CHAPTER _9

표준 루틴과 I/O

이 장에서는 텍스트 및 파일 I/O에 대해 설명하고 표준 라이브러리 루틴을 간단히 설명합니다.

■파일 입력 및 출력 ■텍스트 파일 장치 드라이버
■Null 종료 문자열의 처리 ■기타 표준 루틴

이 장에서 다루는 대부분의 프로시저 및 함수는 System 유닛과 SysInit 유닛에 정의되어 있으며, 이 두 유닛은 모든 애플리케이션에서 암시적으로 uses 됩니다. 컴파일러에서 내장되어 있지만 System 유닛에 있는 것처럼 처리되는 것들도 있습니다.
일부 표준 루틴은 SysUtils 같은 유닛에 정의되어 있습니다. 이 유닛을 프로그램에서 사용하려면 이 유닛을 uses 절에 추가해야 합니다. 그러나 System 유닛은 uses 절에 쓸 수 없으며, System 유닛을 수정하거나 명시적으로 다시 빌드하려고 시도해서는 안됩니다.
나열된 루틴에 대한 자세한 내용은 온라인 헬프를 참조하십시오.

파일 입력 및 출력

아래 표에서는 입력 및 출력 루틴들을 나열합니다.
파일 변수는 타입이 파일 타입인 변수입니다. 파일은 타입 지정 파일, 텍스트 파일, 타입 미

표 9.1 입력/출력 프로시저 및 함수

프로시저/함수	설명
Append	내용을 추가하기 위해 기존 텍스트 파일을 엽니다.
AssignFile	외부 파일의 이름을 파일 변수에 대입합니다.
BlockRead	타입 미지정 파일에서 하나 이상의 레코드를 읽습니다.
BlockWrite	타입 미지정 파일에 하나 이상의 레코드를 씁니다.
ChDir	현재 디렉토리를 변경합니다.
CloseFile	열려 있는 파일을 닫습니다.
Eof	파일의 파일 끝(end-of-file) 상태를 리턴합니다.
Eoln	텍스트 파일의 줄 끝(end-of-line) 상태를 리턴합니다.
Erase	외부 파일을 지웁니다.
FilePos	타입 지정 혹은 타입 미지정 파일의 현재 위치를 리턴합니다.
FileSize	파일의 현재 크기를 리턴합니다. 텍스트 파일에 대해서는 사용하지 않습니다.
Flush	출력 텍스트 파일의 버퍼를 플러시합니다.
GetDir	지정된 드라이브의 현재 디렉토리를 리턴합니다.
IOResult	마지막으로 수행된 I/O 함수의 상태인 정수값을 리턴합니다.
MkDir	하위 디렉토리를 만듭니다.
Read	파일에서 하나 이상의 값을 하나 이상의 변수로 읽어들입니다.
Readln	Read와 동일하지만 텍스트 파일의 한 줄만 읽어 들입니다.
Rename	외부 파일의 이름을 변경합니다.
Reset	존재하는 파일을 엽니다.
Rewrite	새 파일을 만들어 엽니다.
RmDir	빈 하위 디렉토리를 제거합니다.
Seek	타입 지정 파일이나 타입 미지정 파일의 현재 위치를 지정된 컴포넌트로 이동시킵니다. 텍스트 파일에서는 사용하지 않습니다.
SeekEof	텍스트 파일의 파일 끝(end-of-file) 상태를 리턴합니다.
SeekEoln	텍스트 파일의 줄 끝(end-of-line) 상태를 리턴합니다.
SetTextBuf	I/O 버퍼를 텍스트 파일에 할당합니다.
Truncate	타입 지정 파일이나 타입 미지정 파일을 현재 파일 위치에서 잘라냅니다.
Write	파일에 하나 이상의 값을 씁니다.
Writeln	Write와 동일하지만 텍스트 파일에 한 줄을 쓰고 줄 끝에 eoln(end-of-line)을 표시합니다.

지정 파일의 세 가지로 구분됩니다. 파일 타입 선언 문법은 4장의 "파일 타입"을 참고하십시오. 파일 타입은 Win32에서만 사용 가능하다는 것을 알아두십시오.

파일 변수를 사용하기 전에 AssignFile 프로시저 호출로 파일 변수를 외부 파일과 연결해야

합니다. 외부 파일은 일반적으로 이름을 가진 디스크 파일을 말하지만, 키보드나 디스플레이 같은 장치들도 포함됩니다. 외부 파일은 파일에 기록된 정보를 저장하거나 파일에서 읽은 정보를 제공합니다.

일단 파일 변수가 외부 파일과 연결되면, 파일 변수는 입력 또는 출력을 할 수 있도록 "열려야" 합니다. Reset 프로시저로 기존 파일을 열 수 있고, Rewrite 프로시저로 새 파일을 만들고 열 수 있습니다. 텍스트 파일일 경우, Reset으로 여는 파일은 읽기 전용이 되고 Rewrite나 Append로 연 쓰기 전용이 됩니다. 타입 지정 파일과 타입 미지정 파일의 경우에는 Reset 또는 Rewrite 중 어느 프로시저로 열었든 상관없이 항상 읽고 쓸 수 있습니다.

모든 파일은 구성요소들의 직선적인 연속이고, 각 구성요소들은 파일의 구성요소 타입(혹은 레코드 타입)을 가지고 있습니다. 구성요소의 순번은 0부터 시작됩니다.

일반적으로 파일은 순차적으로 액세스됩니다. 즉, 표준 프로시저 Read로 구성요소를 읽거나 표준 프로시저 Write로 구성요소를 쓰면, 현재 파일 위치는 다음 번호의 파일 구성요소로 이동합니다. 표준 프로시저 Seek를 사용하여 타입 지정 파일과 타입 미지정 파일을 임의 액세스(random access)할 수도 있습니다. Seek 함수는 현재 파일 위치를 지정된 구성요소로 이동시킵니다. 표준 함수 FilePos와 FileSize를 사용하면 현재 파일 위치와 파일 크기를 알아낼 수 있습니다.

프로그램이 파일 처리를 완료하면 표준 프로시저 CloseFile을 사용하여 해당 파일을 닫아야 합니다. 파일을 닫으면 연결된 외부 파일이 업데이트됩니다. 그런 다음 다른 외부 파일에 연결하기 위해 이전의 파일 변수를 사용할 수 있습니다.

기본적으로, 표준 I/O 프로시저 및 함수 호출에 대해서는 자동으로 에러를 검사하며, 에러가 발생하면 예외가 발생합니다. 예외를 처리할 수 없으면 프로그램이 종료됩니다. {$I+} 및 {$I-} 컴파일러 지시어를 사용하면 자동 검사 기능을 켜고 끌 수 있습니다. I/O 검사 기능이 꺼져 있으면, 즉 프로시저 또는 함수 호출이 {$I-} 상태에서 컴파일되었으면 I/O 에러로 예외가 발생되지 않습니다. I/O 동작의 결과를 검사하려면 표준 함수 IOResult를 대신 호출해야 합니다.

에러에 대해 알 필요가 없을 경우라도 에러를 지우려면 IOResult 함수를 호출해야 합니다. {$I+} 상태에서 에러를 지우지 않으면 다음 번 I/O 함수 호출은 IOResult 에러와 함께 실패합니다.

텍스트 파일

이 단원에서는 표준 타입 텍스트의 파일 변수를 사용하여 I/O를 요약 설명합니다.

텍스트 파일이 열리면 외부 파일은 특별한 방법으로 해석됩니다. 텍스트 파일은 줄 단위로

나뉘어진 문자들의 연속으로 간주됩니다. 여기서 각 줄은 줄 끝 표시(end-of-line; 캐리지 리턴 문자, 그 뒤에 라인 피드 문자가 따라올 수도 있음)로 끝납니다. Text 타입은 file of Char 타입과는 다릅니다.

텍스트 파일을 위해 Char 타입이 아닌 값을 읽고 쓸 수 있는 특별한 형태의 Read와 Write가 있습니다. 그런 값들은 자동으로 문자 표현으로 바뀌고 문자 표현에서 값으로 바뀝니다. 예를 들면, I가 Integer 타입 변수라면 Read(F, I)는 일련의 숫자들을 읽어서 이를 십진수 정수들의 연속으로 해석한 다음 I에 저장합니다.

Input과 Output이라는 두 개의 표준 텍스트 파일 변수가 있습니다. 표준 파일 변수 Input은 운영 체제의 표준 입력(일반적으로 키보드)에 연결된 읽기 전용 파일입니다. 표준 파일 변수 Output은 운영 체제의 표준 출력(일반적으로 디스플레이)에 연결된 출력 전용 파일입니다. Input과 Output은 애플리케이션이 실행되기 전에 자동으로 열립니다. 마치 다음 문장이 실행된 것과 같습니다.

```
AssignFile(Input, '');
Reset(Input);
AssignFile(Output, '');
Rewrite(Output);
```

Note

텍스트 I/O는 콘솔 애플리케이션에서만 사용 가능합니다. 여기서 콘솔 애플리케이션은 프로젝트 옵션 다이얼로그의 Linker 페이지에서 "Generate console application" 옵션을 선택했거나 -cc 커맨드라인 컴파일러 옵션으로 컴파일된 애플리케이션입니다. 콘솔 애플리케이션이 아닌 GUI 애플리케이션에서 Input이나 Output을 통해 읽거나 쓰려고 하면 I/O 에러가 발생합니다.

텍스트 파일에 사용할 수 있는 표준 I/O 루틴들 중 일부는 명시적으로 파라미터로 파일 변수를 지정하지 않아도 됩니다. 파일 파라미터가 생략되면 프로시저나 함수가 입력용인지 출력용인지에 따라서 Input 또는 Output이 기본값으로 설정됩니다. 예를 들면, Read(X)는 Read(Input, X)에 해당되며 Write(X)는 Write(Output, X)에 해당됩니다.

텍스트 파일에 동작하는 입력/출력 함수들을 호출하면서 파일을 지정하는 경우에는, 그 파일은 AssignFile로 외부 파일과 연결된 후 Reset, Rewrite 또는 Append로 열려야 합니다. Reset으로 연 파일을 출력용 프로시저/함수에 전달하면 에러가 발생합니다. 마찬가지로, Rewrite나 Append로 연 파일을 또한 입력용 프로시저/함수에 전달하면 에러가 발생합니다.

타입 미지정 파일

타입 미지정 파일(untyped file)은 타입이나 구조에 상관 없이 디스크 파일에 직접 액세스하기 위해 주로 사용되는 저수준 I/O 방식입니다. 타입 미지정 파일은 다른 지정 없이 file로만 선언됩니다. 타입 미지정 파일의 예는 다음과 같습니다.

```
var DataFile: file;
```

타입 미지정 파일의 경우, Reset 및 Rewrite 프로시저에서 데이터 전송에 사용될 레코드 크기를 지정할 수 있는 추가 파라미터를 지정할 수 있습니다. 전통적으로 레코드 크기는 기본적으로 128바이트입니다. 레코드 크기 1은 모든 파일의 정확한 크기를 반영하는 유일한 값입니다. (레코드 크기가 1이면 부분 레코드가 발생하지 않습니다.)
Read 및 Write를 제외하고 타입 지정 파일 표준 프로시저 및 함수 모두는 타입 미지정 파일에서도 사용할 수 있습니다. Read 및 Write 대신 고속 데이터 전송을 위한 BlockRead 및 BlockWrite라는 프로시저들이 사용됩니다.

텍스트 파일 장치 드라이버

사용자의 프로그램에서 고유한 텍스트 파일 장치 드라이버를 정의할 수 있습니다. 텍스트 파일 장치 드라이버는 오브젝트 파스칼의 파일 시스템과 일부 장치 사이의 인터페이스를 완벽하게 구현하는 네 가지 함수들로 구성됩니다.
각 장치 드라이버를 정의하는 네 개의 함수는 Open, InOut, Flush, Close입니다. 각 함수의 함수 헤더는 다음과 같습니다.

```
function DeviceFunc(var F: TtextRec): Integer;
```

여기서 DeviceFunc는 함수의 이름, 즉 Open, InOut, Flush 또는 Close입니다. TTextRec 타입에 대한 자세한 내용은 온라인 헬프를 참조하십시오. 장치 인터페이스 함수의 리턴값은 IOResult에 의해 리턴된 값이 됩니다. 리턴값이 0이면 동작이 성공한 것입니다.
장치 인터페이스 함수를 특정 파일과 연결하려면 사용자 지정 Assign 프로시저를 작성해야 합니다. Assign 프로시저는 반드시 텍스트 파일 변수의 멤버들인 네 개의 함수 포인터에 네 개의 장치 인터페이스 함수의 주소를 지정해야 합니다. 또한 이 프로시저는 Mode 필드에 fmClosed "매직" 상수를, BufSize에 텍스트 파일 버퍼 크기를, BufPtr에 텍스트 파일 버퍼에 대한 포인터를 저장해야 하고 Name 문자열의 내용을 지워야 합니다.
예를 들어, DevOpen, DevInOut, DevFlush, DevClose라는 장치 인터페이스 함수가 있다면 Assign 프로시저는 다음과 같을 것입니다.

```
procedure AssignDev(var F: Text);
begin
```

```
  with TTextRec(F) do
  begin
    Mode := fmClosed;
    BufSize := SizeOf(Buffer);
    BufPtr := @Buffer;
    OpenFunc := @DevOpen;
    InOutFunc := @DevInOut;
    FlushFunc := @DevFlush;
    CloseFunc := @DevClose;
    Name[0] := #0;
  end;
end;
```

장치 인터페이스 함수들은 파일 레코드의 UserData 필드에 별도의 정보를 저장할 수 있습니다. 이 필드는 제품 파일 시스템에 의해 수정될 수 없습니다.

장치 함수

텍스트 파일 장치 드라이버를 구성하는 함수는 다음과 같습니다.

■ Open 함수

Open 함수는 Reset, Rewrite, Append 표준 프로시저에 의해 호출되어 장치와 연결된 텍스트 파일을 엽니다. 입력 시 Open 함수가 Reset, Rewrite, Append 중 어느 함수에서 호출되었는지에 따라 Mode 필드에 각각 fmInput, fmOutput, fmInOut 값이 들어갑니다.

Open 함수는 Mode 값에 따라 입력 또는 출력을 할 수 있도록 파일을 준비합니다. Mode에 fmInOut이 지정되었으면(Open이 Append에 의해 호출된 경우) Open이 리턴되기 전에 fmOutput으로 변경해야 합니다.

Open은 항상 다른 장치 인터페이스 함수보다 먼저 호출됩니다. 따라서 AssignDev는 OpenFunc 필드만 초기화하고 나머지 Open을 구성하는 벡터는 초기화하지 않습니다. Mode에 따라 Open은 입력용 함수 또는 출력용 함수에 대한 포인터를 설정할 수 있습니다. 덕분에 InOut, Flush 함수와 CloseFile 프로시저는 현재 모드를 판단할 필요가 없습니다.

■ InOut 함수

InOut 함수는 장치에 입력 또는 출력이 필요할 때마다 Read, Readln, Write, Writeln, Eof, Eoln, SeekEof, SeekEoln, CloseFile 표준 루틴에 의해 호출됩니다.

Mode가 fmInput이면 InOut 함수는 BufSize 문자를 BufPtr^에 읽어 오고 BufEnd에서 읽

은 문자 수를 리턴합니다. 또한 BufPos에 0을 저장합니다. InOut 함수가 입력 요청의 결과로 BufEnd에서 0을 리턴하면 그 파일에 대해 Eof가 True가 됩니다.
Mode가 fmOutput이면 InOut 함수는 BufPtr^로부터 BufPos 문자를 쓰고 BufPos에 0을 리턴합니다.

■ Flush 함수

Flush 함수는 각각의 Read, Readln, Write, Writeln 끝에서 호출됩니다. 이 함수는 텍스트 파일 버퍼를 플러시할 수 있습니다.
Mode가 fmInput이면 Flush 함수는 BufPos 및 BufEnd에 0을 저장하여 버퍼의 나머지 읽지 않은 문자를 플러시할 수 있습니다. 이 기능은 거의 사용되지 않습니다.
Mode가 fmOutput이면 Flush 함수는 장치에 쓰여진 텍스트가 장치에 바로 나타나도록 하는 InOut 함수와 똑같이 버퍼 내용을 쓸 수 있습니다. Flush가 아무 일도 하지 않으면 버퍼가 꽉 차거나 파일이 닫힐 때까지 텍스트는 장치에 나타나지 않습니다.

■ Close 함수

Close 함수는 CloseFile 표준 프로시저에 의해 호출되어 장치와 연결된 텍스트 파일을 닫습니다. 해당 프로시저가 열려는 파일이 이미 열려 있으면 Reset, Rewrite 및 Append 프로시저는 Close를 호출합니다. Mode가 fmOutput이면 Close를 호출하기 전에 파일 시스템은 InOut 함수를 호출하여 모든 문자가 장치에 기록되었는지 확인합니다.

Null 종료 문자열의 처리

델파이 언어의 확장 문법을 사용하면, Read, Readln, Str 및 Val 표준 프로시저를 인덱스가 0부터 시작하는 문자 배열에 적용되도록 할 수 있으며, Write, Writeln, Val, AssignFile 및 Rename 표준 프로시저를 인덱스가 0부터 시작하는 문자 배열 및 문자 포인터에 적용할 수 있습니다.

Null 종료 문자열 함수들

또한 Null 종료 문자열의 처리를 위해 다음과 같은 함수들이 제공됩니다. Null 종료 문자열에 대한 자세한 내용은 4장의 "Null 종료 문자열 사용"을 참조하십시오.

표 9.2 Null 종료 문자열 함수

함수	설명
StrAlloc	특정 크기의 문자 버퍼를 힙(heap)에 할당합니다.
StrBufSize	StrAlloc 또는 StrNew를 사용하여 할당된 문자 버퍼의 크기를 리턴합니다.
StrCat	두 문자열을 연결합니다.
StrComp	두 문자열을 비교합니다.
StrCopy	문자열을 복사합니다.
StrDispose	StrAlloc 또는 StrNew를 사용하여 할당된 문자 버퍼를 해제합니다.
StrECopy	문자열을 복사하고 문자열 끝에 대한 포인터를 리턴합니다.
StrEnd	문자열 끝에 대한 포인터를 리턴합니다.
StrFmt	하나 이상의 값을 문자열 내로 서식화(format)합니다.
StrIComp	대소문자를 구별하지 않고 두 문자열을 비교합니다.
StrLCat	결과 문자열의 지정한 최대 길이로 두 문자열을 연결합니다.
StrLComp	지정한 최대 길이로 두 문자열을 비교합니다.
StrLCopy	지정한 최대 길이까지 문자열을 복사합니다.
StrLen	문자열의 길이를 리턴합니다.
StrLFmt	지정한 최대 길이로 하나 이상의 값을 한 문자열로 서식화(format)합니다.
StrLIComp	대소문자를 구별하지 않고 지정한 최대 길이만큼 두 문자열을 비교합니다.
StrLower	문자열을 소문자로 변환합니다.
StrMove	한 문자열의 문자 블럭을 다른 문자열로 이동합니다.
StrNew	문자열을 힙에 할당합니다.
StrPCopy	델파이 문자열을 Null 종료 문자열에 복사합니다.
StrPLCopy	지정한 최대 길이만큼 델파이 문자열을 Null 종료 문자열에 복사합니다.
StrPos	문자열 내에서 특정 부분 문자열이 처음 나타나는 위치의 포인터를 리턴합니다.
StrRScan	문자열 내에서 특정 문자가 마지막으로 나타나는 위치의 포인터를 리턴합니다.
StrScan	문자열 내에서 특정 문자가 처음 나타나는 위치의 포인터를 리턴합니다.
StrUpper	문자열을 대문자로 변환합니다.

표준 문자열 처리 함수에는 위 함수들의 멀티바이트 버전에 해당하는 함수들이 있으며, 이 함수들은 로케일 특정 문자 정렬도 지원합니다. 멀티바이트 함수의 이름은 Ansi로 시작합니다. 예를 들면, StrPos의 멀티바이트 버전은 AnsiStrPos입니다. 멀티바이트 문자 지원은 운영 체제에 의존하며 현재 로케일을 기반으로 합니다.

와이드 문자 문자열

System 유닛은 Null 종료 와이드 문자 문자열을 1바이트 혹은 2바이트의 긴 문자열로 변환

할 수 있는 WideCharToString, WideCharLenToString 및 StringToWideChar 함수를 제공합니다.
대입으로도 문자열들 사이에 변환을 할 수 있습니다. 다음의 코드는 둘 다 유효합니다.

```
MyAnsiString := MyWideString;
MyWideString := MyAnsiString;
```

와이드 문자 문자열에 대한 자세한 내용은 4장의 "확장 문자셋 정보"를 참조하십시오.

기타 표준 루틴

아래의 표는 코드기어 개발툴 라이브러리의 루틴들 중 자주 사용되는 프로시저 및 함수를 보여줍니다. 이 표는 표준 루틴의 전체 목록은 아닙니다. 이 루틴들과 기타 루틴들에 대한 자세한 내용은 온라인 헬프를 참조하십시오.

표 9.3 기타 표준 루틴

프로시저/함수	설명
Addr	지정된 객체에 대한 포인터를 리턴합니다.
AllocMem	메모리 블럭을 할당하고 각 바이트를 0으로 초기화합니다.
ArcTan	지정된 숫자의 아크탄젠트를 계산합니다.
Assert	전달된 표현식이 true가 아니면 예외를 발생시킵니다.
Assigned	포인터나 프로시저 변수가 nil(unassigned)인지 검사합니다.
Beep	컴퓨터 스피커를 사용하여 표준 경고음을 발생합니다.
Break	for, while, repeat 문을 빠져나갑니다.
ByteToCharIndex	문자열에서 지정된 바이트를 포함하는 문자의 위치를 리턴합니다.
Chr	지정된 정수 값 해당하는 문자를 리턴합니다.
Close	파일을 닫습니다.
CompareMem	두 메모리 이미지의 바이너리 비교를 수행합니다.
CompareStr	대소문자를 구별하여 문자열을 비교합니다.
CompareText	대소문자는 구별하지 않고 문자열을 비교합니다.
Continue	for, while, repeat 문의 다음 반복을 실행합니다.
Copy	문자열의 부분 문자열이나 동적 배열의 일부분을 리턴합니다.
Cos	각도의 코사인을 계산합니다.
CurrToStr	currency 변수를 문자열로 변환합니다.

프로시저/함수	설명
Date	현재 날짜를 리턴합니다.
DateTimeToStr	TDateTime 타입의 변수를 문자열로 변환합니다.
DateToStr	TDateTime 타입의 변수를 문자열로 변환합니다.
Dec	서수 변수 혹은 타입 지정 포인터 변수의 값을 감소시킵니다.
Dispose	변수에 동적 할당된 메모리를 해제합니다.
ExceptAddr	현재 예외가 발생한 주소를 리턴합니다.
Exit	현재 프로시저를 빠져나갑니다.
Exp	X의 지수 값을 계산합니다.
FillChar	연속된 바이트들을 지정된 값으로 채웁니다.
Finalize	동적으로 할당된 변수의 초기화를 해제합니다.
FloatToStr	부동 소수점 값을 문자열로 변환합니다.
FloatToStrF	지정된 서식(format)을 사용하여 부동 소수점 값을 문자열로 변환합니다.
FmtLoadStr	리소스의 서식(format) 문자열을 사용하여 서식화된 출력을 리턴합니다.
FmtStr	일련의 배열로부터 서식화된 문자열을 만들어냅니다.
Format	서식(format) 문자열과 일련의 배열로부터 서식화된 문자열을 만들어냅니다.
FormatDateTime	날짜 및 시간 값을 서식화합니다.
FormatFloat	부동 소수점 값을 서식화합니다.
FreeMem	할당된 메모리를 해제합니다.
GetMem	동적으로 메모리를 할당하고 해당 메모리 블록의 주소를 포인터 파라미터로 돌려줍니다.
Halt	프로그램을 비정상적으로 종료시킵니다.
Hi	표현식 결과의 상위 바이트를 부호없는 값으로 리턴합니다.
High	타입, 배열 또는 문자열의 범위에서 최고 값을 리턴합니다.
Inc	서수 변수의 값을 증가시킵니다.
Initialize	동적으로 할당된 변수를 초기화합니다.
Insert	문자열의 지정된 위치에 부분 문자열을 삽입합니다.
Int ?	실수의 정수 부분을 리턴합니다.
IntToStr	정수를 문자열로 변환합니다.
Length	문자열 또는 배열의 길이를 리턴합니다.
Lo	표현식 결과의 하위 바이트를 부호없는 값으로 리턴합니다.
Low	타입, 배열 또는 문자열의 범위에서 최저 값을 리턴합니다.
LowerCase	아스키 문자열을 소문자로 변환합니다.
MaxIntValue	정수 배열에서 가장 큰 부호있는 값을 리턴합니다.
MaxValue	배열에서 가장 큰 부호있는 값을 리턴합니다.
MinIntValue	정수 배열에서 가장 작은 부호있는 값을 리턴합니다.

프로시저/함수	설명
MinValue	배열에서 가장 작은 부호있는 값을 리턴합니다.
New	동적 변수를 새로 만들어 지정된 포인터로 이를 참조합니다.
Now	현재 날짜와 시간을 리턴합니다.
Ord	서수(ordinal) 타입 표현식의 순서값을 리턴합니다.
Pos	문자열 내에서 지정된 부분 문자열의 첫 번째 문자의 인덱스를 리턴합니다.
Pred	서수 값의 바로 앞 값을 리턴합니다.
Ptr	지정된 값을 포인터로 변환합니다.
Random	지정된 범위 내에서 임의의 수를 생성합니다.
ReallocMem	동적 할당 메모리를 다시 할당합니다.
Round	실수 값을 반올림합니다.
SetLength	문자열 변수 또는 배열의 동적 길이를 설정합니다.
SetString	지정된 문자열의 내용 및 길이를 설정합니다.
ShowException	예외 메시지를 주소와 함께 표시합니다.
ShowMessage	서식화되지 않은 문자열과 OK 버튼이 있는 메시지 상자를 표시합니다.
ShowMessageFmt	서식화된 문자열과 OK 버튼이 있는 메시지 상자를 표시합니다.
Sin	라디안 각도의 사인 값을 리턴합니다.
SizeOf	변수 또는 타입이 차지하는 바이트 수를 리턴합니다.
Sqr	숫자의 제곱을 리턴합니다.
Sqrt	숫자의 제곱근을 리턴합니다.
Str	정수나 실수를 문자열로 변환합니다.
StrToCurr	문자열을 currency 값으로 변환합니다.
StrToDate	문자열을 날짜 서식(TDateTime)으로 변환합니다.
StrToDateTime	문자열을 TDateTime으로 변환합니다.
StrToFloat	문자열을 부동 소수점 값으로 변환합니다.
StrToInt	문자열을 정수로 변환합니다.
StrToTime	문자열을 시간 서식(TDateTime)으로 변환합니다.
StrUpper	ASCII 문자열을 대문자로 리턴합니다.
Succ	서수 값의 다음 값을 리턴합니다.
Sum	배열의 요소 합을 리턴합니다.
Time	현재 시간을 리턴합니다.
TimeToStr	TDateTime 타입의 변수를 문자열로 변환합니다.
Trunc	실수를 잘라내어 정수로 만듭니다.
UniqueString	지정한 문자열의 참조 카운트가 1이 되도록 합니다. (단일 참조를 위해 문자열 복사가 일어날 수도 있습니다.)

프로시저/함수	설명
UpCase	문자를 대문자로 변환합니다.
UpperCase	문자열을 대문자로 리턴합니다.
VarArrayCreate	variant 타입 배열을 만듭니다.
VarArrayDimCount	variant 타입 배열의 차원 수를 리턴합니다.
VarArrayHighBound	variant 타입 배열에서 지정한 차원의 상위 경계를 리턴합니다.
VarArrayLock	variant 타입 배열을 잠그고 데이터에 대한 포인터를 리턴합니다.
VarArrayLowBound	variant 타입 배열에서 지정한 차원의 하위 경계를 리턴합니다.
VarArrayOf	1차원 variant 타입 배열을 만들고 채웁니다.
VarArrayRedim	variant 타입 배열의 크기를 조정합니다.
VarArrayRef	전달된 variant 타입 배열에 대한 참조를 리턴합니다.
VarArrayUnlock	variant 타입 배열의 잠금을 해제합니다.
VarAsType	variant을 지정된 타입으로 변환합니다.
VarCast	variant 타입을 지정된 타입으로 변환하고 결과를 변수에 저장합니다.
VarClear	variant 타입을 지웁니다.
VarCopy	variant 타입을 복사합니다.
VarToStr	variant 타입을 문자열로 변환합니다.
VarType	지정된 variant의 타입 코드를 리턴합니다.

서식(format) 문자열에 대한 자세한 내용은 온라인 헬프의 "Format strings"를 참조하십시오.