_numb
Function void RefreshView ()
Function void RemoveSelected (void)
Function void SelectAll (void)
Compiler Resources Compile Log Debug
Next Step
Step Into
Continue
Debug
Watched Variables Backtrace
Variable Name Value
Add Watch
Remove watch
4: 1 Modified Insert 24 Lines in file

Данные как объект обработки

Реальные данные, с которыми работает программа, — это **числа, строки и логические величины** («истина» и «ложь»). Эти типы данных называют **базовыми**.

Строки - это набор любых символов, заключенных в кавычки (Бейсик) или апострофы (Паскаль).

Имена или **идентификаторы**, присваиваются всем объектам программы: состоит из букв, цифр, знака подчеркивания и некоторых спецсимволов, первым символом должна быть буква.

Константа - это постоянная величина. В ходе выполнения программы константы не меняются.

Переменная - поименованная величина, которая может менять свое значение в ходе выполнения программы.

Данные как объект обработки

Выражение - несколько констант, переменных (и значений функций), объединенные знаками операций.

Структура программы

Наименьшей структурной единицей программы является **оператор.** Операторы размещаются в строках по одному или по несколько. По большому счету написание программы делится на два этапа:

1. Проектирование программы, составление ее алгоритма;
2. Запись этого алгоритма с помощью операторов конкретного языка программирования (**кодинг** (coding) или кодирование).

Данные как объект обработки

Массивы. Массивом называется упорядоченная совокупность данных одного типа, доступ к элементу массива осуществляется по его индексу. В памяти компьютера для массива выделяется единое поле. Например, массив А:

A_1	A_2	A_3	A_4	A_5

Многомерные массивы, в частности — двумерные. Для индексации элементов двумерного массива указываются два индекса — сначала номер строки, затем номер столбца.

$X_{1,1}$	$X_{1,2}$	$X_{1,3}$	$X_{1,4}$
$X_{2,1}$	$X_{2,2}$	$X_{2,3}$	$X_{2,4}$
$X_{3,1}$	$X_{3,2}$	$X_{3,3}$	$X_{3,4}$

В памяти ЭВМ и этот массив хранится в виде одномерной последовательности элементов сначала первой строки, потом второй и затем третьей.

Массивы

Описание массивов:

```
Var   A : array[1..30] of byte;
       S : array[1..30] of string; {или}
      SO: array[1..30] of string[12];
```

Присвоение значений элементам массива:

```
A[1]:= 5;   A[2]:= 4;                    и т. д.
S[1]:= 'Иванов'; S[2]:= 'Петров';     и т. д.
```

```
Randomize;
  for i:= 1 to 100 do R[i]:= - 30 + random(71);
```

y=sin(x), где x= Pi * i/180, 0<= i <=180.

```
for i:= 0 to 180 do y[i]:= sin(Pi * i/180);
```

Подпрограммы: процедуры и функции

"Принцип решения сложных задач"

Функция возвращает результат и может использоваться в выражениях (**функции пользователя**).

```
var x,y,m,n: integer;

procedure MaxNumber(a,b: integer;
                    var max: integer);
begin
  if a>b then max:=a  else max:=b;
end;

begin
  write('Введите x,y ');
  readln(x,y);
  MaxNumber(x,y,m);
  MaxNumber(2,x+y,n);
  writeln('m=',m,'n=',n);
end.
```

```
var x,y,m,n: integer;

function MaxNumber(a,b: integer): integer;
  var max: integer;
begin
  if a>b then max:=a  else max:=b;
  MaxNumber := max;
end;

begin
  write('Введите x,y ');
  readln(x,y);
  m := MaxNumber(x,y);
  n := MaxNumber(2,x+y);
  writeln('m=',m,'n=',n);
end.
```

Программа на СИ

```
# include <stdio.h>
# include <conio.h>
main()
{
float a[100], s; int i, n;
clrscr(); printf("n=");
scanf("%i", &n);
for (i = 1; i <= n; i++) {
    printf("a[%i]=", i);
    scanf("%f", &a[i]);}
s=0;
for (i = 1; i <= n; i++)
    s = s + a[i];
printf("s = % f \ n", s);
return 0;
}
```