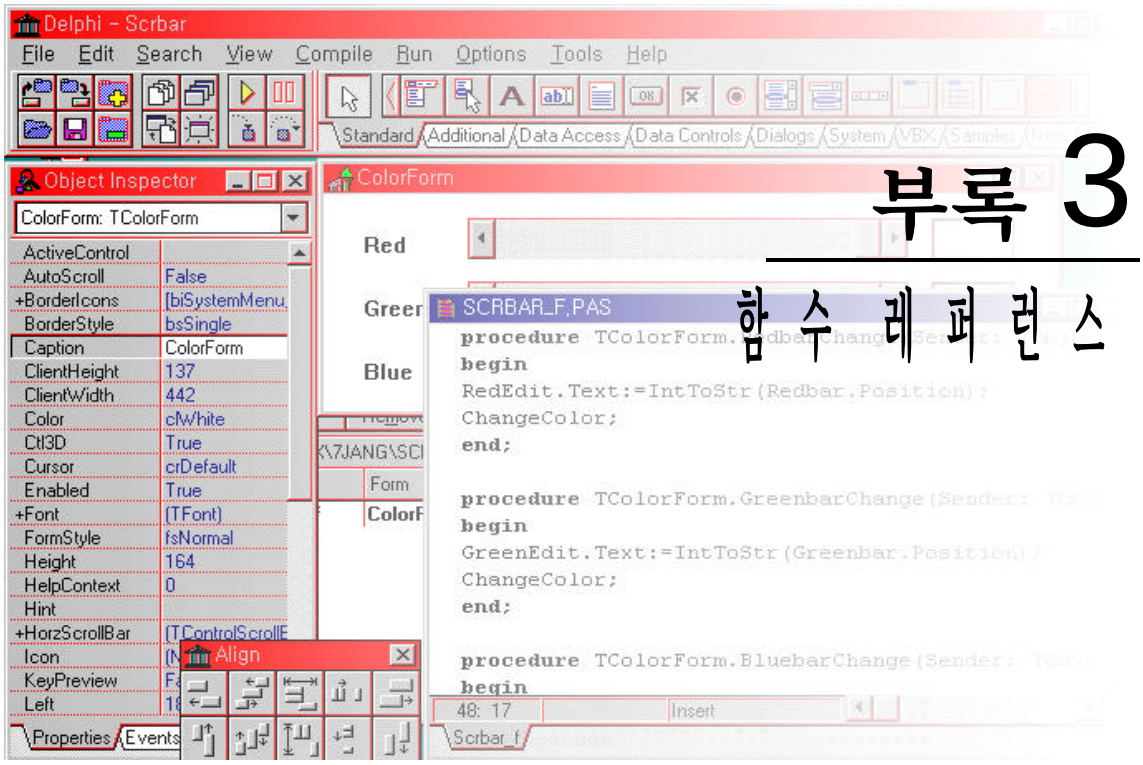


부록 3

함수 레퍼런스

델파이의 유닛에 포함되어 있는 개별 함수들에 관해 정리하였다. 함수의 수가 워낙 많아 전부 다 정리하지는 못하였으며 내부적으로 사용되는 함수, 호환성을 위해 존재하는 함수, 동일 함수가 존재하는 함수 등은 레퍼런스에서 제외하였다. 그러나 기본적으로 사용되는 함수들에 관해서는 상세한 예를 들어 설명하고 있으므로 부족함이 없을 것이다. 함수 게재 순서는 알파벳순이며 함수명 우측에 함수가 정의되어 있는 유닛을 명시하였으므로 이 레퍼런스에서 부족한 면은 유닛을 직접 참조하기 바란다.

Abs — system Unit

문법 : function Abs(X);

X의 절대값을 구한다. 절대값이란 값은 같으며 부호가 양수인 수를 말한다. X는 정수 또는 실수값이다.

Abs(-3.14); { 결과는 3.14 }
Abs(3.14); { 결과는 3.14 }

AddExitProc — system Unit

문법 : procedure AddExitProc(Proc:TRpocedure);

종료 프로시저를 등록한다. 종료 프로시저란 런타임 라이브러리의 exit 프로시저에 등록되는 함수이며 프로그램이 끝날 때 자동으로 호출되는 함수를 말한다. 프로그램을 끝내기 전에 해야할 특별한 일, 예를 들어 메모리 해제나 자원 해제 등의 일이 있다면 종료 프로시저에 코드를 작성하도록 한다. 프로그램이 종료될 때 AddExitProc로 등록된 역순으로 호출된다. 즉 처음 등록된 함수가 제일 끝에 호출된다.

Addr — System Unit

문법 : function Addr(X):Pointer;

X의 번지를 구한다. X는 변수나 함수의 명칭이며 결과는 X의 메모리 번지를 나타내는 포인터 값이며 Pointer형이다. Addr 함수의 리턴값이 Pointer형이므로 Pointer형의 변수에 곧바로 대입될 수 있지만 이 함수의 리턴값으로 메모리를 직접

참조할 수는 없다. 메모리를 참조하기 위해서는 타입 캐스트를 해 주어야 한다.

```
var
 i:Integer;
begin
 P := Addr(i);  { i의 번지를 구한다. }
end.
```

AllocMem SysUtils Unit

문법 : function AllocMem(Size;Cardinal):Pointer;

인수로 주어진 Size 바이트만큼 힙에서 메모리를 할당하며 할당된 메모리는 모두 0으로 초기화된다. 할당한 메모리를 해제하려면 FreeMem 함수를 사용한다.

AnsiCompareStr SysUtils Unit

문법 : function AnsiCompareStr(const S1,S2:string):Integer;

두 개의 문자열 S1과 S2를 비교하되 대소문자를 구분하여 비교한다. S1과 S2 문자열이 같으면 0을 리턴하며 S1<S2이면 음수의 값을 리턴하며 S1>S2이면 양수를 리턴한다. 문자열끼리의 비교는 문자열을 이루는 각 문자의 문자 코드를 비교하여 수행된다.

AnsiCompareText SysUtils Unit

문법 : function AnsiCompareStr(const S1,S2:string):Integer;

두 개의 문자열 S1과 S2를 비교하되 대소문자를 구분하지 않고 비교한다. S1과 S2문자열이 같으면 0을 리턴하며 S1<S2이면 음수의 값을 리턴하며 S1>S2이면 양수를 리턴한다. 문자열끼리의 비교는 문자열을 이루는 각 문자의 문자 코드를 비교하여 수행된다.

AnsiLowerCase SysUtils Unit

문법 : function AnsiLowerCase(const S: string): string;

문자열 내부의 대문자를 모두 소문자로 바꾼다. 영문자 이외의 글자는 영향을 받지 않는다. 문자 변환은 현재 설치된 언어 드라이버의 영향을 받는다.

AnsiUpperCase SysUtils Unit

문법 : function AnsiUpperCase(const S: string): string;

문자열 내부의 소문자를 모두 대문자로 바꾼다. 영문자 이외의 글자는 영향을 받지 않는다. 문자 변환은 현재 설치된 언어 드라이버의 영향을 받는다.

Append System Unit

문법 : procedure Append(var f:Text);

파일을 추가 기록용으로 오픈한다. 파일 변수 F는 Assign으로 외부 파일과 연결되어 있어야 한다. 외부 파일이 없으면 에러가 발생하며 이미 열려 있는 경우는 파일을 닫은 후 다시 연다. 파일을 오픈한 후 FP는 파일 끝(EOF)에 위치시키므로 추가적인 데이터를 파일 끝에 추가할 수 있다. 다음 예는 루트 디렉토리의 AUTOEXEC.BAT 파일 끝에 PROMPT문을 추가한다.

```
var
 F: TextFile;
begin
 AssignFile(F, 'c:\autoexec.bat');
  Append(F); { 파일 오픈 }
  Writeln(F, 'prompt $p$g');
  CloseFile(F);                { 파일 닫음 }
 end;
end.
```

AppendStr SysUtils Unit

문법 : procedure AppendStr(var Dest: string; const S: string);

두 개의 문자열을 결합한다. Dest 문자열 뒷부분에 Src 문자열이 추가된다. Dest:=Dest+S와 문법적으로 동일한 기능을 한다.

```
Src:='Apple';
Dest:='Orange';
AppendStr(Dest,Src);
```

Dest 문자열은 OrangeApple가 된다.

ArcTan system Unit

문법 : function ArcTan(X:Real):Real;

삼각함수 아크 탄젠트를 구한다. 인수 X는 360분법의 각도가 아닌 호도값이다.

AssignCrt WinCrt Unit

문법 : procedure AssignCrt(var f:Text);

텍스트 파일 변수를 CRT 윈도우와 연결한다. CRT와 연결된 텍스트 파일은 텍스트 환경의 화면을 에뮬레이션하며 이 파일로 문자열을 출력하면 CRT 화면으로 문자열이 출력된다. 표준 입출력 명령보다 텍스트 파일을 사용하여 CRT로 입출력하는 것이 속도가 조금 더 빠르다. 다음 예는 SF라는 텍스트 파일을 CRT로 연결하여 문자열을 출력한다.

```
var
  SF:TextFile;
begin
  AssignCRT(SF);
  Rewrite(SF);
  Writeln(SF,'Screen File');
  CloseFile(SF);
end;
```

Assigned

system Unit

문법 : procedure Assigned(var P):Boolean;

포인터 변수가 값을 가지고 있는지 가지고 있지 않은지(즉 nil)를 검사해 본다. P<>nil 과 문법적으로 동일한 조건식이며 P가 nil이면 False를 리턴하며 nil이 아니면 True를 리턴한다. 이 프로시저는 이벤트 핸들러 호출 전에 이벤트가 이벤트 메소드를 가지고 있는지 아닌지를 점검할 때 사용된다.

AssignFile

system Unit

문법 : procedure AssignFile(var f,String);

텍스트 파일 변수와 외부 파일을 연결한다. 즉 파일 핸들에 파일을 할당한다. 일단 파일과 핸들이 연결되면 핸들에 대한 모든 조작은 파일로 입출력된다. 이 프로시저로 핸들에 파일을 할당한 후 Reset이나 Rewrite 등을 사용하여 파일을 오픈하며 WriteLn이나 Readln으로 파일 입출력을 수행한다. 파일 이름은 문자열 형태로 전달하되 필요할 경우 드라이브명과 디렉토리명을 사용할 수 있다. 주의할 것은 이미 열려져 있는 파일 핸들을 재할당해서는 안된다는 점이다. 다음 예는 test.txt 파일에서 첫 줄을 읽어 에디트 박스로 출력한다.

```
var
  F: TextFile;
  S: String;
begin
  AssignFile(F, 'test.txt');
  Reset(F);
  Readln(F, S);
  Edit1.Text := S;
  CloseFile(F);
end.
```

AssignPrn

Printers Unit

문법 : procedure AssignPrn(var F: Text);

텍스트 파일 변수 F를 프린터로 할당한다. 프린터는 쓰기 전용의 텍스트 파일이므로 이 프로시저로 할당을 한 후 Rewrite 프로시저로 파일을 오픈해야 한다. 일단 파일 변수가 프린터로 할당되고 오픈되면 이 파일 변수로 보내는 모든 출력은 프린터로 인쇄된다. 파일로 문자열을 출력하는 Writeln 프로시저로 문자열을 프린터로 전송한다.

AssignStr

system Unit

문법 : procedure AssignStr(var P: PString; const S: String);

동적으로 할당된 문자열에 새로운 문자열을 대입한다. AssignStr(P,S)는 DisposeStr(P); P:=NewStr(S)와 동일하다. AssignStr은 문자열 포인터를 초기화하지 않으므로 문자열 포인터 P는 nil이거나 아니면 유효한 문자열 포인터이어야 한다.

```
var
  P: PString;
begin
  P := NewStr('First string');  { 문자열 할당 }
  AssignStr(P, 'Second string'); { 새문자열 할당 }
  DisposeStr(P);   { 할당 취소 }
end.
```

BlockRead

system Unit

문법 : procedure BlockRead(var f:File; var Buf; Count:Word [;var Result: Word]);

파일로부터 한 개 또는 다수 개의 레코드를 Buf 메모리 영역으로 읽어들인다. F는 언타입드 파일 변수이며 Buf는 어떤 형의 변수든지 상관없다. 읽혀지는 레코드의 개수는 Count인수로 설정하며 실제로 읽혀진 레코드 수는 Result 인수로 리턴된다. 파일로부터 읽혀진 레코드는 Buf 변수가 차지하고 있는 메모리의 시작 위치에 기록된다. 리턴값인 Result가 Count 인수와 같은 값을 가지면 요구한 레코드가 전부 읽혀진 것이며 Result가 Count보다 더 작은 값을 가지면 파일을 읽는 중에 EOF를 만난 것이다.

BlockWrite

system Unit

문법 : procedure BlockWrite(var f:File; var Buf; Count:Word [;var Result: Word]);

파일로 한 개 또는 다수 개의 레코드를 출력한다. F는 언타입드 파일 변수이며 Buf는 어떤 형의 변수든지 상관없다. 파일로 출력되는 레코드의 개수는 Count 인수로 설정하며 실제로 출력된 레코드 수는 Result 인수로 리턴된다. 파일로 출력되는 레코드는 Buf 변수가 차지하고 있는 메모리의 시작 위치에 있어야 한다. 리턴값인 Reulst가 Count 인수와 같은 값을 가지면 요구한 레코드가 이상없이 출력된 것이며 Result가 Count보다 더 작은 값을 가지면 디스크의 용량이 부족하여 레코드를 모두 출력하지 못한 것이다.

Bounds Classes Unit

문법 : function Bounds(ALeft, ATop, AWidth, AHeight: Integer): TRect;

주어진 좌상단 좌표와 높이, 넓이를 사용하여 TRect형의 사각 영역을 구한다. Rect는 좌상단, 우하단 두 점의 좌표로 사각 영역을 구하는 반면 이 함수는 좌상단과 크기를 사용하여 사각 영역을 구한다는 점이 다르다.

```
R := Bounds(X, Y, W, H);
```

즉 위와 같이 사각 영역을 구하는 식은 다음과 동일한 결과를 가져온다.

```
R := Rect(X, Y, X + W, Y + H);
```

Break system Unit

문법 : procedure Break;

for, while, repeat 등의 반복문의 중간에서 사용되며 루프를 끝내고 루프 다음의 명령으로 프로그램의 흐름을 옮기도록 한다. 만약 반복 루프의 외부에서 이 프로시저가 사용되면 에러이다. 다음 예는 While문으로 무한 루프를 구성하고 문자열을 입력받아 다시 출력하기를 무한 반복하되 아무 문자도 입력되지 않으면 루프를 종료한다.

```
var
  S: String;
begin
  while True do {무한 루프}
  begin
    ReadLn(S);
    if S = '' then Break; {빈 행이면 루프 종료}
    WriteLn(S);
  end;
end.
```

이 루프는 다음과 같이 작성해도 동일하다.

```
var
  S: String;
begin
  S:='temp'
  while S<>'' do
  begin
    ReadLn(S);
    WriteLn(S);
  end;
end.
```

S가 널 스트링일 때까지 반복을 하도록 변경하였다. 루프 시작시에 조건 점검을 하므로 루프 진입 전에 S가 널이 아니도록 아무 문자열이나 대입해 주어야 한다. 이렇게 하지 않으려면 선 실행 후 평가문인 repeat를 사용하면 된다.

ChangeFileExt system Unit

문법 : function ChangeFileExt(const FileName, Extension: string):string;

첫 번째 인수 FileName으로 주어진 파일의 이름 중 파일의 확장자만 두 번째 인수 Extension으로 변경한다. 다음 예는 인수로 주어진 파일의 확장자를 무조건 INI로 변경한다.

```
ChangeFileExt(ParamStr(0), '.INI');
```

Chdir system Unit

문법 : procedure ChDir(S: String);

인수 S가 지정하는 경로로 현재 디렉토리를 변경한다. 만약 S가 드라이브 문자를 포함하고 있을 경우 현재 드라이브도 변경된다.

Chr system Unit

문법 : function Chr(X:Byte):Char;

서수값 X가 지정하는 문자를 구한다. 아스키 코드값을 X로 전달하면 해당하는 문자를 얻을 수 있다. Chr(65)는 알파벳 문자 A이다. 이 함수를 사용하면 키보드로 직접 입력할 수 없는 문자를 입력할 수 있다.

Close system Unit

문법 : procedure Close(var F);

파일 핸들과 외부 파일과의 연결을 끊고 파일 핸들을 닫는다. 이 함수는 구 버전의 볼랜드 파스칼 컴파일러와의 호환성을 위해 제공되며 델파이에서는 사용되지 않는다. 델파이에서는 핸들을 닫는 CloseFile이라는 별도의 함수를 제공하므로 이 함수를 사용하도록 한다.

CloseFile system Unit

문법 : procedure CloseFile(var F);

파일 핸들과 외부 파일과의 연결을 끊고 파일 핸들을 닫는다. 파일 변수 F는 Reset, Rewrite, Append 등에 의하여 오픈된 파일 핸들이며 파일 핸들은 사용 후에 반드시 닫아주어야 한다.

ClrEol WinCrt Unit

문법 : procedure ClrEol;

현재 커서가 있는 위치에서 행의 끝부분까지의 모든 문자를 삭제한다. 커서는 움직이지 않으며 지워진 문자는 현재 정의된 문자 속성으로 빈칸이 된다.

Clrscr WinCrt Unit

문법 : procedure Clrscr;

활동중인 윈도우의 모든 문자들을 지우고 커서를 윈도우의 좌상단 위치로 이동시킨다. 모든 문자가 현재 정의된 문자 속성으로 빈칸이 된다.

ColorToIdent Graphics Unit

문법 : function ColorToIdent(Color: Longint; var Ident: string): Boolean;

32비트로 표현된 Color 색상값을 파스칼이 사용하는 컬러 명칭 문자열로 변경한다. Color에 대응되는 명칭이 있으면 Ident 문자열에 색상의 명칭을 대입하고 True를 리턴하지만 대응하는 명칭이 없을 경우 Ident는 변경되지 않으며 False를 리턴한다.
다음 예는 $ffffff라는 색상값에 대응하는 색상 명칭을 찾는다. 색상값이 흰색이므로 레이블로 출력되는 문자열은 clWhite가 된다.

```
var
  ID: string;
  ans:Boolean;
begin
ans:=ColorToIdent($ffffff,ID);
label1.caption:=ID;
end;
```

ColorToRGB Graphics Unit

문법 : function (Color: TColor): Longint;

델파이가 사용하는 TColor형의 색상값을 32비트의 RGB형태로 변경한다. 색상값이 시스템에 정의된 색상일 경우 어떤 색상으로 정의되어 있는지 RGB값을 조사할 수 있다. 다음 예는 폼의 색상을 RGB로 조사한다.

```
var
  L : Longint;
begin
  L := ColorToRGB(Form1.Color);
end;
```

ColorToString Graphics Unit

문법 : function (Color: TColor): string;

TColor형의 값을 문자열로 변경한다. 다음 예는 폼의 색상을 문자열로 변경한 후 레이블로 출력한다. 디폴트 색상을 사용하고 있다면 레이블로 출력되는 값은 clBtnFace일 것이다.

```
var
  ID: string;
begin
ID:=ColorToString(Form1.Color);
label1.caption:=ID;
end;
```

CompareStr system Unit

문법 : function CompareStr(const S1, S2: string): Integer;

두 개의 문자열 S1과 S2를 비교하되 대소문자를 구분하여 비교한다. S1과 S2 문자열이 같으면 0을 리턴하며 S1<S2이면 음수의 값을 리턴하며 S1>S2이면 양수를 리턴한다. 문자열끼리의 비교는 문자열을 이루는 각 문자의 문자 코드를 비교하여 수행되며 현재 설치된 언어 드라이버에는 영향을 받지 않는다.

CompareText system Unit

문법 : function CompareText(const S1, S2: string): Integer;

두 개의 문자열 S1과 S2를 비교하되 대소문자를 구분하지 않고 비교한다. S1과 S2 문자열이 같으면 0을 리턴하며 S1<S2이면 음수의 값을 리턴하며 S1>S2이면 양수를 리턴한다. 문자열끼리의 비교는 문자열을 이루는 각 문자의 문자 코드를 비교하여 수행된다. "Apple"와 "APPLE"를 CompareText 함수로 비교하면 결과는 0이며 두 문자열이 같은 것으로 비교되지만 CompareStr 함수로 비교하면 결과는 두 문자열이 서로 다른 것으로 비교된다.

Concat system Unit

문법 : function Concat(s1 [, s2,..., sn]: String): String;

두 개 이상의 문자열들을 결합하여 새로운 긴 문자열을 만든다. 만약 문자열을 연결한 결과가 255문자 이상일 경우 255번째 이후의 문자는 잘려 나간다. 이 함수 대신 +연산자를 사용하여 문자열을 결합해도 결과는 동일하다. Concat('Korea','China')와 'Korea'+'China'는 같은 결과를 만들어 낸다. 다음 예는 Edit1에 입력된 문자열과 Edit2에 입력된 문자열을 합해 Edit3에 대입한다.

```
Edit3.Text:=ConCat(Edit1.Text,Edit2.Text);
```

Continue system Unit

문법 : procedure Continue;

for, while, repeat 등의 반복문에서 사용되며 다음 반복 부분으로 흐름을 옮기도록 한다. 루프의 남은 뒷부분은 무시되며 루프의 조건 점검부로 점프한다. 만약 반복 루프 외부에서 이 프로시저가 사용되면 에러이다. 다음 예는 i가 1~100까지 증가하며 모종의 처리를 하되 i가 10인 경우만 특별히 처리를 생략한다.

```
for i:=1 to 100 do
begin
  if i=10 then continue;
  ....
end;
```

Copy system Unit

문법 : procedure Copy(S:String; Index, Count:Integer):String;

한 문자열의 부분 문자열을 추출해 낸다. S 문자열의 Index 위치에서부터 Count 문자분의 부분 문자열이 추출된다. Index가 문자열의 전체 길이보다 길 경우 빈 문자열을 리턴하며 Count가 문자열의 남은 부분보다 클 경우 문자열의 끝까지 추출해 낸다. Dest 문자열이 'Orange'일 경우 Copy(Dest,2,3)은 Dest 문자열의 두 번째 문자에서부터 3문자분의 부분 문자열인 'ran'을 추출해낸다.

Cos system Unit

문법 : function Cos(X;Real):Real;

코사인 삼각함수 값을 구한다. 여기서 사용되는 인수 X는 360분법의 각도가 아닌 호도값이다.

CSeg system Unit

문법 : function CSeg:Word

CS 레지스터의 현재값을 구한다. CS 레지스터는 실행 코드의 세그먼트 값을 담는 레지스터이다.

CursorTo system Unit

문법 : procedure CursorTo(X,Y:Integer);

커서의 위치를 인수로 주어진 (X,Y) 좌표로 옮긴다. CRT 윈도우의 좌상단 좌표는 (0,0)이다. CursorTo 프로시저는 Cursor 변수값을 (X,Y)로 변경한다.

Date system Unit

문법 : fun;

시스템의 시계를 참조하여 오늘 날짜를 구한다. 날짜는 TDateTime형이므로 곧바로 문자열로 출력할 수 없으며 DateToStr 함수를 사용하여 문자열로 바꾸어 주어야 한다. Label1.Caption:=DateToStr(Date) 명령에 의해 Label1에 오늘 날짜가 출력된다.

DateTimeToFileDate SysUtils Unit

문법 : function DateTimeToFileDate(DateTime: TDateTime): Longint;

TDateTime형의 날짜, 시간 값을 DOS 형식의 날짜, 시간 형식으로 바꾼다. 도스 형식의 날짜는 4바이트의 정수형이며 각 비트에 날짜, 시간 요소를 포함하고 있다. 다음 예는 도스 형식의 날짜, 시간 형식을 레이블로 출력한다.

```
label1.caption:=IntToStr(DateTimeToFileDate(Now));
```

DateTimeToStr system Unit

문법 : function DateTimeToStr(DateTime: TDateTime): String;

날짜와 시간을 담은 TDateTime형의 변수를 문자열로 바꾼다. 만약 DateTime 인수가 날짜를 포함하지 않으면 날짜는 00/00/00이 되며 시간을 포함하지 않으면 00:00:00 AM이 된다. 다음 예는 현재 시간과 날짜를 레이블로 출력한다.

```
Label1.Caption := DateTimeToStr(Now);
```

DateTimeToString system Unit

문법 : procedure DateTimeToString(var Result: string; const Format: string; DateTime: TDateTime);

날짜, 시간을 담고 있는 DateTime을 Format에 주어진 형식대로 문자열로 변환하여 Result 문자열에 기입한다. Format 인수에 사용할 수 있는 지정자에 관한 사항은 FormatDateTime 함수를 참조하기 바란다.

DateToStr system Unit

문법 : function DateToStr(Date: TDateTime): String;

날짜를 담는 TDateTime형의 변수에서 날짜를 문자열로 바꾼다. 다음 예는 오늘 날짜를 레이블로 출력한다.

```
Label1.Caption := DateToStr(Date);
```

DayOfWeek system Unit

문법 : function DayOfWeek(Date: TDateTime): Integer;

특정 날짜의 요일을 계산한다. 리턴되는 값은 1~7까지의 정

수이며 1이 일요일, 7이 토요일이다. 리턴되는 값이 정수형이므로 월, 화, 수, 목 등의 실제 요일 이름으로 바꾸어 주어야 한다. 다음 예는 요일 이름을 레이블로 출력한다. 다음 예는 오늘이 무슨 요일인지 조사해서 요일 이름을 레이블로 출력해준다.

```
var
  YO:string;
  ONUL:TDateTime;
begin
  ONUL:=Now;
  case DayOfWeek(ONUL) of
    1:YO:='일';
    2:YO:='월';
    3:YO:='화';
    4:YO:='수';
    5:YO:='목';
    6:YO:='금';
    7:YO:='토';
  end;
  label1.caption:='오늘은 '+YO+'요일입니다.';
end;
```

Dec

system Unit

문법 : procedure Dec(var X[; N:Longint]);

변수의 값을 1 감소시킨다. Dec(X)는 X:=X-1과 동일한 동작을 한다. 두 번째 인수를 사용하면 1 이상의 값을 감소시킬 수도 있다. Dec(X,5)는 X:=X-5와 동일한 동작을 한다. 첫 번째 인수는 서수형의 변수이며 확장 문법이 적용될 경우 PChar형도 가능하다. N은 정수형의 상수 또는 변수여야 한다. Dec 함수는 가장 최적화된 코드를 생성해 내므로 루프 내부에서 제어 변수값을 감소시키는 용도로 사용하기에 적합하다.

DecodeDate

system Unit

문법 : procedure DecodeDate(Date: TDateTime; var Year, Month, Day: Word);

날짜를 담는 TDateTime형의 변수에서 년, 월, 일의 값을 분리시킨다. 분리된 값들은 각각 Year, Month, Day 등의 정수형 변수에 대입된다. 날짜값은 DateToStr 함수로, 문자열로 바꾼 후 한꺼번에 출력할 수 있지만 개별적인 요소를 가공한 후 출력하고자 할 경우는 이 함수를 사용한다. 이 함수의 반대 함수는 EncodeDate 함수이다. 다음 예는 오늘 날짜와 현재 시간을 조사한 후 문자열 조립을 통해 말로 시간과 날짜를 알려 준다.

```
var
  Present: TDateTime;
  Year, Month, Day, Hour, Min, Sec, MSec: Word;
begin
  Present:= Now;
  DecodeDate(Present, Year, Month, Day);
  Label1.Caption := '오늘은' + IntToStr(Year) + '년'+ IntToStr(Month) + '월' + IntToStr(Day)+'일입니다.';
  DecodeTime(Present, Hour, Min, Sec, MSec);
  Label2.Caption := '지금은' + IntToStr(Hour) + '시'+ IntToStr(Min)+'분입니다.';
end;
```

DecodeTime

system Unit

문법 : fun;

시간을 담는 TDateTime형의 변수에서 시, 분, 초의 값을 분리시킨다. 분리된 값들은 각각 Hour, Min. Sec, MSec 등의 정수형 변수에 대입된다. 시간값은 TimeToStr 함수로 문자열로 바꾼 후 한꺼번에 출력할 수 있지만 개별적인 요소를 가공한 후 출력하고자 할 경우는 이 함수를 사용한다. 이 함수의 반대 함수는 EncdoeTime 함수이다. DecodeDate 함수의 예제를 참조하기 바란다.

Delete

system Unit

문법 : procedure Delete (var S:String; Index, Count:Integer);

한 문자열에서 부분 문자열을 삭제한다. S 문자열의 Index 위치에서부터 Count 문자분의 부분 문자열이 삭제된다. Index가 문자열의 전체 길이보다 길 경우 삭제는 이루어지지 않으며 Count가 문자열의 남은 부분보다 클 경우 문자열이 끝까지 삭제한다. Dest 문자열이 'Orange'일 경우 Delete(Dest,2,3)은 Dest 문자열의 두 번째 문자에서부터 3문자분의 부분 문자열인 'ran'을 삭제하며 Dest 문자열은 'Oge'가 된다.

DeleteFile

SysUtils Unit

문법 : function DeleteFile(const FileName: string): Boolean;

디스크 상의 파일을 지운다. 파일이 없거나 읽기 전용 속성을 가질 경우 False를 리턴하며 파일은 지워지지 않는다. 그러나 파일을 지울 수 없는 경우라도 예외는 발생하지 않는다. 파일은 경로를 포함할 수 있다. 다음 예는 C 드라이브의 루트에 있는 COMMAND.COM 파일을 지운다. 물론 이 명령을 실행한 후부터 도스는 부팅되지 않는다.

```
DeleteFile('c:\command.com');
```

DirectoryExists

FileCtrl Unit

문법 : function DirectoryExists(Name: string): Boolean;

인수로 전달된 디렉토리가 존재하는지 검사하며 존재할 경우 True를 리턴하고 존재하지 않을 경우 False를 리턴한다.

DiskFree SysUtils Unit

문법 : function DiskFree(Drive: Byte): Longint;

인수로 지정한 Drive에 사용 가능한 용량을 조사해준다. 드라이브 번호는 0이 현재 드라이브, 1이 A, 2가 B 등이다. 드라이브 번호가 잘못되었을 경우는 -1을 리턴한다.

DiskSize SysUtils Unit

문법 : function DiskSize(Drive: Byte): Longint;

인수로 지정한 Drive의 총 용량을 조사해준다. 드라이브 번호는 0이 현재 드라이브, 1이 A, 2가 B 등이다. 드라이브 번호가 잘못되었을 경우는 -1을 리턴한다.

Dispose system Unit

문법 : procedure Dispose(var P:Pointer);

동적으로 할당된 포인터의 메모리를 할당 해제한다 할당이 해제된 포인터 값은 정의되어 있지 않으며 이 포인터를 사용하여 메모리를 참조할 수 없다. 다음 예제는 문자열 변수를 동적으로 생성하여 사용한 후 해제한다.

```
var
  P: ^string;
begin
  New(P);
  P^ := 'Now you see it...';
  Dispose(P);
end.
```

DisposeStr SysUtils Unit

문법 : fun;

NewStr 함수에 의해 힙에 할당한 메모리를 해제한다. 주어진 문자열 포인터가 nil이거나 빈 문자열이면 아무 동작도 하지 않는다.

DoneWinCrt WinCrt Unit

문법 : procedure DoneWinCrt;

CRT윈도우를 파괴한다.

DrawButtonFace Buttons Unit

문법 : function DrawButtonFace(Canvas: TCanvas; const Client: TRect; BevelWidth: Integer; Style: TButtonStyle; IsRounded, IsDown, IsFocused: Boolean): TRect;

스피드 버튼과 비트맵 버튼의 Paint, DrawItem 메소드에서 내부적으로 버튼의 모양을 그리기 위해 사용하는 함수이다. 그러나 사용자도 직접 이 함수를 사용하면 폼이나 이미지의 캔버스 표면에 버튼 모양의 그림을 쉽게 그릴 수 있다. 인수의 의미는 인수의 이름으로 쉽게 유추할 수 있으므로 설명을 생략한다. 다음 예는 폼의 좌상단에 버튼 모양의 불쑥 튀어나온 모양을 그린다. 인수를 바꾸어 가면 다양한 버튼 모양을 직접 디자인 해 보아라. 실행 중에 폼을 장식하는 용도로 적합한 함수이다.

```
procedure TForm1.Button1Click(Sender: TObject);
begin
DrawButtonFace(Canvas,RECT(10,10,120,50)
  ,3,bsNew,True,False,True);
end;
```

DSeg system Unit

문법 : function DSeg:Word;

DS 레지스터의 현재값을 구한다. DS 레지스터는 변수 저장 영역의 세그먼트 값을 담는 레지스터이다.

EncodeDate SysUtils Unit

문법 : function EncodeDate(Year, Month, Day: Word): TDateTime;

정수형으로 주어진 년, 월, 일의 값을 사용하여 날짜를 담은 TDateTime형의 변수 하나를 만든다. Year는 1~9999까지의 값을 가지며 Month는 1~12, Day는 1~31까지 가능하다. 단 Day 값의 범위는 Month 값의 영향을 받으며 또는 Year가 윤년인 경우의 영향도 받는다. 만약 날짜값이 무효일 경우, 예를 들어 2월 30일 등의 값이 주어지면 EConvertError 예외가 발생한다. 년, 월, 일을 개별적으로 입력받아 하나의 날짜값을 만들고자 할 때 이 함수를 사용한다.

EncdoeTime SysUtils Unit

문법 : function EncodeTime(Hour, Min, Sec, MSec: Word): TDateTime;

정수형으로 주어진 시, 분, 초의 값을 사용하여 시간을 담은 TDateTime형의 변수 하나를 만든다. Hour는 0~12까지 가능하며 만약 Time24Hour가 True일 경우 0~23까지의 24시간을 표현할 수 있다. Min, Sec은 0~59까지 가능하며 MSec은 0~999까

지 가능하다. 시간값이 무효일 경우, 예를 들어 8시 86분 등의 값이 주어지면 EConvertError 예외가 발생한다. 시, 분, 초를 개별적으로 입력받아 하나의 시간값을 만들고자 할 때 이 함수를 사용한다.

Eof

system Unit

문법 : function Eof [(var F:Text)]:Boolean;

파일 변수 F의 현재 위치(FP)가 파일의 끝일 경우 True를 리턴한다. 다음 예제는 input.dat파일을 output.dat파일로 복사한다. 두 파일을 열어놓고 입력 파일에서 읽어 출력 파일로 쓰기를 입력파일의 끝까지 반복한다. 이 때 입력 파일의 끝인가를 점검하기 위해 Eof 함수가 사용되었다. 파일의 처음부터 끝까지 한 바이트씩 순서대로 작업을 해야할 경우에 사용한다.

```
var
  F1, F2: TextFile;
  Ch: Char;
begin
  AssignFile(F1, 'input.dat');
  Reset(F1);
  AssignFile(F2, 'output.dat');
  Rewrite(F2);
  while not Eof(F1) do
    begin
    Read(F1, Ch);
    Write(F2, Ch);
    end;
  CloseFile(F2);
  CloseFile(F1);
end.
```

Eoln

system Unit

문법 : function Eoln [{var F:Text)]:Boolean;

파일 변수 F의 현재 위치(FP)가 한 줄의 끝인지를 점검하고 줄끝일 경우 True를 리턴한다. 한 줄씩 처리하고자 할 때 이 함수를 사용한다.

Erase

system Unit

문법 : procedure Erase(var F);

외부 파일을 삭제한다. DeleteFile은 파일명으로 삭제하는 반면 이 프로시저는 파일 핸들로 파일을 삭제한다. 파일을 삭제한 후 파일 핸들은 자동으로 닫혀진다.

Exclude

system Unit

문법 : procedure Exclude(var S:Set of T;I:T);

집합형 변수 S로부터 요소 I를 삭제한다. 이 프로시저는 S:=S-(I);동일한 동작을 하지만 Exclude가 더 효율적인 코드를 만든다.

Exit

system Unit

문법 : procedure Exit;

현재 블럭을 종료하며 현재 블럭이 main 블럭이면 프로그램을 종료한다. 중첩 루프 내에서 Exit 프로시저는 현재 블럭의 바로 다음 블럭으로 제어를 옮기도록 한다. 함수나 프로시저의 내부에서 Exit는 함수 호출문 바로 다음 문장을 수행하도록 한다.

Exp

system Unit

문법 : function Exp(X:Real):Real;

자연대수 exp를 계산한다. 이는 수학적으로 e의 x승을 의미하며 e는 2.718281...의 무리수이다.

ExpandFileName

SysUtils Unit

문법 : function ExpandFileName(const FileName: string): string;

파일 이름에 드라이브명과 디렉토리명을 붙여 완전 경로(full path)를 만들어낸다. 덧붙여지는 경로는 현재 드라이브와 현재 디렉토리이다.

ExtractFileExt

SysUtils Unit

문법 : function ExtractFileExt(const FileName string): string;

완전 경로에서 파일의 확장자만 분리시킨다. 특정 파일의 확장자를 알고자 할 때 이 함수를 사용하여 확장자만 분리한 후 문자열 비교를 수행하면 된다.

ExtractFileName

SysUtils Unit

문법 : function ExtractFileName(const FileName: string): string;

완전 경로에서 파일 이름만 분리해 낸다. 파일의 확장자도 같이 분리된다. 즉 완전 경로에서 드라이브명과 디렉토리 명이 제외된다.

ExtractFilePath

SysUtils Unit

문법 : function ExtractFilePath(const FileName: string): string;

완전 경로에서 파일 이름을 제외하고 드라이브명과 디렉토리명만 분리해 낸다. 다음 예는 완전 경로를 가진 FileName으로부터 경로만 분리해낸 후 파일이 있는 디렉토리로 이동한

다.

```
ChDir(ExtractFilePath(FileName));
```

FindControl — Controls Unit

문법 : function FindControl(Handle: HWnd): TWinControl;

윈도우 핸들에 대응되는 윈도우 컨트롤을 찾아준다. 핸들에 대응되는 윈도우 컨트롤이 없거나 핸들이 0이면 nil을 리턴한다.

FileAge — SysUtils Unit

문법 : function FileAge(const FileName: string): Longint;

파일의 날짜와 시간을 구해낸다.

FileClose — SysUtils Unit

문법 : procedure FileClose(Handle: Integer);

파일 핸들을 닫는다. 파일 입출력을 위해 파일 변수에 외부 파일을 연결한 경우 핸들을 다 사용한 후에 반드시 파일 핸들을 닫아 주어야 한다.

FileDateToDateTime — SysUtils Unit

문법 : function FileDateToDateTime(FileDate: Longint): TDateTime;

도스에서 사용하는 날짜, 시간 형식의 변수를 TDateTime형으로 바꾼다. 도스 형식의 날짜는 4바이트의 정수형이며 각 비트에 날짜, 시간 요소를 포함하고 있다.

FileExists — SysUtils Unit

문법 : function FileExists(const FileName: string): Boolean;

파일이 디스크 상에 존재하는지를 조사하며 파일이 없을 경우 False를 리턴한다. 존재하지 않는 파일을 사용하고자 할 경우 예외가 발생하므로 먼저 이 함수를 사용하여 파일이 있는지를 먼저 확인해 보아야 한다. 다음 예는 파일의 존재 여부를 점검해 본 후 파일을 삭제한다.

```
if FileExists(FileName) then
        DeleteFile(FileName);
```

FileGetAttr — SysUtils Unit

문법 : function FileGetAttr(const FileName: string): Integer;

파일의 속성을 조사한다. 조사되는 파일의 속성은 다음과 같다. 속성을 나타내는 이 상수들과 FileGetAttr이 리턴하는 값을 AND 연산시켜 특정 속성의 유무를 조사한다. 속성 변경에 실패하면 음수의 에러 코드를 리턴한다.

상수	값	의미
faReadOnly	$01	읽기전용
faHidden	$02	숨은 파일
faSysFile	$04	시스템 파일
faVolumeID	$08	디스크 볼륨
faDirectory	$10	디렉토리
faArchive	$20	기록 속성
faAnyFile	$3F	모든 파일

FileGetDate — SysUtils Unit

문법 : function FileGetDate(Handle: Integer): Longint;

파일이 처음 생성된, 또는 최후 수정된 날짜와 시간을 구한다. 이 날짜는 도스의 dir 명령시 나타나는 날짜이며 도스 고유의 포맷을 사용하므로 델파이에서 이 날짜 정보를 사용하려면 FileDataToDateTime 등의 포맷 변환 함수를 사용해야 한다.

FileOpen — SysUtils Unit

문법 : function FileOpen(const FileName: string; Mode: Word): Integer;

델파이가 사용하는 내부적인 함수이므로 사용자가 직접 사용하는 경우는 드물다. 파일을 오픈할 때는 델파이가 제공하는 Reset, Rewrite, Append 등의 프로시저를 사용한다.

FilePos — system Unit

문법 : function FilePos(var F):Longint;

파일의 현재 위치(FP)를 구한다. 이 함수는 텍스트 파일에는 사용할 수 없으며 파일이 열려 있어야 한다.

FileRead — SysUtils Unit

문법 : function FileRead(Handle: Integer; var Buffer; Count: Longint): Longint;

핸들이 지시하는 파일로부터 Count 바이트 분만큼을 버퍼로 읽어들인다. 파일로부터 실제 읽은 바이트 수가 리턴된다.

FileSearch — SysUtils Unit

문법 : function FileSearch(const Name, DirList: string): string;

DirList에 주어진 디렉토리의 경로에 Name 파일이 있는지 조사해준다. DirList는 도스의 PATH문에서와 같이 여러 개의 디렉토리명을 세미콜론으로 끊어 기입한다. 파일을 찾을 경우 파일의 완전 경로를 리턴하며 파일을 찾지 못할 경우 널 스트링을 리턴한다.

FileSeek SysUtils Unit

문법 : function FileSeek(Handle: Integer; Offset: Longint; Origin: Integer): Longint;

파일의 현재 위치(FP)를 특정 위치로 옮긴다. 옮겨지는 위치는 Origin이 정하는 기준에서 Offset바이트만큼 떨어진 위치이다. Origin의 값은 다음과 같다.

Origin	의미
0	파일의 처음을 기준으로 한다.
1	파일의 현재 위치를 기준으로 한다.
2	파일의 끝을 기준으로 한다.

FileSetAttr SysUtils Unit

문법 : function FileSetAttr(const FileName: string; Attr: Integer): Integer;

파일의 속성을 설정한다. 설정할 수 있는 속성에는 다음과 같은 종류가 있다. 여러 개의 속성을 동시에 설정하려면 속성 상수를 OR 연산으로 연결하여 사용한다. 속성 변경이 정상적으로 수행되면 0을 리턴하며 에러 발생시 음수의 에러 코드를 리턴한다.

상수	값	의미
faReadOnly	$01	읽기 전용
faHidden	$02	숨은 파일
faSysFile	$04	시스템 파일
faVolumeID	$08	디스크 볼륨
faDirectory	$10	디렉토리
faArchive	$20	기록 속성
faAnyFile	$3F	모든 파일

FileSetDate SysUtils Unit

문법 : procedure FileSetDate(Handle: Integer; Age: Longint);

파일의 날짜를 변경한다. 이 날짜는 도스의 dir 명령시 나타나는 날짜이며 도스 고유의 포맷을 사용한다.

FileSize system Unit

문법 : function FileSize(var F);Longint;

파일 핸들 F가 지정하는 파일의 크기를 바이트 단위로 조사한다. 단 F가 레코드 파일일 경우는 레코드의 개수를 조사한다.

FillChar system Unit

문법 : procedure FillChar(var X:Count:Word;Value);

Count수만큼 값 Value를 채운다. Value는 Byte형이거나 Char형이어야 한다. 이 프로시저는 범위 체크를 전혀 하지 않으므로 사용에 주의를 기울여야 한다. 다음 예는 문자열 S에 문자열의 길이만큼 모두 문자 'T'로 가득 채운다.

```
FillChar(S, SizeOf(S), 'T');
```

FindClose SysUtils Unit

문법 : procedure FindClose(var SearchRec: TSearchRec);

FindFirst, FindNext 함수를 사용한 파일 검색을 종료한다. 굳이 파일 검색 작업을 종료해 주어야 할 필요는 없으며 실제로 16비트 버전의 윈도우즈에서 이 함수는 아무런 일도 하지 않는다. 하지만 32비트 버전에서는 반드시 검색 종료를 해 주어야 하므로 호환성을 위해 검색 후 종료해 주는 것이 좋다.

FindFirst SysUtils Unit

문법 : function FindFirst(const Path: string; Attr: Word; var F: TSearchRec): Integer;

주어진 검색 조건과 파일의 속성을 사용하여 첫 번째 일치하는 파일을 검색해 낸다. Path는 검색하고자 하는 디렉토리와 파일의 조건이며 와일드 카드식으로 표현된다. 예를 들어 'c:\windows*.exe'는 windows 디렉토리의 확장자가 EXE인 첫 번째 파일을 검색한다. Attr인수는 검색 대상이 되는 파일의 속성을 지정하며 다음과 같은 속성 상수를 사용한다.

상수	값	의미
faReadOnly	$01	읽기 전용
faHidden	$02	숨은 파일
faSysFile	$04	시스템 파일
faVolumeID	$08	디스크 볼륨
faDirectory	$10	디렉토리
faArchive	$20	기록 속성
faAnyFile	$3F	모든 속성

여러 가지 속성을 사용할 경우 or 연산자로 속성을 연결한다. 예를 들어 읽기 전용이면서 숨은 파일을 검색하고 싶다면 (faReadOnly + faHidden)과 같이 Attr 인수를 설정한다. 검색한 결과는 TSearchRec 레코드형의 F변수에 저장되며 다음과 같은 정보를 가지고 있다.

```
TSearchRec = record
  Fill: array[1..21] of Byte;
  Attr: Byte;
```

```
  Time: Longint;
  Size: Longint;
  Name: string[12];  {파일 이름}
end;
```

이 정보는 FindNext 함수에 의해 다시 사용되며 계속 검색을 수행한다.

FindNext SysUtils Unit

문법 : function FindNext(var F: TSearchRec): Integer;

FindFirst에 이어 계속 검색을 수행한다. FindFirst가 첫 번째 검색한 결과를 저장한 F를 다시 인수로 넘겨주면 계속해서 일치하는 조건을 가진 파일을 검색해 내며 검색의 결과는 역시 F로 저장된다. 검색이 계속 이루어지고 있다면 0을 리턴하며 검색 중 에러가 발생했거나 더 이상 조건이 일치하는 파일이 없으면 음수의 에러 코드를 리턴한다. FindFirst와 FindNext는 조건에 맞는 모든 파일을 검색해 낼 때 사용한다.

FloatToDecimal SysUtils Unit

문법 : procedure FloatToDecimal(var Result: TFloatRec; Value: Extended; Precision, Decimals: Integer);

부동 소수점 형태의 수를 문자열과 소수점 위치값 등을 가진 TFloatRec형의 레코드로 변환한다. 이 레코드는 실수값을 적당히 포맷하여 출력하는 용도로 사용된다. Precision은 유효숫자를 지정하며 1~18까지의 값을 가진다. Decimal은 소수점 좌측의 정수부를 몇 자리까지 나타낼 것인가를 지정한다. 변환된 결과는 TFloatRec 형의 구조체에 저장되어 리턴되며 이 구조체는 다음과 같이 정의되어 있다.

```
TFloatRec = record
  Exponent: Integer;
  Negative: Boolean;
  Digits: array[0..18] of Char;
end;
```

각 필드는 다음과 같은 의미를 가진다.

필드	의미
Exponent	10의 거듭승을 나타내며 이는 실수값에 있는 정수부의 자리수이다. 실수값이 1보다 더 작다면 이 필드값은 음수가 된다. 실수값이 수치가 아니라면(NAN) -32768, 무한대(INF)라면 32767의 값을 가진다.
Negative	음수이면 True, 양수이면 False
Digits	18자리수의 유효숫자를 가지는 널 종료 문자열이다. 소수점은 문자열에 포함되어 있지 않으며 후행 제로는 모두 제거된다.

다음 예는 이 함수의 동작을 살펴보기 위해 제작한 것이다. 폼의 레이블에 어떻게 출력되는지 값을 변경해 가며 살펴보도록 하자.

```
var
  FR:TFloatRec;
  Value:double;
begin
Value:=314.159265;
FloatToDecimal(FR,Value,5,10);
Label1.Caption:='Exponent='+IntToStr(FR.Exponent);
Label2.Caption:='Digits='+FR.Digits;
end;
```

이 코드를 실행하면 Exponent는 3,Digits는 31416이 출력된다.

FloatToStrF SysUtils Unit

문법 : function FloatToStrF(Value: Extended; Format: TFloatFormat; Precision, Digits: Integer): string;

인수로 주어진 포맷과 정밀도를 사용하여 실수값을 문자열로 바꾼다. Precision은 유효자리수를 지정하며 실수값이 Single형일 경우는 7이하, Double일 경우는 15이하, Extended형일 경우는 18이하여야 한다. Digits의 의미는 Format 인수에 따라 달라진다. Format 인수에는 다음과 같은 종류가 있다.

값	의미
ffGeneral	가장 일반적인 숫자 포맷을 사용한다. 변환된 결과는 고정 소수점 형식, 공학적 표기법 중 길이가 짧은 쪽이 선택되며 후행 제로는 모두 제거된다. 소수점은 꼭 필요한 경우에만 나타난다.
ffExponent	공학적 표기법, 즉 부동 소수점 형태의 문자열로 변환된다.
ffFixed	고정 소수점 형태의 문자열로 변환된다.후행 제로는 제거되지 않는다.
ffNumber	ffFixed와 마찬가지로 고정 소수점 포맷 형식을 사용하나 천단위마다 콤마가 삽입된다.
ffCurrency	화폐 단위 형태의 문자열로 변환된다. 화폐 단위 표현의 형태는 Currency, CurrencyFormat 등의 전역 변수에 영향을 받으며 우리나라의 경우 수치 앞에 원(₩)기호가 첨가된다.

주어진 실수가 수치가 아닐 경우 변환된 문자열은 'NAN'이

되며 양의 무한대 수이면 'INF', 음의 무한대 수이면 '-INF'가 된다. 다음 예제는 이 함수를 사용하여 실수값을 문자열로 바꾼다.

```
var
  F:double;
  st:string;
begin
F:=3141.59265;
st:=FloatToStrF(F,ffGeneral,7,10);
label1.Caption:=st;
end;
```

변환된 결과는 3141.593이다. Format 인수를 ffExponent로 바꾸면 3.131593E+3으로 변환되며 ffNumber로 바꾸면 3,141.5930000000로 변환된다. ffCurrency로 바꾸면 W4,131.5930000000로 변환된다.

FloatToStr SysUtils Unit

문법 : function FloatToStr(Value: Extended): string;

FloatToStrF 함수와 동일한 기능을 가지되 일반적인 수치 포맷(ffGeneral)을 사용하며 15자리의 유효 숫자를 사용한다. 즉 이 함수는 변환 포맷이 미리 정의되어 있는 FloatToStrF 함수라고 할 수 있다.

FloatToText SysUtils Unit

문법 : function FloatToText(Buffer: PChar; Value: Extended; Format: TFloatFormat; Precision, Digits: Integer): Integer;

실수를 문자열로 변경하며 변경된 결과 버퍼에 저장된 문자의 개수를 리턴한다. 각 인수의 의미는 FloatToStrF 함수와 동일하므로 참고하기 바란다.

FloatToTextFmt SysUtils Unit

문법 : function FloatToTextFmt(Buffer: PChar; Value: Extended; Format: PChar): Integer;

FloatToText 함수와 동일한 기능을 하며 실수값을 문자열로 변경하여 버퍼에 저장한다.

Flush system Unit

문법 : procedure Flush(var F:Text);

출력용으로 오픈된 텍스트 파일의 버퍼를 비운다. 버퍼를 비운다는 말은 파일로 출력되기 위해 버퍼에 대기하고 있던 데이터를 외부 파일로 강제 전송한다는 뜻이다. 입력용으로 열려진 파일에는 아무런 동작도 하지 않는다.

FmtLoadStr SysUtils Unit

문법 : function FmtLoadStr(Ident: Word; const Args: array of const): string;

실행 파일의 리소스 문자열 테이블에서 문자열을 읽어들인다. Ident 인수는 리소스 파일 내의 문자열 리소스 ID이다.

FmtStr SysUtils Unit

문법 : procedure FmtStr(var Result: string; const Format: string; const Args: array of const);

Args 배열에 있는 항목들을 적절한 형태로 포맷하여 Result문자열에 대입해준다. 포맷 형식은 Format 인수의 서식 문자열에 따라 달라진다. 자세한 포맷 형식에 대해서는 Format 함수를 참고하기 바란다.

ForceDirectories FileCtrl Unit

문법 : procedure ForceDirectories(Dir: string);

완전 경로를 주면 경로에 이르는 모든 디렉토리를 생성한다. 일반적으로 디렉토리는 한 번에 하나만 만들 수 있지만 이 함수를 쓰면 여러 개의 디렉토리를 한꺼번에 만들 수 있다. 예를 들어 C;\a\b\c 디렉토리를 만들고자 할 경우 MkDir을 사용하면 a부터 만들고 b를 만든 후 c를 만들어야 하며 한꺼번에 세 개의 디렉토리를 만들 수 없다. 이 프로시저를 사용하면 한꺼번에 디렉토리를 생성할 수 있으므로 만들고자 하는 디렉토리의 완전 경로만 입력해 주면 된다. 다음 예는 C드라이브에 한꺼번에 12개의 디렉토리를 계층적으로 생성한다.

```
ForceDirectories('c:\a\b\c\d\e\f\g\h\i\j\k\l');
```

Format SysUtils Unit

문법 : function Format(const Format: string; const Args: array of const): string;

서식 문자열과 뒤따라오는 변수에 의해 문자열을 포맷한다. C의 printf 함수와 사용 방법이 유사하며 문자열 내부에 변수값을 삽입할 수 있는 아주 편리한 방법을 제공한다. 서식 문자열에는 일반 문자열과 서식이 삽입되며 일반 문자열은 그대로 출력되지만 서식은 대응되는 인수값으로 채워진다. 서식은 %기호와 변수의 타입으로 이루어지며 %와 타입 사이에 인덱스, 좌측 정렬 지정, 폭, 정밀도 지정이 삽입된다.

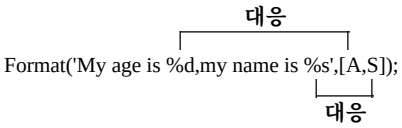
%[인덱스:][-][폭][.정밀도]타입

일단 Format 함수를 사용하는 예를 보자.

```
var
  A:integer;
```

```
 S,st:string;
begin
A:=정수값;
S:=문자열값;
st:=Format('My age is %d,my name is %s',[A,S]);
end;
```

서식 문자열에 포함된 %d 자리에 정수형 변수 A의 값이 삽입되며 %s 자리에 문자열 변수 S의 값이 삽입된다. A가 26, S가 '홍길동'이라면 이 문자열은 'My age is 26,my name is 홍길동'이 된다.



서식 문자열 내의 서식은 인수 리스트의 변수들과 순서대로 대응되며 서식의 개수와 인수의 개수가 일치해야 할뿐만 아니라 대응되는 서식과 변수의 타입도 일치해야 한다. 만약 개수나 타입이 일치하지 않을 경우 컴파일은 가능하지만 실행중에 에러가 발생한다.

폭 지정이란 변수값이 문자열 내부에서 차지하는 자리수를 지정하며 생략시는 변수값의 길이만큼만 자리를 차지한다. 강제로 자리를 늘리고자 할 경우 서식과 타입사이에 폭을 정수로 지정한다. 예를 들어 %5d라고 하면 이 서식에 대입되는 정수가 5자리가 안되더라도 5자리를 강제로 차지한다. 단 여기서 지정하는 폭 지정은 최소한의 폭을 의미할 뿐이며 폭이 값의 실제 자리보다 작더라도 강제로 폭에 맞추지는 않는다. 예를 들어 I가 12345라는 다섯 자리 값을 가지는데 %2d와 대응될 경우 폭 지정 2를 지키기 위해 45만 출력하지 않으며 이 경우는 폭 지정이 무시된다.

정밀도 지정이란 실수에 사용되며 소수점 이하 몇 자리까지를 문자열로 바꿀 것인가를 지정한다. %10.5f 서식은 10자리를 차지하며 소수점 이하 5자리까지 문자열로 출력한다.

좌측 정렬 지정은 폭이 변수값보다 길 경우 공백을 우측에 배치하도록 하며 폭 지정 앞에 '-'기호를 붙여 표현한다. %5d에 대응되는 정수값이 12일 경우 출력은 '□□□12'가 되지만 서식을 %-5d'로 변경하면 '12□□□'가 된다. 단 여기서 □표시는 공백을 의미한다.

인덱스 지정자는 인수 리스트의 몇 번째 인수를 사용할 것인가를 지정하며 정수 하나와 :을 사용한다. 인덱스 지정자가 없으면 인수 리스트에 나타나는 순서대로 서식과 대응되지만 인덱스를 밝힘으로써 중복되는 인수 리스트를 재사용할 수 있다. 예를 들어 '%d%d%d',1,2,3은 앞쪽 서식부터 순서대로 정수 1,2,3에 대응되지만 '%d%d%0:d'는 앞쪽 둘은 순서대로 1,2에 대응되고 세 번째 서식은 인덱스 지정에 의해 첫 번째 인수 1에 대응된다.

타입 문자는 다음과 같다. C의 printf문에서 사용하는 타입 문자와 거의 동일하되 종류가 조금 적다.

문자	의미
d	정수값 하나에 대응된다. 대응되는 변수는 반드시 정수이어야 한다.
e	실수값을 공학적 표기법, 즉 부동 소수점 형식으로 변환한다.
f	실수값을 부동 소수점 형식으로 변환한다.
g	실수값과 대응하되 고정 소수점 형식과 부동 소수점 형식 중 길이가 짧은 쪽으로 변환한다.
n	실수값에 대응되며 천단위로 콤마를 삽입해준다.
p	포인터 값에 대응된다. 출력되는 문자열은 SSSS:0000 형태의 세그먼트, 오프셋 형태를 띄며 이때 세그먼트, 오프셋은 모두 네 자리의 16진수를 사용한다.
m	실수값에 대응되며 화폐 형태의 문자열로 변환한다.
s	문자열 변수, 문자 변수에 대응된다.
x	정수에 대응되며 16진수 형태의 문자열로 변환된다.

타입 문자들은 대문자와 소문자를 구분하지 않는다. 서식에 삽입되는 폭 지정, 정밀도 지정대신 *를 사용하면 이 위치의 값을 인수 리스트로 부터 읽는다. 예를 들어

```
Format('%*d',[5,128]);
```

은 *자리에 첫 번째 인수 5를 읽어 값을 취하며 이 값이 폭 지정이 된다. 그래서 위 Format문은 다음과 동일하다.

```
Format('%5d',[128]);
```

서식 내에 *를 사용할 경우는 서식 내의 폭, 정밀도 지정까지도 인수 리스트의 변수값으로 지정할 수 있다.

다음 예는 Format 함수를 사용하여 문자열 내에 정수, 실수값을 삽입한 것이다.

```
var
  I,J:integer;
  F:double;
  st:string;
begin
I:=34;
j:=128;
F:=3.14159265;
st:=Format('I is %5d,J is %-7d,F is %8.6F',[I,J,F]);
label1.Caption:=st;
end;
```

실행 결과 레이블로 출력되는 문자열은 다음과 같다.

I is 34,J is 128 ,F is 3.141593

이 예제의 서식과 폭, 정밀도, 좌측 정렬을 다양하게 변경해 보면 서식을 어떻게 사용하는지 알 수 있을 것이다.

FormatBuf SysUtils Unit

문법 : procedure FormatBuf(var Buffer; BufLen: Word; const Format; FmtLen: Word; const Args: array of const): Word;

서식에 의해 문자열을 작성한다. 자세한 문법은 Format 함수와 같으므로 참고하기 바란다. Format과 이 함수와의 차이점은 버퍼의 길이가 지정되어 있다는 점이다.

FormatDateTime SysUtils Unit

문법 : function FormatDateTime(const Format: string; DateTime: TDateTime): string;

TDateTime형의 변수에 저장된 시간과 날짜 정보를 특정한 형태의 문자열 포맷으로 변환한다. 문자열로 출력되는 형식은 다음과 같은 지정자에 의해 결정되며 각 지정자는 한 칸 이상의 공백으로 구분되어야 한다. 포맷을 지정하는 Format인수에는 지정자와 일반 문자열들이 올 수 있으며 일반 문자열들은 겹 따옴표로 싸 주어야 한다. 지정자의 대소문자는 구분하지 않는다.

지정자	의미
c	전역변수 ShortDateFormat, LongTimeFormat이 지정하는 형식대로 출력되며 지정자가 생략될 경우의 디폴트 지정자이다.
d	날짜값을 출력하되 선행 제로를 생략한다. (1~31)
dd	날짜값을 출력하되 선행 제로를 출력한다. (01~31)
ddd	요일을 생략형으로 출력한다. Sun, Mon, Sat 등과 같이 출력된다. 이 생략형 요일 이름은 ShortDayNames 변수에 의해 지정된다.
dddd	요일 이름을 완전한 형식으로 출력한다. Sunday, Monday 등과 같이 출력된다. 완전한 요일 이름은 LondDayNames 변수에 의해 지정된다.
ddddd	ShortDateFormat 전역 변수가 지정하는 형식으로 날짜가 출력된다.
dddddd	LongDateFormat 전역 변수가 지정하는 형식으로 날짜가 출력된다.
m	월을 출력하되 선행 제로를 생략한다. (1~12)
mm	월을 출력하되 선행 제로를 출력한다. (01~12)
mmm	월 이름을 생략형으로 출력한다. Jan, Dec 등과 같이 출력된다. 이 생략형 월 이름은 ShortMonthName 변수에 의해 지정된다.
mmmm	월 이름은 완전한 형식으로 출력한다. January, December 등과 같이 출력된다. 완전한 월 이름은 LongMonthName 변수에 의해 지정된다.
yy	년도를 두자리 수자로 출력한다. (00~99)
yyyy	년도를 네자리 숫자로 출력한다. (0000~9999)
h	시간을 출력하되 선행 제로를 생략한다. (1~12)
hh	시간을 출력하되 선행 제로를 출력한다. (01~12)
n	분을 출력하되 선행 제로를 생략한다. (1~59)
nn	분을 출력하되 선행 제로를 출력한다. (01~59)
s	초를 출력하되 선행 제로를 생략한다. (1~59)
ss	초를 출력하되 선행 제로를 출력한다. (01~59)
t	시간값을 ShortTimeFormat 전역 변수가 지정하는 형식으로 출력한다.
tt	시간값을 LongTimeFormat 전역 변수가 지정하는 형식으로 출력한다.
am/pm	오전과 오후를 출력하며 h나 hh 지정자의 시간을 12시간제로 출력한다. 오전이면 am이 출력되며 오후면 pm이 출력된다. 대문자 소문자 모두 쓸 수 있으며 혼합해서 쓸 수도 있다. 출력되는 결과는 이 지정자의 대소문자 구성을 따른다. 즉 Am/Pm 지정자를 쓰면 Am이나 Pm이 출력되며 AM/PM 지정자를 쓰면 AM이나 PM이 출력된다.
a/p	오전과 오후를 출력하며 h나 hh 지정자의 시간을 12시간제로 출력한다. 오전이면 a가 출력되며 오후면 p가 출력된다. 대문자 소문자 모두 쓸 수 있으며 혼합해서 쓸 수도 있다. 출력되는 결과는 이 지정자의 대소문자 구성을 따른다.
ampm	12시간제로 시간을 출력하며 오전일 경우 전역 변수 TimeAmString에 저장된 문자열을 출력하며 오후일 경우 TimePmString에 저장된 문자열을 출력한다.
/	DateSeparatoer 지정자가 지정하는 날짜 구분 기호를 출력한다.
:	TimeSeparatoer 지정자가 지정하는 시간 구분 기호를 출력한다.

출력 예를 보이면 다음과 같다.

FormatDateTime('dddd mmmm d yyyy hh:mm AM/PM', Now);
Friday November 24 1995 12:25 PM

FormatDateTime('"Now Time is" ddd mmm d yyyy hh:mm AM/PM', Now);
Now Time is Fri Nov 24 1995 12:26 PM

다양한 출력 포맷을 변경하여 직접 실습해 보기 바란다.

FormatFloat* SysUtils Unit

문법 : function FormatFloat(const Format: string; Value: Extended): string;

인수로 주어진 Value 실수값을 Format이 지정하는 포맷 형식에 따라 문자열로 변환한다.

Frac system Unit

문법 : function Frac(X:Real):Real;

실수 x의 소수부를 구한다. Frac(X)는 X-Int(X)와 같은 결과를 반납한다. 예를 들어 Frac(3.1415)는 정수부 3의 값을 빼고 남은 소수부 0.1415의 값을 계산한다.

Free system Unit

문법 : procedure Free;

이 프로시저를 호출한 인스턴스가 nil인지를 점검해 보고 nil이 아닐 경우 Destroy를 호출하여 객체를 메모리에서 해제한다. 인스턴스가 nil일 경우 Free 호출은 무시된다.

FreeMem system Unit

문법 : procedure FreeMem(var P:Pointer;Size:Word);

GetMem 프로시저에 의해 동적으로 할당된 메모리를 해제한다. size 인수는 할당 해제할 메모리의 크기이며 이 크기는 GetMem 프로시저에 의해 할당된 크기와 일치해야 한다. 다음 예는 정수형 변수를 위한 메모리를 할당한 후 해제한다.

```
var
  p: pointer;
begin
 { 메모리를 할당한다. }
  GetMem(p, SizeOf(integer));
 { 할당한 메모리를 사용한다.}
 { 할당을 해제한다. }
  FreeMem(p, SizeOf(integer));
end;
```

GetDir system Unit

문법 : procedure GetDir(D:Byte;var S:String);

지정한 드라이브의 현재 디렉토리를 조사한다. D로 드라이브의 번호를 지정하면 현재 디렉토리가 문자열 S에 대입된다. 드라이브 번호 D는 0일 경우, 디폴트 드라이브 1일 경우 A, 2일 경우 B 등과 같이 지정한다. 이 프로시저는 일체의 에러 체크를 하지 않는다. 만약 드라이브 번호로 주어진 D가 존재하지 않을 경우 조사되는 디렉토리는 X:\가 된다. 다음 예는 현재 드라이브의 디렉토리를 조사하여 레이블에 출력한다.

```
var
  dir:string;
begin
  GetDir(0,dir);
  label1.caption:=dir;
end;
```

GetMem system Unit

문법 : procedure GetMem(var P:Pointer; Size:Word);

동적으로 메모리를 할당하며 할당된 번지를 포인터형의 변수 P에 대입한다. Size는 동적으로 할당할 메모리의 크기를 바이트 수로 나타낸다. 할당된 메모리의 변수는 P^로 읽거나 쓸 수 있다. 다음 예는 정수형 변수를 위한 메모리를 할당한 후 사용하며 사용 후 FreeMem으로 할당을 해제한다.

```
var
  p: pointer;
begin
  GetMem(p, SizeOf(integer));
  integer(P^):=2;
  label1.caption:=IntToStr(integer(P^));
  FreeMem(p, SizeOf(integer));
end;
```

GotoXY WinCrt Unit

문법 : procedure GotoXY(X,Y:Byte);

가상 화면의 (X,Y) 좌표로 커서를 옮긴다. 가상 화면의 좌상단 원점의 좌표는 (1,1)이다. GotoXY를 사용하는 것보다는 CursorTo를 사용하는 것이 더 바람직하다.

Halt system Unit

문법 : procedure Halt [(ExitCode:Word)];

프로그램을 강제로 종료하고 운영체제로 제어권을 넘긴다. Exitcode는 프로그램의 탈출 코드이다.

Hi system Unit

문법 : function Hi(X):Byte;

상위 바이트 값을 부호없는 8비트의 정수로 읽어낸다. 인수로 주어진 X는 Integer 또는 Word형의 16비트 정수값이다.

```
var
  B: Byte;
begin
```

```
 B := Hi($1234);  { $12가 대입된다. }
end.
```

High

system Unit

문법 : function High(X);

인수로 주어진 X의 값 중 가장 큰값을 찾는다. X가 integer형이라면 결과는 32767이 되며 LongInt형이라면 결과는 2147483647이 된다. 결과값은 인수로 주어진 X와 동일한 형이다. 다음 예는 열거형 중 가장 큰 값을 찾는다. 레이블로 출력되는 값은 6이다.

```
type
  Day = (Sun, Mon, Tue, Wed, Thu, Fri, Sat);
var
  i:integer;
begin
  i:=integer(High(Day));
  label1.caption:=IntToStr(i);
end;
```

IdentToColor

Graphics Unit

문법 : function IdentToColor(const Ident: string; var Color: Longint): Boolean;

색상 명칭 문자열을 32비트로 표현된 Color 색상값으로 변경한다. Ident 문자열이 파스칼이 사용하는 색상 명칭이 아닐 경우 Color 인수는 변경되지 않으며 False를 리턴한다.

```
ans:=IdentToColor('clYellow',Col);
```

Inc

system Unit

문법 : procedure Inc(var X [; N:Longint]);

변수의 값을 1 증가시킨다. Inc(X)는 X:=X+1과 동일한 동작을 한다. 두 번째 인수를 사용하면 1 이상의 값을 증가시킬 수도 있다. Inc(X,5)는 X:=X+5와 동일한 동작을 한다. 첫 번째 인수는 서수형의 변수이며 확장 문법이 적용될 경우 PChar형도 가능하다. N은 정수형의 상수 또는 변수여야 한다. Inc 함수는 가장 최적화된 코드를 생성해 내므로 루프 내부에서 사용하기에 적합하다.

Include

system Unit

문법 : procedure Include(var S:Set of T; I:T);

집합형 S에 요소 I를 추가한다. Include(S,I)는 S := S + (I)와 동일한 동작을 하지만 좀 더 효율적인 코드를 생성해낸다.

InitWinCrt

WinCrt Unit

문법 : procedure InitWinCrt;

가상의 CRT 윈도우를 만든다. CRT에 연결된 파일 변수에 Read, Writeln 등으로 데이터를 입출력하면 이 프로시저는 자동으로 호출된다. 생성되는 윈도우의 캡션은 WindowTitle 변수에 의해 설정된다.

```
begin
StrCopy(WindowTitle,'가상의 CRT 윈도우');
InitWinCRT;
writeln('hello');
end;
```

이 예에 의해 다음과 같은 윈도우가 생성되고 문자열이 윈도우로 출력된다.

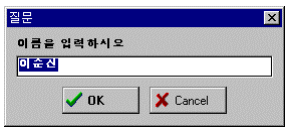
InputBox

Dialogs Unit

문법 : function InputBox(const ACaption, APrompt, ADefault: string): string;

문자열을 입력할 수 있는 대화상자를 보여주고 사용자에게 문자열을 입력받아 리턴한다. 인수의 의미는 인수의 이름을 참조하기 바라며 다음에 간단한 예를 보인다.

```
var
  str:string;
begin
str:=InputBox('질문','이름을 입력하시요','이순신');
end;
```



사용자가 문자열 입력 후 Cancel 버튼을 누르면 디폴트 문자열이 리턴되며 OK를 누르면 에디트 박스에 입력한 문자열이 입력된다.

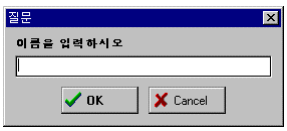
InputQeury

Dialogs Unit

문법 : function InputQuery(const ACaption, APrompt: string; var Value: string): Boolean;

문자열을 입력할 수 있는 대화상자를 보여주고 사용자에게 문자열을 입력받아 리턴한다. 대화상자가 열리면 아무것도 입력되어 있지 않은 빈 에디트를 보여주며 이 에디트에 사용자가 문자열을 입력한다. 입력된 결과는 리턴값으로 전달되는 것이 아니라 참조 호출로 전달된 value로 전달되며 리턴값은 사용자가 누른 버튼의 종류이다. 다음에 예를 보인다.

```
var
  str:string;
  ans:Boolean;
begin
ans:=InputQuery('질문','이름을 입력하시오',str);
end;
```



사용자가 대화상자에서 어떤 버튼을 눌렀는가를 꼭 알아야 할 경우는 InputQuery를 사용하며 단순히 문자열만 입력받고자 할 경우는 InputBox를 사용한다.

Insert

system Unit

문법 : procedure Insert(Source:String;var S:String;Index:Integer);

한 문자열의 중간에 다른 문자열을 삽입한다. 삽입되는 위치는 Index 인수가 지정하는 위치이다. 문자열을 삽입한 후의 문자열 길이가 255자를 넘을 경우 255문자 이후의 문자는 잘려진다. Dest 문자열이 'Orange'이고 Src 문자열이 'Apple'인 경우 Insert(Src,Dest,3);는 Dest 문자열의 3번째 문자인 a위치에 Src 문자열을 삽입하며 Dest 문자열은 'OrAppleange'가 된다.

```
var
  S: String;
begin
  S := 'abcdefghijkl';
  Insert(' hotdog ', S, 5);
  label1.caption:=S;
end;
```

이 예를 실행시키면 레이블로 abcd hotdog efghijkl 문자열이 출력된다.

Int

system Unit

문법 : function Int(X:Real):Real;

실수 X의 정수부를 구한다. 소수부는 반올림되지 않으며 버려진다. Int(3.14)는 3의 값을 반납한다. 주의할 것은 Int 함수의 리턴값은 정수값을 담지만 데이터형은 여전히 실수형이라는 점이다.

```
R:=int(3.1415);    {R은 3이 된다.}
```

IntToHex

SysUtils Unit

문법 : function IntToHex(Value: Longint; Digits: Integer): string;

10진 정수값을 16진수 형태의 문자열로 바꾸어 문자열을 리턴한다. 예를 들어 IntToHex(100,2) 명령에 의해 십진수 100을 16진수로 바꾼 문자열 '64'가 리턴된다. Digits 인수는 만들어질 16진수의 자리수를 지정한다. 그러나 이 값은 최소한의 자리수를 지정할 뿐이며 자리수보다 16진수의 크기가 더 많을 경우는 이 인수의 값이 무시된다. 즉 IntToHex(100,3)에 의해 최소한 세 자리를 할당하여 '064'가 리턴되지만 IntToHex(100,1)로 한다고 하여 두 자리의 16진수를 강제로 한 자리 숫자로 만들지는 않는다. 다음 예는 십진수 1234를 4자리수의 16진수 04D2로 바꾸어 레이블로 출력한다.

```
label1.caption:=IntToHex(1234,4);
```

IntToStr

SysUtils Unit

문법 : function IntToStr(Value: Longint): string;

정수를 문자열로 바꾸어 리턴한다. 정수값을 레이블이나 에디트 박스에 출력하고자 할 경우 직접 그 값을 출력할 수는 없다. 왜냐하면 레이블의 Caption 속성이나 에디트의 Text 속성은 문자열형이므로 정수형 값을 대입받을 수 없기 때문이다. 정수값을 문자열로 바꾸고자 할 경우에는 이 함수를 사용한다. 다음은 Label1에 정수형 변수 Age의 값을 출력한 예이다.

```
Label1.Caption:=IntToStr(Age);
```

IOResult

system Unit

문법 : function IOResult:Integer;

마지막 입출력 동작에서 발생한 상태값을 조사한다. 이 함수를 사용하여 입출력 에러를 검사하려면 {$I-}옵션을 반드시 꺼 주어야 한다. IOResult 함수를 호출하면 내부적인 에러값은 리셋된다. 이 함수가 리턴하는 값이 0이면 에러가 없는 것이다.

IsValidIdent SysUtils Unit

문법 : function IsValidIdent(const Ident: string): Boolean;

문자열의 내용이 명칭으로서 유효한가 아닌가를 점검해준다. 명칭은 알파벳, 숫자, 밑줄로 구성되며 첫 자에 숫자가 올 수 없다. 명칭으로서 유효한 문자열이면 True를 리턴하며 명칭으로서 유효하지 않은 문자가 있으면 False를 리턴한다.

KeyPressed system Unit

문법 : function KeyPressed:Boolean;

키보드에서 키가 눌러지면 True를 리턴하며 키가 눌러져있지 않으면 False를 리턴한다. ReadKey 함수로 키값을 읽기 전에 키가 눌러져 있는지 점검해 볼 때 사용한다.

Length system Unit

문법 : function Length(S:String):Integer;

주어진 문자열의 길이를 구한다. Length('Kora')는 5의 값을 리턴한다.

Ln system Unit

문법 : function Ln(X:Real):Real;

자연 로그값을 구한다.

Lo system Unit

문법 : function Lo(X):Byte;

하위 바이트값을 부호없는 8비트의 정수로 읽어낸다. 인수로 주어진 X는 Integer 또는 Word형의 16비트 정수값이다.

```
var
  B: Byte;
begin
  B := Lo($1234);  { $34가 대입된다. }
end.
```

LoadStr SysUtils Unit

문법 : function LoadStr(Ident: Word): string;

실행 파일에서 인수 Ident가 지정하는 문자열 리소스를 읽어들인다. 문자열 리소스가 정의되어 있지 않으면 빈 문자열이 리턴된다. 문자열을 리소스로 분리시키면 프로그램의 유지 및 보수에 많은 이점이 있다.

Low system Unit

문법 : function Low(X);

인수로 주어진 X의 값 중 가장 작은값을 찾는다. X가 integer 형이라면 결과는 -32768이 되며 LongInt형이라면 결과는 -2147483648이 된다. 결과값은 인수로 주어진 X와 동일한 형이다. 다음 예는 열거형 중 가장 작은 값을 찾는다. 레이블로 출력되는 값은 0이다.

```
type
  Day = (Sun, Mon, Tue, Wed, Thu, Fri, Sat);
var
  i:integer;
begin
  i:=integer(Low(Day));
  label1.caption:=IntToStr(i);
end;
```

LowerCase SysUtils Unit

문법 : function LowerCase(const S: string): string;

문자열 내부의 대문자를 모두 소문자로 바꾼다. 문자 변환은 7비트의 아스키 코드 내에서 이루어지며 문자 'A'~'Z'까지가 변환의 영향을 받는다. 나머지 기호나 숫자, 한글은 변환되지 않는다.

MaxAvail system Unit

문법 : function MaxAvail:Longint;

힙에서 사용 가능한 메모리 블럭 중 연속적으로 배치되어 있는 가장 큰 메모리의 크기를 조사한다. 이 함수가 조사하는 크기는 한 번에 생성할 수 있는 가장 크기가 큰 동적 변수의 크기가 된다. 즉 한 번에 할당할 수 있는 메모리 크기이다. 이 양은 남아 있는 메모리양과는 의미가 다르다. 설사 남아 있는 메모리의 총량이 500이더라도 제일 큰 덩어리가 100밖에 되지 않으면 크기 200의 메모리를 할당할 수 없다. 왜냐하면 하나의 변수는 연속된 메모리에 있어야 하기 때문이다. 그래서 레코드나 배열 등의 크기가 큰 메모리를 할당할 때는 이 함수로 사용 가능한 연속된 블럭의 크기를 먼저 조사해 보아야 하며 메모리가 없을 경우는 에러 메시지를 출력해 주어야 한다.

```
 if MaxAvail < SizeOf(변수형) then
   MessageDlg('메모리가 없네요',
   mtWarning, [mbOk], 0);
 else
{메모리를 할당한다.}
```

MemAvail system Unit

문법 : function MemAvail:Longint;

힙의 미사용 메모리, 즉 사용 가능한 메모리 총량을 계산한

다. 사용 가능한 총 메모리 용량은 한번에 사용 가능한 메모리 양과는 다르다. 한 번에 사용할 수 있는 가장 큰 메모리 크기는 MaxAvail 함수로 구한다.

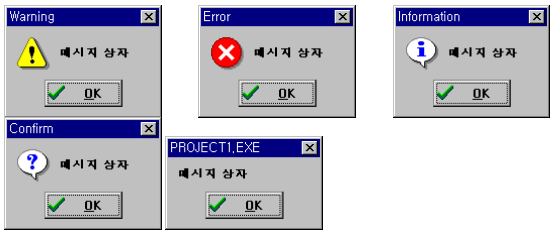
MessageDlg Dialogs Unit

문법 : function MessageDlg(const Msg: string; AType: TMsgDlgType; AButtons: TMsgDlgButtons; HelpCtx: Longint): Word;

메시를 전달하는 대화상자를 보여주며 사용자로부터 응답을 받아들여 리턴한다. Msg 문자열이 메시지의 내용이며 AType에 의해 메시지 상자의 형태와 캡션에 출력된 문자열이 결정된다.

값	형태
mtWarning	노란색 느낌표 비트맵이 나타난다.
mtError	빨간색의 엑스 비트맵이 나타난다.
mtInformation	파란색의 i자 비트맵이 나타난다.
mtConfirmation	물음표 비트맵이 나타난다.
mtCustom	비트맵을 사용하지 않는다.

AType에 따른 메시지 상자의 모양은 다음과 같다.



AButton 인수는 메시지 상자에 나타날 버튼의 종류를 지정하며 여러 개의 버튼을 집합형으로 전달한다. 리턴값은 사용자가 대화상자에 나타난 버튼 중 어떤 버튼을 눌렀는가를 나타내는 값이다.

버튼	모양	리턴값
mbYes	Yes	mrYes
mbNo	No	mrNo
mbOK	OK	mrOk
mbCancel	Cancel	mrCancel
mbHelp	Help	
mdAbort	Abort	mrAbort
mbRetry	Retry	mrRetry
mbIgnore	Ignore	mrIgnore
mbAll	All	mrAll

버튼 이름을 일일이 지정해 주는 것이 번거로우면 미리 정의되어 있는 다음과 같은 집합명을 사용한다.

집합명	버튼
mbYesNoCancel	Yes, No, Cancel 버튼
mbOkCancel	Ok, Cancel 버튼
mbAbortRetryIgnore	Abort, Retry, Ignore 버튼

이 집합명은 그 자체가 집합형이므로 [ ] 괄호를 쓰지 않아도 된다.

MessageDlgPos Dialogs Unit

문법 : function MessageDlgPos(const Msg: string; AType: TMsgDlgType; AButtons: TMsgDlgButtons; HelpCtx: Longint; X, Y: Integer): Word;;

MessageDlg 함수와 동일한 동작을 하는 함수이다. 단 MessageDlg는 대화상자를 무조건 화면 중앙에 열어주지만 이 함수는 X,Y 인수를 사용하여 대화상자가 출력될 화면 좌표를 지정할 수 있다.

MkDir system Unit

문법 : procedure MkDir(S:String);

새로운 서브 디렉토리를 생성한다. 문자열 S에 생성하고자하는 서브 디렉토리의 경로를 준다. 디렉토리를 생성하는 규칙은 도스에서와 동일하다. 즉 같은 이름의 디렉토리나 파일이 있어서는 안된다.

Move system Unit

문법 : procedure Move(var Source, Dest;Count:Word);

Source에서 Dest로 Count 바이트만큼 메모리 단위로 복사를 수행하며 범위 체크는 하지 않는다. 두 변수가 같은 세그먼트 영역에 있을 때 설사 두 변수의 복사 대상 번지가 겹쳐 있더라도 Move는 원본을 파괴하지 않고 정상적인 복사를 수행한다. 그러나 세그먼트가 다를 경우는 원본이 덮여져 복사가 제대로 수행되지 않을 수도 있다. 그러나 델파이는 겹쳐진 영역에 변수를 생성하지 않으므로 이 문제는 그리 큰 문제는 아니다. 다만 직접 변수의 메모리를 지정했거나(비디오 램의 경우), 번지를 변경하였을 경우에는 이런 문제가 발생한다. 메모리 단위로 복사를 수행하므로 Source, Dest의 데이터 타입은 어떤 형이라도 상관없으며 심지어 두 데이터 타입이 달라도 복사는 수행된다. 다음 예제는 문자열 데이터를 Longint형의 데이터로 복사한다.

```
var
  A: array[1..4] of Char;
  B: Longint;
```

```
begin
  Move(A, B, SizeOf(A));
  label1.caption:=IntToStr(B);
end;
```

물론 문자열이 초기화되지 않았으므로 B에 대입되는 값은 이 경우 알 수 없는 쓰레기 값이다.

New

system Unit

문법 : procedure New(var P: Pointer);
function New(<pointer type>): Pointer;

함수와 프로시저 두 가지 형태가 있으며 포인터 변수를 곧바로 인수로 사용할 수도 있고 포인터 타입을 인수로 사용할 수도 있다. 동적 변수를 힙에 생성하며 생성된 변수를 포인트하는 포인터 값을 리턴한다. 생성된 변수는 P^에 의해 참조된다. 메모리가 부족할 경우는 실행시 에러를 발생시킨다. 동적으로 할당한 메모리는 사용 후에 반드시 Dispose로 할당을 해제해 주어야 한다. 다음 예제는 문자열 형의 변수를 동적으로 할당하여 사용한 후 할당 해제한다.

```
type
  Str = String[50];
 var
  P: ^Str;
begin
  New(P);
  P^ := 'dynamic string';
  label1.caption:=P^;
  Dispose(P);
end;
```

위 예제는 포인터 변수를 직접 인수로 사용하는 New 프로시저를 사용하였다. 다음과 같이 포인터 타입을 만든 후 이 타입을 인수로 사용하여 New 함수를 사용할 수도 있다. 두 예제의 실행 결과는 동일하다.

```
type
  Str = String[50];
  PStr = ^Str;
 var
  P: PStr;
begin
  P := New(PStr);
  P^ := 'dynamic string';
  label1.caption:=P^;
  Dispose(P);
end;
```

NewStr

SysUtils Unit

문법 : function NewStr(const S: string): PString;

문자열 S의 복사본을 힙에 생성하며 그 시작 번지를 리턴해 준다. NewStr 함수로 문자열을 할당했다면 DisposeStr 함수로 반드시 해제해 주어야 한다. 문자열을 힙에 생성한 후 길이를 바꾸는 일이 없어야 한다. 만약 길이가 길어지면 힙의 다른 변수를 파괴할 가능성이 있으며 길이가 짧아지면 완전한 해제가 불가능해진다.

Now

SysUtils Unit

문법 : function Now: TDateTime;

현재 날짜와 시간을 한꺼번에 구한다. 날짜와 시간은 TDateTime형이므로 곧바로 문자열로 출력할 수 없으며 DateTimeToStr 함수를 사용하여 문자열로 바꾸어 주어야 한다. Label1.Caption:=DateTimeToStr(Now) 명령에 의해 Lable1에 오늘 날짜와 시간이 출력된다.
StrToDate: 문자열을 날짜 형태로 바꾸어 TDateTime형의 변수에 저장해준다. 문자열에 저장된 날짜는 반드시 유효한 날짜 정보를 가지고 있어야 하며 만약 날짜가 무효할 경우 EConverError 예외를 발생시킨다. 문자열은 년, 월, 일을 나타내는 두 개 또는 세 개의 숫자를 담고 있어야 하며 각 요소는 DateSeparator 변수가 지정하는 구분문자(보통 슬래시 기호)에 의해 구분되어 있어야 한다. 년, 월, 일의 순서는 ShortDateTime 변수에 따라 달라지며 mm/dd/yy 또는 dd/mm/yy 또는 yy/mm/dd 중 하나가 된다. 만약 문자열에 두 개의 숫자만 있으면 이는 각각 월, 일의 정보로 인식되며 년은 올해의 년도가 된다. 년의 값은 0~99 또는 1900~1999 둘 다 가능하다.

Odd

system Unit

문법 : function Odd(X:Longint):Boolean;

인수로 주어진 정수가 홀수인지를 검사한다. 홀수이면 True를 리턴하고 짝수이면 False를 리턴한다. 다음 예는 i값이 홀수인지를 점검하여 결과를 레이블로 출력한다. 물론 i에 3을 대입한 후 점검하므로 이 예제의 결과는 당연히 홀수이다.

```
var
  i:Integer;
begin
i:=3;
if Odd(i) then
  label1.caption:='홀수입니다.'
else
  label1.caption:='홀수가 아닙니다.';
end;
```

Ofs — system Unit

문법 : function Ofs(X):Word;

인수로 주어진 X의 오프셋 번지를 구한다. X는 어떤 변수형도 가능하며 함수나 프로시저도 가능하다.

Ord — system Unit

문법 : function Ord(X):Longint;

순서형 값의 순서값을 조사한다. 다음 예제는 열거형의 열거요소 중 Soo가 몇 번째 값인가를 조사한다.

```
type
  Yoil = (Wol,Hwa,Soo,Mok,Gum,TTo,Il);
begin
label1.caption:='Soo member is '+IntToStr(Ord(Soo));
end;
```

ParamCount — system Unit

문법 : function ParamCount:Word;

명령행에서 프로그램으로 전달된 인수의 개수를 전달한다. 각 인수는 공백이나 탭으로 분리되어 있다. 인수가 전달되지 않을 경우 이 값은 0이다.

ParamStr — system Unit

문법 : function ParamStr(Index):String;

명령행으로 전달된 인수를 리턴하며 Index로 인수의 번호를 지정한다. 이 함수는 특히 실행 파일이 실행된 디렉토리 경로를 알고자 할 때 유용하며 ParamStr(0)로 디렉토리를 읽는다. 또는 인수를 받아들이는 프로그램일 경우는 인수를 읽을 때 이 함수를 사용한다.

Pi — system Unit

문법 : function Pi:Real;

원주율을 구한다. 원주율은 3.14159265358979323846264338327950288419716939937510582097494459230781640628620 8.....로 정의되어 있으며 이 값은 부동 소수점 연산 보조 프로세서가 있는지 없는지에 따라 정확도가 달라질 수 있다. 좀 더 정확한 원주율 값은 다음과 같다.

3.14159265358979323846264338327950288419716939937510582097494459230781640628620 8.....

Point — Classes Unit

문법 : function Point(AX, AY: Integer): TPoint;

인수로 전달된 X좌표와 Y좌표를 사용하여 TPoint형의 레코드를 생성한다. TPoint형은 화면 상의 한 점을 기억하는 레코드이며 다음과 같이 정의되어 있다.

```
TPoint = record
  X: Integer;
  Y: Integer;
end;
```

TPoint형의 인수를 요구하는 Polygon 등의 함수에 사용된다.

Pos — system Unit

문법 : function Pos(Substr:String; S:String);Byte;

문자열 내의 부분 문자열을 검색한다. 부분 문자열이 검색된 위치를 리턴해준다. 만약 부분 문자열이 발견되지 않으면 리턴값은 0이다.

Pred — system Unit

문법 : function Pred(X);

순서형 변수의 바로 앞 값을 읽는다.

Ptr — system Unit

문법 : function Ptr(Seg, Ofs:Word):Pointer;

세그먼트와 오프셋 값을 사용하여 포인터 값을 만들며 리턴되는 값은 Seg:Ofs 번지를 포인터하는 포인터이다. Ptr 함수로 만들어진 포인터를 사용하여 메모리 값을 읽으려면 다음과 같이 타입 캐스팅을 해 주어야 한다.

```
if Byte(Ptr(Seg0040, $49)^) = 7 then
  Writeln('Video mode = mono');
```

세그먼트와 오프셋을 사용하여 특정 메모리를 직접 읽어야 할 경우는 극히 드물지만 비디오 램이나 BIOS 데이터 영역을 읽을 때 가끔 사용한다.

Random — system Unit

문법 : function Random [(Range:Word)];

0보다 크거나 같고 Range보다 작은 범위에서 난수를 만든다. Range 인수가 생략될 경우는 0~1사이의 실수 난수가 생성된다.

Randomize — system Unit

문법 : procedure Randomize;

난수 루틴을 초기화한다. Random 함수는 예측 불가능한 난수를 만들기는 하지만 생성되는 난수가 매번 동일하므로

Randomize로 난수 생성기를 시스템 클럭을 사용하여 초기화 시켜주어야 완전한 난수를 발생시킨다.

Read

system Unit

문법 : 타입드 파일 function Read(F, V1 [,V2...Vn]);

텍스트 파일 function Read([var F:Text;]
V1 [,V2,....,Vn]);

타입드 파일일 경우 파일로부터 한 요소(예를 들어 레코드 하나)를 읽어들여 변수에 대입해 주며 텍스트 파일일 경우 파일로부터 변수값을 읽어들인다. 문자열을 읽을 경우 읽고난 후에 다음줄로 개행을 하지 않으며 읽은 문자열이 변수의 길이보다 길 경우 문자열이 잘려나간다. 정수나 실수를 읽을 때 일체의 공백은 무시되며 수치로 변경될 수 없는 문자가 있을 경우는 에러가 발생된다.

ReadBuf

WinCrt Unit

문법 : function ReadBuf(Buffer:PChar; Count: Word) :Word;

CRT 윈도우로부터 한 줄을 읽어들여 Buffer가 지정하는 메모리로 복사된다. Buffer의 길이는 Count 바이트이며 읽어들인 줄 끝에 EOL마크(#13, #10)이 자동으로 추가되므로 읽어들일 수 있는 최대 길이는 Count-2 문자가 된다. 리턴되는 값은 입력된 문자의 개수와 EOL, EOF 마크의 개수이다.

ReadKey

system Unit

문법 : function ReadBuf(Buffer:PChar;Count:Word):Word;

키보드로부터 문자값을 읽는다. 단 이 함수는 표준 ASCII 코드만 읽을 뿐 기능키나 커서 이동키 등의 확장키 코드는 읽지 못한다.

Readln

system Unit

문법 : procedure Readln ([var F;Text;] V1 [,V2,...Vn]);

파일로부터 문장 한 줄을 읽어 변수 V1이하에 대입하며 다음 문장으로 이동한다. Readln(F)와 같이 읽어들일 변수를 지정하지 않으면 다음 줄로 이동하기만 한다. 파일 변수가 생략될 경우는 표준 입력 장치인 키보드로부터 문자열을 입력 받아 V1이하의 변수에 대입한다. 인수없이 Readln만 사용되면 Enter 키가 입력될 때까지 대기한다. 다음 예제는 키보드로부터 문장을 입력받아 다시 레이블로 출력한다.

```
var
  s : String;
begin
Write('문장을 입력하시요: ');
Readln(s);
label1.caption:=s;
end;
```

ReAllocMem

SysUtils Unit

문법 : function ReAllocMem(P: Pointer; CurSize, NewSize: Cardinal): Pointer;

힙에 할당되어 있는 메모리 블럭을 재할당한다. 현재 크기와 변경하고자 하는 새로운 크기를 인수로 지정해준다. 현재 크기보다 더 큰 크기로 할당하고자 할 경우, 즉 할당된 메모리를 확장할 경우 확장되는 메모리 영역은 0으로 채워진다. 리턴되는 값은 크기가 조정된 메모리 블럭의 포인터이며 이 값은 원래의 포인터 값과는 항상 다르다.

Rect

Classes Unit

문법 : function Rect(ALeft, ATop, ARight, ABottom: Integer): TRect;

TRect형 구조체는 두 점의 좌표를 사용하여 사각형 영역을 정의한다. 좌상단, 우하단의 각 X,Y 좌표로도 표현할 수 있으며 TPoint형의 레코드 두 개로 표현할 수도 있다.

```
TRect = record
  case Integer of
    0: (Left, Top, Right, Bottom: Integer);
    1: (TopLeft, BottomRight: TPoint);
end;
```

Rect 함수는 네 개의 좌표를 입력받아 TRect형의 레코드를 만들면 TRect형의 인수를 사용하는 함수나 TRect형의 속성을 정의할 때 사용한다.

Rename

system Unit

문법 : procedure Rename(var F:Newname);

파일의 이름을 변경한다. 변경하고자 하는 이름을 문자열로 전달해 주면 된다. 파일명을 변경하려면 AssignFile 함수로 파일 핸들에 외부 파일명을 할당해 주고 이 함수를 호출하면 된다. 파일을 오픈할 필요는 없다.

Reset

system Unit

문법 : procedure Reset(var F [:File;RecSize:Word]);

파일을 읽기 전용으로 오픈한다. F는 AssignFile 함수로 외부 파일과 연결된 파일 핸들이며 RecSize는 F가 untyped 파일일 경우 한 번에 읽을 레코드의 크기를 지정한다. RecSize가 생략될 경우 이 값은 128바이트로 간주된다. 외부 파일이 존재하지 않을 경우는 에러가 발생하며 파일이 이미 열려져 있는 상태이면 파일을 닫은 후에 다시 오픈한다. 이 함수로 파일을 오픈한 후에 Readln 등의 함수로 파일로부터 데이터를 읽는

다.

Rewrite system Unit

문법 : procedure Rewrite(var F;File [;Recsize:Word]);

파일을 쓰기 전용으로 오픈한다. F는 AssignFile 함수로 외부 파일과 연결된 파일 핸들이며 RecSize는 F가 untyped 파일일 경우 한번에 출력할 레코드의 크기를 지정한다. RecSize가 생략될 경우 이 값은 128바이트로 간주된다. 파일이 없을 경우에는 파일을 생성하고 파일이 이미 존재할 경우는 원래의 파일을 지운 후 다시 만든다. 파일이 이미 열려져 있으면 파일을 닫은 후에 다시 생성하며 파일 위치(FP)는 파일의 선두에 맞추어진다. 이 함수로 파일을 오픈한 후에 Writeln 등의 함수로 데이터를 파일로 출력한다. 다음 예는 test.txt라는 간단한 파일을 만든 후 문자열 하나를 출력한다.

```
var
  F:TextFile
begin
  AssignFile(F, 'test.txt');
  Rewrite(F);
  Writeln(F, '출력할 내용');
  CloseFile(F);
end;
```

파일 입출력이 끝난 후에 파일 핸들은 반드시 CloseFile 프로시저로 닫아 주어야 한다.

RmDir system Unit

문법 : procedure RmDir(S: String);

서브 디렉토리를 삭제한다. 삭제할 디렉토리명을 문자열로 넘겨준다. 도스의 RM 명령과 거의 동일한 규칙을 사용하며 디렉토리가 비어있지 않은 경우, 현재 디렉토리인 경우, 루트 디렉토리인 경우는 디렉토리를 삭제할 수 없다.

Round system Unit

문법 : function Round(X:Real):Lonint);

실수값 X를 정수값으로 만든다. 즉 실수의 소수 부분을 버리고 정수 부분만 결과로 취한다. 이때 소수부 첫째 자리에서 반올림이 일어나며 반올림의 결과는 부호에 상관없이 일어난다. 반올림의 결과와 Longint형의 범위를 벗어날 경우는 실행시 에러가 발생한다. 다음 예제는 실수 3.1415를 반올림하여 정수로 만든다.

```
var
  F:double;
begin
  F:=3.1415;
  label1.caption:=IntToStr(Round(F));
end;
```

레이블로 출력되는 결과는 소수부가 잘려나간 3이다. 실수값을 3.6415로 변경하면 반올림에 의해 4가 출력된다.

RunError system Unit

문법 : procedure RunError [(Errorcode:Byte)];

실행시 에러를 발생시키고 프로그램을 종료한다. 치명적인 에러가 발생했을 경우 이 함수를 사용한다.

ScrollTo WinCrt Unit

문법 : procedure ScrollTo(X,Y:Integer);

CRT 윈도우의 X,Y 위치가 좌상단 좌표가 되도록 스크롤 한다.

Seek system Unit

문법 : procedure Seek(var F;N:Longint);

파일의 현재 위치(FP)를 N위치로 옮긴다. untyped 파일일 경우 N의 의미는 파일 선두에서의 바이트 단위 거리이지만 typed 파일일 경우 N은 N번째 요소의 위치를 의미한다. 예를 들어 레코드형 파일일 경우 N이 5이면 여섯 번째 레코드의 위치로 이동한다. 첫 번째 레코드의 번호가 0번이다. Seek(F, FileSize(F))는 파일의 끝으로 현재 위치를 이동시킨다.

SeekEof system Unit

문법 : function SeekEof[(var F:Text)]:Boolean;

파일의 현재 위치가 끝(EOF)인지 점검한다.

SeekEoln system Unit

문법 : function SeekEoln[(var F:Text)]:Boolean;

파일의 현재 위치가 줄의 끝(EOL)인지 점검한다.

Seg system Unit

문법 : function Seg(X):Word;

X의 세그먼트 값을 구한다. X는 정수, 실수, 문자열 등 어떠한 변수도 가능하며 오브젝트도 가능하다.

SelectDirectory FileCtrl Unit

문법 : function SelectDirectory(var Directory: string; Options: TSelectDirOpts; HelpCtx: Longint): Boolean;

디렉토리 선택 대화상자를 보여주고 사용자로 하여금 디렉토리를 선택하도록 한다. 드라이브 콤보 박스와 디렉토리 리스트 박스를 사용하여 디렉토리를 선택하며 파일 리스트 박스는 어떤 파일이 있는지만 보여준다. 디렉토리를 선택하는 것뿐만 아니라 에디트 박스에 디렉토리명을 입력하여 없는 디렉토리를 직접 생성할 수도 있다. 인수로 주어지는 Dierectory 문자열은 대화상자가 처음 열릴 때 선택될 디렉토리이며 사용자가 대화상자에서 선택한 디렉토리명이 이 인수로 리턴된다. 사용자가 OK 버튼을 누르면 True를 리턴하고 Cancel 버튼을 누르면 False를 리턴한다.

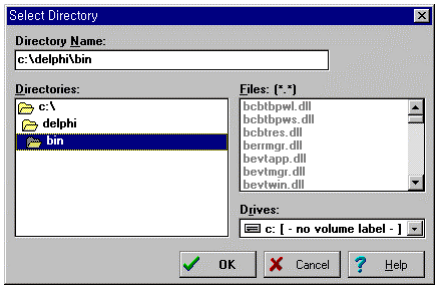
두 번째 인수 Options는 대화상자의 모양과 기능을 정의하는 옵션이며 다음과 같은 값의 집합형이다.

옵션	의미
[]	사용자는 존재하는 디렉토리를 선택할 수 있을 뿐이며 없는 디렉토리를 만들 수 없다.
sdAllowCreate	에디트 박스에 사용자가 직접 존재하지 않는 디렉토리를 입력하는 것을 허용한다. 그러나 이 옵션을 사용한다고 해서 디렉토리를 생성해 주는 것은 아니며 사용자가 직접 Directory 인수를 참조하여 생성해 주어야 한다.
sdPerformCreate	이 옵션을 sdAllowCreate 옵션과 함께 사용할 경우 존재하지 않는 디렉토리가 입력되면 직접 디렉토리를 만들어 준다.
sdPrompt	디렉토리를 새로 만들기 전에 사용자로부터 디렉토리를 정말로 만들 것인가를 물어본다. 위의 두 옵션이 선택되었을 때만 의미가 있는 옵션이다.

다음 예는 이 함수를 사용하는 간단한 예제이다.

```
var
  Dir: string;
begin
  Dir := 'C:\DELPHI';
  if SelectDirectory(Dir, [sdAllowCreate, sdPerformCreate, sdPrompt],0)
then
    Label1.Caption := Dir;
end;
```

디렉토리 선택 대화상자의 모습은 다음과 같다.



SetTextBuf

system Unit

문법 : procedure SetTextBuf(var F:Text;var Buf [;Size:Word]);

텍스트 파일의 입출력에 사용될 버퍼를 지정한다. 디폴트로 주어진 128바이트의 버퍼는 대부분의 경우는 충분하지만 입출력이 잦을 경우 버퍼를 늘려주면 속도를 조금 더 향상시킬 수 있다. 텍스트 파일 F가 사용하는 버퍼를 내부 버퍼를 사용하지 않고 Buf로 변경하며 Size 인수는 Buf의 크기를 지정한다. Size가 생략될 경우 이 값은 SizeOf(Buf)가 되며 이는 Buf가 가진 모든 버퍼를 다 사용한다는 뜻이다. 이 함수는 Reset, Rewrite 등의 오픈 함수가 호출된 직후에 사용한다. 그러나 일단 입출력을 수행한 파일에는 이 함수를 사용하여 버퍼를 변경하지 않는 것이 좋다. 왜냐하면 버퍼를 변경하면 기존 버퍼에 있던 내용을 잃어버리기 때문이다.

Sin

system Unit

문법 : function Sin(X;Real):Real;

삼각함수 Sin을 계산한다. 인수로 전달된 X는 360분법의 각도가 아닌 라디안 값이다.

SizeOf

system Unit

문법 : function SizeOf(X):Word;

X의 크기를 바이트 단위로 리턴한다. X는 변수이거나 데이터 타입이다. SizeOf(integer)는 2이며 SizeOf(double)은 8이다. 주로 레코드나 오브젝트 등 크기가 큰 데이터의 크기를 계산할 때 사용한다. 가상 메소드 테이블을 가진 오브젝트의 인스턴스 크기를 조사하면 VMT에 저장된 크기가 리턴된다.

SPtr

system Unit

문법 : function SPtr:Word

스택 세그먼트의 스택 포인터 값을 출력하며 이는 스택의 오프셋 값이다.

Sqr

system Unit

문법 : function Sqr(X:Real):Real;

X의 제곱값을 구한다. Sqr(X)는 X*X와 동일한 결과를 계산해 낸다.

Sqrt — system Unit

문법 : function Sqrt(X:Real):Real;

X의 제곱근을 구한다. Sqrt(2)는1.414213이다. 이때 계산되는 제곱근은 양의 제곱근이다.

SSeg — system Unit

문법 : function SSeg:Word;

스택 세그먼트 값(SS)을 리턴한다.

Str — system Unit

문법 : procedure Str(X [:Width [:Decimals]];var S);

숫자값 X를 문자열 S로 출력한다. 이때 출력되는 값의 폭은 Width 인수가 지정하는 폭을 사용하며 Decimals 인수가 지정하는 정밀도를 사용한다. 숫자값 X는 정수형이거나 실수형 모두 가능하다. 다음 예는 3.1415를 폭 5, 정밀도 3으로 문자열로 변환한다. 레이블로 출력되는 문자열은 3.142이다.

```
var
  F:double;
  S:string;
begin
F:=3.1415;
Str(F:5:3,S);
label1.caption:=S;
end;
```

StrAlloc — SysUtils Unit

문법 : function StrAlloc(Size: Word): PChar;

널 종료 문자열을 담기 위한 버퍼를 할당하며 버퍼의 시작 번지를 리턴한다. 버퍼의 크기는 인수로 주어진 Size 바이트이며 최대 문자열 길이는 종료 표시를 나타내는 NULL 값을 제외한 Size-1 문자이다. Size의 최대 크기는 65526까지 가능하다. StrAlloc로 문자열을 위해 할당한 메모리는 반드시 StrDispose로 할당 해제해 주어야 한다.

StrBufSize — SysUtils Unit

문법 : function StrBufSize(Str: PChar): Word;

StrAlloc에 의해 동적으로 할당된 버퍼의 크기를 구한다. 이 함수가 리턴하는 값을 널 종료 문자를 포함한 값이다. 만약 Str이 동적으로 할당된 메모리를 가리키고 있지 않을 경우 에러 메시지는 발생하지 않지만 결과는 예측할 수 없다.

StrCat — SysUtils Unit

문법 : function StrCat(Dest,Source:PChar):PChar;

두 개의 문자열을 연결한다. Source 문자열을 Dest 문자열에 추가하며 연결된 문자열을 리턴한다. 문자열의 길이 점검은 하지 않으므로 Dest 문자열이 Source의 문자열을 충분히 수용할만한 크기를 가지도록 해야 한다. 다음 예제는 두 개의 문자열을 결합한다. 레이블로 출력되는 결과는 made in Korea이다.

```
var
  S1:array [0..128] of Char;
  S2:array [0..128] of Char;
begin
StrCopy(S1,'made in ');
StrCopy(S2,'Korea');
StrCat(S1,S2);
label1.caption:=StrPas(S1);
end;
```

StrComp — SysUtils Unit

문법 : function StrIComp(Str1, Str2:PChar): Integer;

두 개의 문자열을 비교한다. S1과 S2 문자열이 같으면 0을 리턴하며 S1<S2이면 음수의 값을 리턴하며 S1>S2이면 양수를 리턴한다. 문자열끼리의 비교는 문자열을 이루는 각 문자의 문자 코드를 비교하여 수행된다.

StrCopy — SysUtils Unit

문법 : function StrCopy(Dest, Source: PChar): PChar;

널 종료 문자열을 복사한다. Source 문자열을 Dest 문자열에 복사하며 Dest 문자열을 리턴한다. 이 함수는 문자열의 길이 점검을 하지 않으므로 Dest 문자열의 길이가 Source 문자열의 길이보다 짧지 않도록 주의해야 한다. 만약 Dest 문자열의 길이가 Source 문자열을 충분히 수용하지 못할 경우 Dest 뒷부분의 다른 변수값이 파괴된다. 널 종료 문자열끼리는 :=연산자에 의해 직접 대입할 수 없으므로 이 함수를 사용하여 문자열끼리 복사해 주어야 한다. 예를 들어 Source가 'Korea'이고 이 값을 Dest 문자열로 대입하고 싶다고 해서 Dest:=Source와 같은 문장을 쓸 수 없다. 이때는 StrCopy(Dest,Source)와 같이 함수를 사용해야 한다.

StrDispose — SysUtils Unit

문법 : function StrDispose(Str: PChar;

StrAlloc에 의해 할당된 문자열을 위한 메모리를 해제한다. StrAlloc에 의해 할당된 메모리는 반드시 할당 해제되어야 한다.

StrEcopy SysUtils Unit

문법 : function StrECopy(Dest, Source: Pchar): PChar;

문자열끼리 복사하되 복사가 끝난 지점의 번지를 리턴해 준다. 따라서 이 함수를 연속적으로 사용하면 여러 개의 문자열을 한 문자열에 합할 수 있다. 이 함수는 문자열의 길이 점검을 하지 않으므로 Dest 문자열의 길이가 Source 문자열의 길이보다 작지 않아야 한다.

StrEnd SysUtils Unit

문법 : function StrEnd(Str: PChar): Pchar;

문자열의 끝 번지를 찾아준다. 문자열의 끝 번지란 곧 문자열의 끝을 나타내는 NULL 문자의 번지를 말한다.

StrFmt SysUtils Unit

문법 : function StrFmt(Buffer, Format: PChar; const Args: array of const): PChar;

문자열을 포맷하여 Buffer에 저장한다. 문자열 포맷에 관한 자세한 사항은 Format을 참조하기 바란다.

StrlComp SysUtils Unit

문법 : function StrLComp(Str1, Str2: PChar; MaxLen: Word): Integer;

두 개의 문자열을 비교하되 대문자와 소문자를 구분하지 않는다. S1과 S2 문자열이 같으면 0을 리턴하며 S1<S2이면 음수의 값을 리턴하며 S1>S2이면 양수를 리턴한다. 문자열끼리의 비교는 문자열을 이루는 각 문자의 문자 코드를 비교하여 수행된다.

StrLCat SysUtils Unit

문법 : function StrLCat(Dest, Source: PChar; MaxLen: Word): PChar;

StrCat와 마찬가지로 문자열을 연결한다. 차이점은 문자열을 연결한 결과 생성되는 문자열이 MaxLen이상의 길이가 되지 않도록 길이 점검을 해준다는 점이다.

StrLComp SysUtils Unit

문법 : function StrLIComp(Str1, Str2: PChar; MaxLen: Word): Integer;

StrComp 함수와 마찬가지로 두 개의 문자열을 비교한다. 차이점이라면 인수로 주어진 MaxLen 문자분까지만 비교한다는 점이다.

StrLCopy SysUtils Unit

문법 : function StrLCopy(Dest, Source: PChar; MaxLen: Cardinal): PChar;

StrCopy 함수와 마찬가지로 문자열을 복사한다. 차이점이라면 Dest 문자열의 길이를 점검하여 MaxLen이상의 문자가 복사되지 않도록 해준다는 점이다.

SetLen SysUtils Unit

문법 : function StrLen(Str: PChar): Cardinal;

주어진 문자열의 길이를 조사한다. 이 길이에는 널종료 문자열이 제외된다.

ShowMessage Dialogs Unit

문법 : procedure ShowMessage(const Msg: string);

대화상자를 열어 Msg 문자열을 보여주기만 하며 OK 버튼을 누르면 대화상자를 닫는다. 캡션에는 이 대화상자를 호출한 프로그램의 실행 파일 이름이 출력된다.

ShowMessage('지금부터 검색을 시작합니다.');

간단한 메시지만 전달하고 사용자의 입력을 받아야 할 필요가 없을 때 이 프로시저를 사용한다.

ShowMessagePos Dialogs Unit

문법 : procedure ShowMessagePos(const Msg: string; X, Y: Integer);

ShowMessage 프로시저와 동일한 동작을 하는 프로시저이다. 단 ShowMessage는 대화상자를 무조건 화면 중앙에 열어주지만 이 함수는 X,Y 인수를 사용하여 대화상자가 출력될 화면 좌표를 지정할 수 있다.

StringToColor Graphics Unit

문법 : function (S: string): TColor;

문자열로 전달된 색상의 명칭을 TColor형의 색상값으로 변경한다.

StrLFmt SysUtils Unit

문법 : function StrLFmt(Buffer: PChar; MaxLen: Word; Format: PChar; const Args: array of const): PChar;

문자열을 포맷한다. StrFmt 함수와 동일하되 버퍼의 길이를 지정한다는 점만 다르다.

StrLIComp SysUtils Unit

문법 : function StrLIComp(Str1, Str2: PChar; MaxLen: Word): Integer;

StrIComp 함수와 마찬가지로 대소문자 구분없이 두 개의 문자열을 비교한다. 차이점이라면 인수로 주어진 MaxLen 문자분까지만 비교한다는 점이다.

StrLower SysUtils Unit

문법 : function StrLower(Str: PChar): PChar;

주어진 문자열을 소문자로 바꾼다.

StrMove SysUtils Unit

문법 : function StrMove(Dest, Source: PChar; Count: Cardinal): PChar;

Source 문자열에서 Count 문자분만큼 Dest 문자열로 복사한다. 이 때 두 문자열의 복사 영역이 상호 겹치더라도 정확하게 복사를 수행한다.

StrNew SysUtils Unit

문법 : function StrNew(Str: PChar): PChar;;

Str 문자열의 복사본을 힙에 생성한다. Str이 NIL이거나 빈 문자열일 경우는 아무런 동작도 수행하지 않는다. 똑같은 문자열의 복사본을 만들어 준다.

StrPas SysUtils Unit

문법 : function StrPas(Str: PChar): String;

널종료 문자열을 파스칼형 문자열로 바꾸어주며 바꾸어진 파스칼형 문자열을 리턴한다. 델파이에서 기본적으로 사용하는 문자열의 형태는 파스칼형이다. 에디트 박스의 Text 속성, 각종 컴포넌트의 Caption 속성이 모두 파스칼형 문자열이다. 그러나 윈도우즈 API 함수에서 사용하는 함수는 널종료 문자열이어서 두 문자열 간의 대입이 불가능하다. API함수에서 얻은 널 종료 문자열을 컴포넌트의 Caption 속성에 대입하거나 파스칼형 문자열 변수에 대입하고자 할 경우는 반드시 이 함수를 사용하여 문자열 형식을 바꾸어 주어야 한다.

StrPCopy SysUtils Unit

문법 : function StrPCopy(Dest: PChar; Source: String): PChar;

파스칼형 문자열을 널 종료 문자열로 바꾸어 복사해준다. 델파이에서 기본적으로 사용하는 문자열의 형태는 파스칼 형이다. 에디트 박스의 Text 속성, 각종 컴포넌트의 Caption 속성이 모두 파스칼형 문자열이다. 그러나 윈도우즈 API 함수에서 사용하는 함수는 널종료 문자열이어서 두 문자열 간의 대입이 불가능하다. 컴포넌트의 Caption 속성이나 파스칼 형 문자열 변수의 문자열을 윈도우즈 API 함수에 사용하고자 할 경우는 반드시 이 함수를 사용하여 문자열 형식을 바꾸어 주어야 한다. 이 함수는 길이 점검을 하지 않으므로 Dest의 길이가 Source의 길이보다 짧지 않도록 유의해야 한다.

StrPLCopy SysUtils Unit

문법 : function StrPLCopy(Dest: PChar; const Source: string; MaxLen: Word): PChar;

StrPCopy 함수와 동일한 동작을 하되 복사되는 문자열의 길이가 MaxLen 문자분 이상이 되지 않도록 제한해준다.

StrPos SysUtils Unit

문법 : function StrPos(Str1, Str2: PChar): PChar;

널 종료 문자열인 Str1에서 부분 문자열 Str2를 찾아 그 번지를 리턴해준다. 만약 부분 문자열이 발견되지 않을 경우 NIL 값을 리턴해준다.

StrRScan SysUtils Unit

문법 : function StrRScan(Str: PChar; Chr: Char): PChar;

StrScan 함수와 마찬가지로 문자열에서 문자 하나를 찾아준다. 차이점이라면 StrScan 함수는 문자열의 좌측에서부터 문자를 찾아 나가지만 StrRScan은 문자열의 우측에서부터 문자를 찾아 나간다. 'Father and mother'라는 문자열에서 문자 a를 찾을 때 좌측에서 찾을 경우와 우측에서 찾을 경우 그 결과가 다르다.

StrScan SysUtils Unit

문법 : function StrScan(Str: PChar; Chr: Char): PChar;

널종료 문자열인 Str에서 문자 Chr을 찾아 그 번지를 리턴해준다. 만약 Chr 문자가 발견되지 않을 경우 NIL 값을 리턴해준다.

StrToDate SysUtils Unit

문법 : function StrToDate(const S: string): TDateTime;

문자열을 날짜 형식으로 바꾼다. 문자열은 날짜 형식으로 되어 있어야 한다. 즉 문자열 안에는 날짜를 이루는 세 가지 요소가 순서에 맞게 배치되어 있어야 하며 각 요소는 날짜 구분문자(/)로 분리되어 있어야 한다. 문자열에 날짜로 변경할 수 없는 문자가 있을 경우는 EConvertError 예외가 발생한다.

StrToDateTime SysUtils Unit

문법 : function StrToDateTime(const S: string): TDateTime;

문자열은 날짜와 시간을 담는 TDateTime형의 변수로 바꾼다. 문자열에는 MM/DD/YY HH:MM:SS 형태로 날짜와 시간이 담겨 있어야 하며 시간값은 24시간제로 표현하거나 AM, PM을 추가로 뒤에 붙여 주어야 한다.

StrToFloat SysUtils Unit

문법 : function StrToFloat(const S: string): Extended;

문자열의 실수를 실수값으로 변경한다. 문자열에는 부호를 나타내는 +/-와 숫자, 그리고 소수점만 있어야 하며 지수를 나타내는 E가 올 수 있다. 그 외의 문자가 있을 경우는 EConvertError 예외가 발생한다.

StrToInt SysUtils Unit

문법 : function StrToInt(const S: string): Longint;

문자열을 정수값으로 바꾼다. 이때 문자열에는 숫자로 바꿀 수 없는 문자가 있어서는 안되며 만약 무효한 문자가 있을 경우 EConvertError 예외를 발생시킨다. 에디트 박스에 사용자가 입력한 값을 정수형 변수에 대입하고자 할 때 에디트 박스의 Text 속성과 정수형 변수의 데이터형이 일치하지 않아 대입이 불가능 하다. 문자열에 입력된 정수를 정수형 값으로 바꾸어 주어야 정수형 변수에 대입할 수 있다. 다음은 Edit1에 입력한 정수값을 정수형 변수 Age에 대입하는 예이다.

```
Age:=StrToInt(Edit1.Text);
```

StrToIntDef SysUtils Unit

문법 : function StrToIntDef(const S: string; Default: Longint): Longint;

문자열을 정수로 변환한다. StrToInt 함수는 변환이 불가능할 경우 예외를 발생시키지만 이 함수는 변환이 불가능할 경우 Default 값을 사용한다.

StrToTime SysUtils Unit

문법 : function StrToTime(const S: string): TDateTime;

문자열을 시간 형태로 바꾸어 TDateTime형의 변수에 저장해 둔다. 문자열에는 HH:MM:SS 형태로 시, 분, 초가 담겨 있어야 하며 AM, PM은 있어도 되고 없어도 된다. 시간은 반드시 24시간 형태로 표시해 주어야 한다. 예를 들어 오후 8:26은 20:26으로 표현한다. 만약 문자열에 무효한 시간값이 들어 있을 경우 EConvertError 예외를 발생한다.

StrUpper SysUtils Unit

문법 : function StrUpper(Str: PChar): PChar;

주어진 문자열을 대문자로 바꾼다.

Succ system Unit

문법 : function Succ(X);

순서형 변수의 바로 다음 값을 읽는다.

Swap system Unit

문법 : function Swap(X);

X의 상위 워드와 하위 워드를 교체한다.

```
var
  X: Word;
begin
  X := Swap($1234);  { $3412가 된다. }
end.
```

TextToFloat SysUtils Unit

문법 : function TextToFloat(Buffer: PChar; var Value: Extended): Boolean;

Buffer에 문자열 형태로 보관된 실수를 실수값으로 변경한다. 변경에 성공하면 True를 리턴하고 실패하면 False를 리턴한다.

Time SysUtils Unit

문법 : function Time: TDateTime;

시스템의 시계를 참조하여 현재 시간을 구한다. 시간은 TDateTime형이므로 곧바로 문자열로 출력할 수 없으며 TimeToStr 함수를 사용하여 문자열로 바꾸어 주어야 한다. Label1.Caption:=TimeToStr(Time) 명령에 의해 Label1에 현재 시간이 출력된다.

TimeToStr system Unit

문법 : function TimeToStr(Time: TDateTime): String;

시간을 나타내는 Time을 문자열 형태로 변경한다. 시간값을 곧바로 레이블이나 에디트 등으로 출력할 수 없기 때문에 이 함수를 사용하여 문자열로 변경해 주어야 한다. 다음 예는 현재 시간을 레이블로 출력한다.

```
label1.caption:=TimeToStr(Time);
```

TrackCursor — WinCrt Unit

문법 : procedure TrackCursor;

커서가 보일 수 있도록 CRT윈도우를 스크롤시킨다.

Trunc — system Unit

문법 : function Trunc(X: Real): Longint;

실수의 소수점 이하를 버리고 정수부만을 취한다. Round 함수는 소수점 첫 째 자리에서 반올림을 하지만 이 함수는 반올림없이 무조건 소수점을 버림한다. 다음 예는 3.6415를 버림하여 3을 레이블로 출력한다.

```
var
  F:double;
begin
F:=3.6415;
label1.caption:=IntToStr(Trunc(F));
end;
```

Trunc 함수 대신 Round 함수를 사용한다면 반올림되어 4가 출력된다.

Truncate — system Unit

문법 : procedure Truncate(var F);

현재 파일 위치 이후의 모든 레코드를 잘라 버리며 파일 위치는 EOF가 된다. 텍스트 파일에서는 이 프로시저를 사용할 수 없다.

TypeOf — system Unit

문법 : function TypeOf(X) : Pointer;

X 오브젝트의 가상 메소드 테이블에 대한 포인터를 리턴한다. VMT를 가지지 않은 오브젝트에 이 함수는 사용할 수 없다.

UpCase — system Unit

문법 : function UpCase(Ch: Char): Char;

문자 Ch를 대문자로 바꾼다. 소문자 a에서 대문자 z까지의 문자가 대문자 A~Z로 변경되며 그 외 나머지 문자일 경우는 전혀 영향을 받지 않는다.

UpperCase — system Unit

문법 : fun;

문자열 내부의 소문자를 모두 대문자로 바꾼다. 문자 변환은 7비트의 아스키코드 내에서 이루어지며 문자 'A'~'Z'까지가 변환의 영향을 받는다. 나머지 기호나 숫자, 한글은 변환되지 않는다.

Val — system Unit

문법 : fun;

문자열 변수 S를 숫자형 변수로 바꾼다. 문자열 변수 S에는 숫자가 담겨 있어야 한다. 문자열을 바꾼 결과를 받기위한 인수 V는 정수형이거나 실수형 변수 모두 가능하다. S에 숫자로 바꿀 수 없는 무효 문자가 있을 경우 Code에는 무효 문자의 위치가 들어가며 만약 무효 문자가 없다면 Code 값은 0가 된다. 널 종료 문자열의 경우 Code에 리턴되는 에러 위치는 실제 문자의 위치보다 하나가 더 많은 값이 된다.

WhereX — WinCrt Unit

문법 : function WhereX: Byte;

현재 커서 위치의 X좌표를 구한다. CRT 윈도우의 좌표는 1부터 시작하므로 이 함수가 조사하는 값은 Cursor.X+1과 동일하다.

WhereY — WinCrt Unit

문법 : function WhereY: Byte

현재 커서 위치의 Y좌표를 구한다. CRT 윈도우의 좌표는 1부터 시작하므로 이 함수가 조사하는 값은 Cursor.Y+1과 동일하다.

Write — system Unit

문법 : 텍스트 파일 : procedure Write([var F: Text;] P1 [,P2,...,Pn]);
타입드 파일 : procedure Write(F, V1 [V2,...Vn]);

변수값을 파일로 출력한다. F는 출력용으로 오픈된 텍스트 파일이어야 하며 파일이 생략될 경우 표준 출력 장치인 화면으로 출력된다. 파일 변수 다음의 인수 P1이하는 파일로 실제 출력될 값들이며 이 값에 필드 폭, 정밀도를 지정할 수 있다. 출력할 수 있는 타입은 문자형, 정수형, 실수형, 문자열, 진위형 등이 있다.
타입드 파일일 경우 출력되는 변수 V1이하는 파일의 요소와 같은 형이어야 한다. 즉 레코드형 파일이라면 출력되는 값도 같은 형의 레코드이어야 한다. 파일의 현재 위치가 파일의 끝이라면 파일의 크기는 출력된 데이터의 길이만큼 늘어난다.

WriteBuf — WinCrt Unit

문법 : procedure WriteBuf(Buffer: PChar; Count: Word);

문자열을 CRT 윈도우로 출력한다. 출력되는 문자의 개수는 Count 인수로 지정한다.

WriteChar

WinCrt Unit

문법 : procedure WriteChar(Ch: Char);

CRT 윈도우의 현재 위치로 문자 하나를 출력한다. WriteBuf(@Ch, 1)와 같은 호출문이다.

Writeln

system Unit

문법 : procedure Writeln([var F: Text;] P1 [, P2, ...,Pn]);

Write 함수의 기능을 확장한 함수이다. Write를 호출하여 변수를 파일로 출력하되 변수 값 다음에 EOL을 출력하여 자동으로 개행되도록 한다.