목 차

Crownix Report 스크립트란?

Crownix Report 에서 제공하는 **스크립트**는 C 언어와 유사한 자체 문법을 가지고 있으며 Crownix Report 로 작성되는 **스크립트 문서**에 프로그래밍 기능을 제공하는 **프로그램 언어**입니다.

이러한 기능은 지금까지의 문서에 대한 개념을 바꾸는 것은 물론, 한국형 문서 작성에 대한 높은 이식률을 제공할 수 있는 것입니다. 물론 이와 유사한 시도는 다른 개발 도구들에서도 그 내용이 부분적으로 도입된 적은 있으나 그 적용 분야가 상당히 제한되거나 사용이 불편했었습니다.

스크립트가 지원하는 프로그램의 특징

첫째, 프로그램의 입출력에 대한 기술은 Crownix Report Designer 를 이용한 문서 편집 작업과 동일합니다. 이는 3 세대 언어로 일반 응용 프로그램을 작성할 경우, 60~70%의 코드가 자료의 입출력 형태에 대한 처리에 소요되는 것으로 볼 때, 프로그램의 생산성을 높일 수 있는 중요한 기능입니다.

둘째, 자료 흐름의 표현이 매우 자연스럽습니다. 한 응용 프로그램에서 자료의 변동 사항을 관련된 여러 응용 프로그램에 간단하게 적용시켜 표현할 수 있습니다.

셋째, 클라이언트/서버 환경에서의 응용 프로그램 개발을 쉽게 지원합니다. 이는 프로그램 작성자에게 클라이언트 또는 서버 어느 곳에 있는 프로그램과 자료의 이용을 동일하게 바라볼 수 있는 환경을 제공함으로써 가능합니다. ODBC 를 이용한 데이터베이스 접속은 물론이고 특화된 DLL 을 통한 효율적인 클라이언트/서버 환경을 제공합니다.

넷째, 그래픽 사용자 접속 환경 하에서 문서와 프로그램을 동일시합니다. 즉, 문서가 곧 프로그램이며 프로그램이 곧 문서의 기능을 가지고 있습니다. 문서를 작성하면서 동시에 프로그램을 작성할 수 있으며 이미 작성된 문서에 프로그램 기능을 추가할 수 있고 이미 작성된 프로그램을 문서와 같은 형태로 쉽게 변경할 수도 있습니다.

다섯째, 질적으로 우수한 데이터베이스 응용 프로그램을 가능하게 함은 물론 짧은 시간 내에 많은 응용 프로그램을 생성할 수 있는 기능을 제공합니다. 간단한 4GL 및 SQL 질의어 작성을 가능하게 합니다.

이와 같이 문서와 프로그램의 기능이 결합되어 나타나는 효과는 그래픽 화면에 보이는 대로 고품위 출력이 가능합니다. 사용자 기능이 프로그램되는 지능형 문서로부터 편리한 인터페이스를 갖춘 일반 자료 처리 프로그램까지 광범위하게 적용되어 업무 및 개발 생산성을 크게 증가시킵니다.

1. 스크립트 작성

스크립트 작성은 Crownix Report 의 4 가지 **보고서 형태 (일반, 표, 라벨, 스크립트 문서)** 중에서 **스크립트 문서**에서 주로 사용하며 일반문서, 표, 라벨 문서에서는 부분적으로만 사용 가능합니다.

1.1. 스크립트 기본 설명

1.1.1. Crownix Report 스크립트 문서의 특징

Crownix Report 에서 작성된 문서의 내용과 프로그램은 하나의 파일에 존재합니다. 문서는 Crownix Report 의 편집 기능을 이용하여 작성되며 프로그래밍은 작성된 문서에 필요한 스크립트를 기술함으로써 이루어집니다.

스크립트는 **Query 정의**에서 작성한 SQL 문을 가지고 데이터베이스로부터 **하나의 결과 셋 (레코드)** 을 가져올 때마다 **기술한 스크립트도 한 번 실행**되게 됩니다. 스크립트 내에서도 다른 Query 를 작성하여 데이터베이스에 데이터를 요구할 수 있고 그렇게 읽어들인 자료를 편집하여 보여줄 수도 있습니다. Query 정의에서 **주쿼리를 작성하지 않는 경우라면 스크립트가 한 번만 실행**됩니다.

앞에서 설명한 사항처럼 일반적으로 스크립트를 사용하는 문서는 데이터베이스를 연결하는 경우에 사용하지만 파일접속 등의 문서에서도 사용할 수 있습니다. 단, **파일접속인 경우에는 스크립트 보고서에서만 사용 가능**합니다.

1.1.2. 객체와 객체 변수

객체

객체란 기능과 관련 자료를 갖는 대상체를 뜻하는 것으로서 Crownix Report 에서 표현하는 모든 것은 객체이거나 객체의 일부입니다.

Crownix Report 의 기능은 크게 편집 기능과 프로그래밍 기능으로 분류되는데 이들 기능은 또한 객체와 밀접한 관련이 있습니다.

Crownix Report 에서의 편집 기능이란 객체를 만들거나 변경하는 일이라할 수 있고 프로그래밍 기능이란 만들어진 객체에 스크립트를 기술하는 일이라할 수 있습니다.

객체 변수

Crownix Report 에서 사용하는 변수는 크게 **객체 변수, 지역 변수, 임시 변수**로 분류됩니다. 여기에서는 객체와 관련이 있는 객체 변수에 대해 간략히 살펴보겠습니다.

객체 변수란 Crownix Report 에서 작성한 객체에 사용자가 부여한 이름으로 Crownix Report 문서 내의 모든 객체에 변수를 부여할 수 있습니다.

객체 변수명은 대소문자 구분이 없고 변수명은 스크립트 명령문이나 내장 함수에도 적용됩니다. 그리고 객체 변수명은 전역 변수이므로 한 문서내에서 유일해야 합니다.

객체 변수를 선언하려면 스크립트 대화상자에서 객체를 선택한 다음 변수명을 입력하고 변수형을 선택하면 됩니다.

객체 변수의 변수형은 변수에 입력되는 값의 데이터 형(Data Type)을 말합니다. 변수형에는 숫자(NUM), 문자(CHAR), 날짜(DATE), 시간(TIME)이 있습니다. 그러나 모든 변수가 변수형을 가질 수 있는 것은 아닙니다. 예를 들면 텍스트 상자, 표의 셀은 변수형을 가질 수 있으며 도형객체, 비트맵객체 등은 변수형을 가질 수 없습니다.

객체 변수명 짓기의 규칙

객체 변수명을 지정할 때 주의할 것은 **한 파일 내에서 중복되는 객체 변수명은 허용되지 않는다**는 점과 **숫자를 변수명 맨 처음에서 쓸 수 없다**는 점입니다.

또한 **스크립트에서 지정해 놓고 사용하는 단어(예약어)**도 사용할 수 없습니다.

객체 변수명 짓기의 규칙은 다음 표와 같습니다.

	영문/숫자/_(under bar)	한글,한자,심볼 등 (제어 문자 제외)
변수의 길이	1 ~ 20 개 문자를 조합하여 표현	1~ 10개의 문자를 조합하여 표현
변수의 시작문자	숫자를 제외한 모든 문자를 사용	숫자를 제외한 모든 문자를 사용
보기	a, xy, abcdefghijk, _title_, search, A1 ,a_b	가, 가나다라자차, _제목_, 아홉_TO_다섯, 檢索, ☎, ㄱ , ゾ

객체 이름 짓기의 예외는 다음 표와 같습니다.

내용	자판에서 사용하는 심볼, 제어문자, 기능키 등은 객체의 이름을 정의 하는 데 사용할 수 없습니다.	스크립트에서 사용하는 단어는 객체의 이름으로 사용할 수 없습니다.
보기	#hour, <Small, >Large, //term, -mid-	Endif, else, sum, find, scrolldn 등

표의 셀 객체에 변수 부여

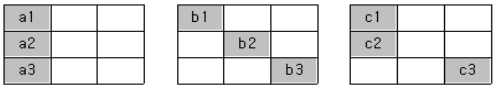
표에 대해서는 한 번에 다수의 셀에 **일련 번호**를 갖는 객체 변수명을 일괄적으로 줄 수 있습니다. 이는 반복되는 형태의 자료를 표현하는데 표를 많이 사용하며 표 객체를 구성하는 셀들이 늘어나는 표의 특성과 밀접하게 관련되어 있습니다. **(이 일련번호를 갖는 객체변수명은 스크립트 내에서 동일한 이름의 배열로 사용될 수 있습니다.)**

표의 셀에 변수를 지정한 형태는 다음 표의 모양과 같습니다.

제 품 명	단 가	수 량	금 액
제품명1	단가1	수량1	금액1
제품명2	단가2	수량2	금액2
제품명3	단가3	수량3	금액3
제품명4	단가4	수량4	금액4
제품명5	단가5	수량5	금액5

제품명 열에는 순서대로 제품명 1, 제품명 2, 제품명 3, 제품명 4, 제품명 5 라는 객체 변수명이 주어져 있습니다. 이것은 같은 특성을 갖는 자료로서 스크립트 대화상자를 통하여 변수를 부여하는 방법은 **변수가 부여될 열을 블록지정**한 다음 문자열 **‘ 제품명’** 을 **변수 입력란**에 입력하면 블록지정한 첫 셀에서 부터 다음 예제와 같은 순서로 셀에 변수명이 부여됩니다.

다음 그림과 같이 세로, 대각선, 임의로 선택해서 객체변수를 지정하는 것이 가능합니다.



(1)세로 선택 (2)대각선 선택 (3)임의의 선택

위 예제에서 보이는 것처럼 한 번에 다수의 셀에 객체 변수명을 부여하는 경우에는 블록 지정된 셀 중에서 가장 위쪽 행의 맨 왼쪽셀을 시작으로, 먼저 왼쪽에서 오른쪽으로 그리고 위쪽에서 아래쪽으로 블록 지정된 셀을 찾아 일련번호를 갖는 객체 변수를 부여합니다.

만약, (1)번 예제에서 스크립트 대화상자의 변수 입력란에 ‘ a10’ 이라고 변수를 부여했다면 맨 위의 셀에서부터 순서대로 일련번호 10, 11, 12 가 붙어 첫 번째 셀은 a10, 두 번째 셀은 a11 , 세 번째 셀은 a12 로 변수명이 주어지게 됩니다.

1.1.3. 객체와 스크립트

사용자는 Crownix Report 문서 내의 모든 객체에 변수명을 부여할 수 있고 한 문서당 하나의 스크립트를 기술할 수 있습니다. 또한 변수형이 있는 객체는 값을 가질 수 있으나 변수형이 없는 객체는 값을 가질 수 없습니다.

변수형이 있는 객체 (값을 가질 수 있는 객체)

텍스트 상자나 표의 셀 객체은 스크립트로 프로그램을 작성할 때 매우 유용하게 사용할 수 있는 객체입니다.

구분	객체 종류	가질 수 있는 변수형	스크립트가 기술되었을 때의 주요기능
글틀	글틀	숫자 문자 날짜 시간	객체 변수명이 스크립트 내에서 변수로 사용될 수 있으며, 객체가 가질 수 있는 변수형은 왼쪽 네 가지 중에 하나입니다. 객체 변수명을 통하여 자료를 입력하거나 출력할 수 있습니다.글틀 객체는 다른 개발 도구들의 필드와 비교될 수 있습니다.
표	칸	숫자 문자 날짜 시간	칸 객체 변수명은 스크립트 내에서 변수로 사용될 수 있으며, 객체가 가질 수 있는 변수형은 왼쪽 네 가지 중에 하나입니다. 객체 변수명을 통하여 자료를 입력하거나 출력할 수 있습니다. 칸 객체는 표의 내부에 묶여 있으며, 반복되는 자료의 표현에 유용합니다.

[스크립트와 밀접한 관계의 객체]

변수형이 없는 객체 (값을 가질 수 없 는 객체)

변수형을 가질 수 없는 객체는 스크립트를 이용하여 자료의 입력이나 출력을 할 수 없습니다.
변수형을 가질 수 없는 객체는 다음과 같습니다.

구분	객체 종류	표현 가능한 자료 형태	스크립트가 기술되었을 때의 주요기능
그룹	그룹	없음	객체 변수는 가질 수 있지만 변수형을 정의 할 수 없으며 변수로 사용할 수도 없습니다.
	곡선	없음	"
	다각선	없음	"
	다각형	없음	"
	둥근 사각형	없음	"
	마름모	없음	"
	사각기둥	없음	"
	사각형	없음	"
	원기둥	없음	"
	직성	없음	"
	평행 사변형	없음	"
비트맵	비트맵	없음	"

1.1.4. Crownix Report 스크립트의 실행

스크립트 문서인 경우, 스크립트가 실행되는 시점
데이터베이스 접속에서 주쿼리의 결과 레코드 또는 파일접속 문서에서 기본필드정의의 결과 레코드에서 하나의 레코드를 가져와 처리할 때마다 실행됩니다.

일반, 표, 라벨 형태의 보고서가 스크립트를 포함하는 경우, 스크립드가 실행되는 시점
파일을 처음 open 했을 때에만 스크립트를 실행하게 됩니다.

1.2. 스크립트 기술 절차

스크립트 기술은 다음과 같은 순서로 합니다.

1. **데이터베이스-스크립트 입력**을 선택하거나 도구상자의 해당버튼을 눌러서 **스크립트 입력 대화상자**를 엽니다.

2. **스크립트에서 기술할 객체**를 선택한 후, 변수명 입력란에 **변수명**을 입력하고 **변수형**을 선택합니다

객체에 변수명을 입력해야만 스크립트에서 사용할 수 있습니다.

3. 스크립트 편집창에서 스크립트를 기술한 후 **검증**을 하거나 **닫기** 버튼을 누릅니다.

1.2.1. 변수

Crownix Report 에서 **변수를 정의하여 사용할 수 있는 방법**은 크게 세 가지로 나누어집니다.

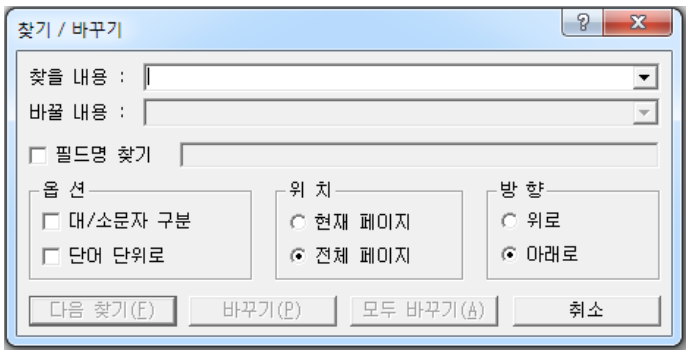
첫째는 객체에 변수명을 부여한 **객체 변수**이고
둘째는 객체의 변수명은 아니지만 스크립트 전체에서 사용할 수 있는 **지역 변수**이고
셋째로는 스크립트 내부에서 임시로 사용되는 **임시 변수**가 있습니다.

1.2.2. 스크립트 편집

스크립트를 편집하는데 편리함을 제공하기 위한 기능으로는 **블록 단위의 복사 (<Ctrl+Ins>, <Ctrl+C>), 오려두기 (<Ctrl+Del>, <Ctrl+X>), 붙이기 (<Shift+Ins>,< Ctrl+V>)** 기능과 **문자열 찾기/바꾸기(<Ctrl+F>)** 등을 제공합니다.

문자열 찾기/바꾸기는 어떠한 문자열의 위치를 찾거나 다른 문자열로 바꾸고자 할 때 유용하게 사용될 수 있습니다.

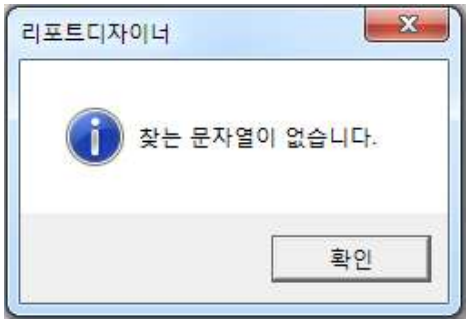
스크립트를 편집하는 중에 **<Ctrl+F>**을 누르면 **문자열 찾기/바꾸기** 대화상자가 나타납니다.



단어만 체크박스를 선택하면 입력한 문자열로만 구성된 단어를 찾고 **대소문자구별** 체크박스를 선택하면 대소문자가 모두 일치하는 문자열을 찾습니다

문자열을 찾는 위치는 **현재 편집 창에서 깜빡 거리고 있는 입력 커서 위치에서부터** 찾게 되며 맨 끝까지 커서가 다 이동했을 경우, 처음부터 다시 찾습니다.

문자열 찾기/바꾸기 기능에서 더 이상 찾을 문자열이 없을 경우에는 다음과 같은 대화상자가 나타납니다.



1.2.3. 스크립트 문법검사

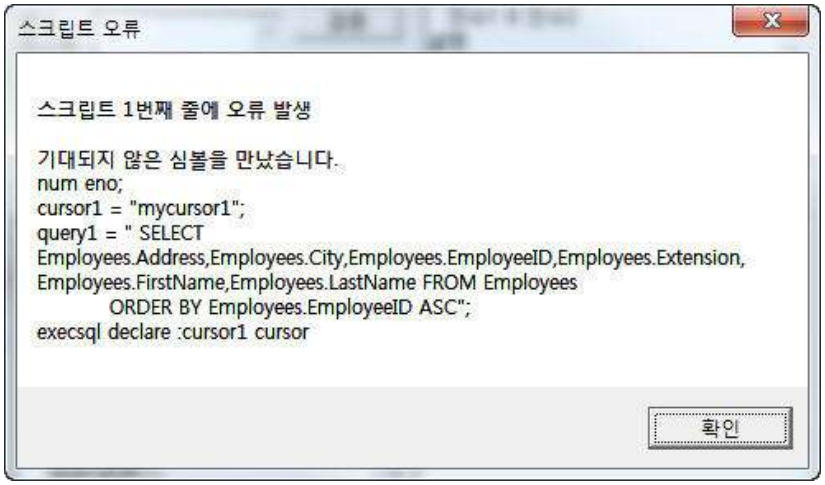
스크립트 입력 대화상자에 기술된 스크립트는 Crownix Report 에 의해서 해석됩니다.

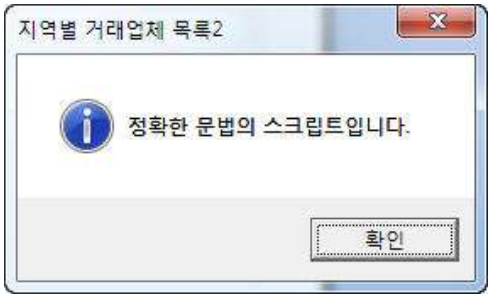
Crownix Report Designer 에서는 스크립트 작성 및 문법검사를 하고 Crownix Report Viewer 에서는 기술된 스크립트를 해석을 하고 이 해석된 결과를 가지고 스크립트를 실행합니다.

스크립트 입력 대화상자가 열려 있고 스크립트가 기술되어 있는 한 개의 객체가 선택되어 있는 상태라고 가정할 때, 이 객체에 기술된 스크립트를 Crownix Report 에서 해석하는 시점은 다음과 같습니다.

- 스크립트 대화상자의 **확인**이나 **검증** 버튼을 눌렀을 경우
- 스크립트 대화상자를 **닫는** 경우
- Crownix Report Viewer 에서 스크립트를 포함하는 **문서를 열었을** 경우

스크립트가 해석되어질 때, 스크립트의 문법이 검사되고 이 때, 문법에 어긋나는 사항에 대해서는 오류 메시지를 오류가 생기지 않았다면 확인 메시지를 보여줍니다.





오류 메시지는 오류가 발생한 **행의 번호, 오류의 내용**을 알려주고 오류를 유발한 문장과 스크립트 내의 문자열을 "☞ ☜" 표시 사이에 보여줍니다.

2. 스크립트 문법

2.1. 상수 (constant)

스크립트에서 사용 가능한 상수에는 문자, 숫자, 날짜, 시간 상수가 있습니다.

2.1.1. 문자 상수 (string constant)

문자 상수는 Null 문자 (Null Character : "") 이상의 문자를 표현하며 **제어 문자를 제외한 모든 문자**를 사용할 수 있습니다.

- 반드시 **인용 부호(")**로 묶어서 표기해야 합니다.
- 최대 길이가 **영문, 숫자, 기호인 경우는 256 자, 한글인 경우에는 128 자**입니다.
- 문자 상수 내부에 인용 부호를 포함시킬 때는 **물결무늬 문자 (~ : Escape Character)**를 사용합니다.

  예) "ABC", "1234", "소프트", "&$%@#", "상품 0+상품 1", "~"나라~"", "~~abc"

  여기에서 **" ~"나라~""** 는 **" 나라"** 라는 문자열을, **" ~~abc"**는 **~abc** 라는 문자열을 나타냅니다.

물결무늬(~) 부호는 스크립트 내에서 Query 를 작성하는 경우에 SQL 문 내부에 쓰이는 특수문자 앞에도 사용됩니다.

2.1.2. 숫자 상수 (number constant)

- 숫자 상수는 **정수**나 **실수**로 나타냅니다.
- **음수**나 **소수점 이하의 수**도 가능합니다.
- 숫자 상수의 최대 유효 자리 수는 **15 자리**입니다.

  예) 123, 123.0, -123, 16.98

2.1.3. 날짜 상수 (date constant)

날짜 상수는 **인용 부호(“)**로 묶어서 “ 년년년년/월월/일일” , "년년/월월/일일" 또는 "월월/일일" 과 같이 **' / '를 구별자**로 사용하여 표기합니다. "월월/일일"의 형식에서는 년도를 현재 시스템에 설정되어 있는 년도로 간주합니다

예) "1994/01/20", "94/2/5". "9/1"

 Report Designer **스크립트에서의 날짜 상수는** 1900 **년 이후부터 표기가 가능합니다.**

2.1.4. 시간 상수 (time constant)

시간 상수는 **인용 부호(“)**로 묶어서 "시시:분분:초초" 또는 "시시:분분" 과 같이 **' : ' 를 구별자**로 사용하고 **24 시간** 표기법을 사용합니다.

예) "12:00", "3:53:29", "17:23"

2.2. 변수 (variable)

2.2.1. 변수의 종류

변수는 영문의 경우, 최대 20 자 한글의 경우, 최대 10 자까지 사용할 수 있으며 **문자** 또는 '_'로 시작해야 합니다. 변수명은 대/소문자 구별을 하지 않고 종류에 따라 3 가지로 구분됩니다.

객체 변수 (object variable)

객체 변수는 Crownix Report 에서 작성한 **객체**에 **사용자가 부여한 이름**으로 한 개의 파일 내에서 같은 변수명을 둘 이상의 객체에 부여할 수 없습니다.

객체 변수를 선언하려면 스크립트 대화상자에서 객체를 선택한 다음 **변수명**과 **변수형**을 주면 됩니다.

 Crownix Report **에서는 표의 셀과 텍스트 상자객체를 제외한 그 밖의 객체들(예: 원, 선, 원기둥..)은 변수형을 가질 수 없습니다.**

변수값 변경은 **스크립트 입력** 대화상자에서 바꿀 수 있습니다.

지역 변수 (local variable)

지역 변수는 **스크립트 내부에서만 의미를 가지는** 변수입니다.

객체에 영향을 주지 않으며 수행 시 문자 변수는 ""(Null Character)로 숫자 변수는 0 으로 항상 **초기화**되어서 사용됩니다. 즉, 지역 변수는 선언되어 있는 **스크립트가 실행되는 동안에만 유효**하며 실행이 끝난 후에는 값을 보관하지 않습니다.

지역 변수의 변수형으로 **숫자, 문자, 날짜, 시간** 형태를 사용할 수 있으며 다음과 같이 사용합니다.

```
NUM i, j;

CHAR s;

DATE d;

TIME t;
```

임시 변수 (temporary variable)

임시 변수는 스크립트 수행 결과, 객체 변수 또는 지역 변수가 아닌 변수에 값이 할당되었을 때, **자동으로 생성되는 변수**입니다. 그렇기 때문에 임시 변수는 **파일이 열려 있는 동안 유효**하고 파일에 값이 저장되지는 않습니다.

예를 들어, B 가 객체 변수로 선언되어 있지 않은 상태(A 는 객체 변수)에서 스크립트에 B = A; 와 같은 스크립트를 기술하면 B 라는 임시 변수가 생성됩니다. 물론 이후 다른 객체에 B 라는 변수명을 주면 임시 변수는 객체 변수로 변합니다.

임시 변수는 특정 변수형이 선언되지 않고 **문자열 변수**로 인식되기 때문에 대소 비교에 임시 변수를 사용하는 경우는 **ASCII 순서**로 비교됩니다.

문자열 변수의 대소 비교는 ASCII 순서이므로 "10"<"2" 입니다. 임시 변수는 역시 문자형 변수이므로 대소 비교하는 문에 사용할 때 주의해야 합니다.

2.2.2. 변수형 (variable type)

변수형은 **변수에 보관하는 자료의 형태**로 변수는 변수형(variable type : **문자열, 숫자, 날짜, 시간**)에 따라 4 가지로 구분됩니다.

날짜 변수는 1900 년 이후의 날짜를 줄리언 날짜(Julian Date) 값으로 보관합니다.

시간 변수는 시간값으로 보관합니다.

 **변수형이 다른 변수를 형 변환할 때에는 문자열 변수가 우선입니다.**

2.2.3. 스크립트에서 변수 사용

스크립트에서의 변수 사용은 앞에서 설명한 변수를 그대로 사용할 수도 있고 **?, $, [] (배열 변수), .., #** 와 같은 **메타 문자**를 이용하여 보다 효과적으로 사용할 수도 있습니다. 메타 문자 사용 시 장점은 여러 개의 변수를 일일이 나열하지 않고 **한번에 표기**할 수 있다는 것입니다.

? 는 하나의 글자에 해당하는 메타 문자입니다. 예를 들어, a?라고 표기하면 a 로 시작하는 변수 중에서 a 다음에 오는 문자가 하나인 변수를 모두 가리킵니다.

**$**는 0 개 이상의 문자에 해당하는 메타 문자입니다. 예를 들어 a$라고 표기하면 a 를 포함하여 a 로 시작하는 모든 변수를 가리킵니다.예) ?와 $를 사용한 스크립트

```
a? = 10;

s = SUM (a$);
```

한편, 변수명이 **1** 또는 **1 보다 큰 숫자**로 끝나면 이를 **배열 변수**로 봅니다. 예를 들어 a1, a2, a3, ... , a10 은 모두 a 라는 이름을 가진 배열 변수입니다.

배열 변수는 **객체 각각**에 선언할 수 있고 **표의 셀**을 선택하여 한번에 선언할 수도 있습니다. 이 때, 변수형은 서로 달라도 무방합니다.

배열 변수로 선언되어 있으면 **“ ..“ 과 같은 메타 문자**를 사용하여 스크립트를 기술할 수 있습니다. **“ ..“** 는 앞뒤의 배열 변수 사이에 있는 모든 배열 변수를 가리킵니다. 예를 들어 m = avg(a1..a10); 은 a 로 시작하는 배열 변수 중에서 숫자가 1 과 10 사이에 있는 모든 배열 변수의 평균을 m 에 저장합니다.

2.3. 산술 연산식 (arithmetic expression)

산술 연산식은 **산술 연산자**를 이용하여 나타낸 식으로 다음과 같은 기호를 사용할 수 있습니다.

□ +

양쪽의 값을 더합니다. 또한 양수부호로 사용할 수 있습니다.

□ -

앞의 값에 뒤의 값을 뺍니다. 또한 음수부호로 사용할 수 있습니다.

□ *

양쪽의 값을 곱합니다.

□ /

앞의 값을 뒤의 값으로 나눕니다.

□ %

앞의 값을 뒤의 값으로 나눈 몫을 뺀 나머지입니다.

산술 연산식에서 **피연산자로 상수, 변수, 논리 연산식, 관계 연산식, 내장 함수**가 올 수 있으며 결과값은 **숫자**입니다.

예

```
a = 2 + 3 * 5;

b = (a - b) / 2 + (a + b) * 2;

순이익 = SUM (매출액$) - SUM(비용$);
```

2.4. 관계 연산식 (relational expression)

관계 연산식은 **관계 연산자**를 이용하여 나타낸 식으로 다음과 같은 기호를 사용할 수 있습니다. 관계 연산자은 피연산자의 대소 비교를 합니다.

□ <=

왼쪽의 피연산자가 오른쪽의 피연산자 보다 작거나 같으면 ‘ **참**’ 입니다. 오른쪽이 작다면 ‘ **거짓**’ 입니다.

□ <

왼쪽의 피연산자가 오른쪽의 피연산자 보다 작으면 ‘ **참**’ 입니다. 오른쪽이 작거나 같다면 ‘ **거짓**’ 입니다.

□ >=

왼쪽의 피연산자가 오른쪽의 피연산자 보다 크거나 같으면 ‘ **참**’ 입니다. 오른쪽이 크다면 ‘ **거짓**’ 입니다.

□ >

왼쪽의 피연산자가 오른쪽의 피연산자 보다 크면 ‘ **참**’ 입니다. 오른쪽이 크거나 같다면 ‘ **거짓**’ 입니다.

□ ==

양쪽의 피연산자가 같으면 ‘ **참**’ 입니다. 양쪽의 피연산자가 다르면 ‘ **거짓**’ 입니다.

□ !=

양쪽의 피연산자가 다르면 ‘ **참**’ 입니다. 양쪽의 피연산자가 같으면 ‘ **거짓**’ 입니다.

관계 연산식에서 **피연산자로 상수, 변수, 논리 연산식, 산술 연산식, 내장 함수**가 올 수 있습니다. 결과값은 ‘ **참**’ 또는 ‘ **거짓**’ 으로 나타나며 ‘ 참’ 인 경우, 1, ‘ 거짓’ 인 경우는 0 입니다.

피연산자로 문자 상수 또는 문자 변수가 올 수 있는데 피연산자가 모두 문자 상수이거나 문자 변수인 경우에는 **사전식 비교**를 합니다. 예를 들어, "report" 와 "designer"를 비교하면 사전에 "designer"가 먼저 나와 있으므로 "designer"가 더 작습니다. 주의할 비교는 "2"와 "100"의 비교인데 사전식 비교 결과는 "2"가 더 큽니다.

사전식 비교란 일반적으로 사전(dictionary)에 나열된 순서로 비교를 한다는 뜻입니다.

예

```
ret = (200 >= 100);
//ret 결과값: 참

ret = "Crownix Report" > "엠투소프트";
//ret 결과값: 거짓

a = 100;
b = "200";
ret = a < b;
//ret 결과값: 참

a = "Crownix Report";
b = 100;
ret = a < b;
//ret 결과값: 오류
```

2.5. 논리 연산식 (logical expression)

논리 연산식은 논리 연산자를 이용하여 나타낸 식으로, 다음과 같은 기호를 사용할 수 있습니다.

□ AND, &&

양쪽의 피연산자가 모두 ‘ 참’ 이어야 결과가 **‘ 참’** 입니다. 한 쪽이라도 ‘ 거짓’ 이면 결과는 **‘ 거짓’** 입니다.

□ OR, ||

양쪽에 있는 피연산자 중에서 한쪽이라도 ‘ 참’ 이면 결과는 **‘ 참’** 입니다. 양쪽 모두가 ‘ 거짓’ 이어야만 결과가 **‘ 거짓’** 입니다.

□ !

"!" 다음에 오는 피연산자의 **결과를 부정**합니다.

논리 연산식에서 **피연산자로 상수, 변수, 관계 연산식, 산술 연산식, 내장 함수**가 올 수 있습니다. 결과값은 **‘ 참’** 또는 **‘ 거짓’** 으로 나타나며 ‘ 참’ 인 경우, 1, ‘ 거짓’ 인 경우는 0 입니다.

예

```
a = 1;
b = 0;
c = 1;
d = 0;
ret = (a && b) || (c || d);
// ret 결과값: 참

a = 100;
b = 200;
c = 300;
ret = ((a < 200) AND (b == 200)) OR (c == 400);
// ret 결과값: 참

a = "ABC";
b = 100;
ret = a || (b == 100);
// ret 결과값: 오류
```

2.6. 문자열 연산식

□ &

양쪽의 문자열을 붙여 줍니다. 함수에서 strcat() 함수와 같은 역할을 합니다.

2.7. 복합 연산식

복합 연산식은 **피연산자가 연산식(산술 연산식, 관계 연산식 또는 논리 연산식)**으로 구성되어 있습니다. 연산자는 **산술 연산자, 논리 연산자, 관계 연산자 모두** 올 수 있습니다.

예

```
ret = ((a + b) > c) AND (k < b);

ret = (생년월일 > "1964/3/1") && (남자 && 병역필) &&
      (SUM (급여$) < 부양가족 * 일인당 생활비);
```

2.8. 연산자의 우선순위(priority)

연산자에는 ( ), !, +(양수 부호), -(음수 부호), *, /, %, &, +, -, > , >=, <, <=, ==, != , AND, &&, OR, || 등이 있습니다.

연산자 사이에는 다음의 표와 같은 우선 순위가 존재하며 **같은 행**에 있는 연산자는 우선순위가 없이 **사용된 순서대로** 처리됩니다.

연산자 우선순위는 다음 표와 같습니다.

연 산 자	우선순위
( )	높음
+(양수 부호), -(음수 부호)	
*, /, %, &	
+, -	
>, >=, <, <=, ==, !=	
AND, &&, OR, ||	낮음

2.9. 자료의 형 변환 규칙(Type Conversion Rule)

스크립트에서 상수나 변수의 형 변환 규칙은 다음과 같습니다.

2.9.1. 연산식/내장 함수에서 변수나 상수의 형 변환

연산식의 피연산자 또는 내장 함수의 인자로 사용되는 변수나 상수는 일반적으로 연산자 또는 내장 함수의 종류에 따라 적절한 형(type)으로 사용되어야 합니다. 만약 그렇지 않은 경우에는 자동으로 형 변환되어 처리됩니다.

연산식과 내장함수에 적용되는 형 변환규칙은 다음 표와 같습니다.

	숫자	문자	날짜	시간
숫자	숫자	숫자①	날짜②	시간③
문자		문자	날짜④	시간⑤
날짜			날짜	⑥
시간				시간

첫 번째 열은 연산자 또는 내장 함수의 종류에 따라 필요한 형을 나타내며 첫 번째 행은 사용자가 실제로 사용한 변수나 상수의 형을 나타냅니다. 번호가 매겨져 있는 셀의 경우에는 자동으로 형 변환이 일어납니다.

① 숫자와 문자의 연산은 문자를 숫자로 변환하여 연산합니다.

② 숫자와 날짜의 연산은 날짜를 줄리언 날짜(Julian Date)값으로 연산합니다.

③ 숫자와 시간의 연산은 시간을 시간값으로 연산합니다.

④ 문자와 날짜의 연산은 문자가 "년년년년/월월/일일"이나 "년년/월월/일일" 또는 "월월/일일"의 형식으로 되어 있다면 날짜로 변환하여 연산합니다.

⑤ 문자와 시간의 연산은 문자가 "시시:분분:초초" 나 "시시:분분" 의 형식으로 되어 있다면 문자를 시간으로 변환하여 연산합니다.

⑥ 날짜와 시간은 함께 혼용하여 사용할 수 없습니다.

위의 형 변환 규칙에 의해 변환되지 않는 경우에는 **"형 변환이 적절히 수행되지 않았습니다." 라는 오류 메시지**가 발생합니다. 이때에는 반드시 변수형을 사용 용도에 맞게 바꾸어야 합니다.

2.9.2. 대입문에 사용되는 변수나 상수의 형 변환

대입문은 **우변의 연산 결과를 좌변의 변수에 대입**하는 문장입니다. 대입문의 표현 형식은 **"변수 = 연산식 | 내장 함수;"**로 나타내어 집니다.

우변의 연산식 또는 내장 함수의 결과가 좌변 변수의 변수형과 맞지 않은 경우, **좌변 변수의 변수형을 기준으로 자동 형 변환**을 합니다.

3. 명령문

스크립트의 명령문(statement)에는 **지역 변수 선언문, 대입문, 제어문**이 있습니다. 모든 명령문은 특정 블록을 의미하는 문장 뒤를 제외하고는 모두 **; (세미콜론-semicolon)**으로 마쳐야 합니다. 특정 블록의 예로서는 ENDIF (IF 문의 끝), ENDFOR (FOR 문의 끝), ENDWHILE (WHILE 문의 끝), ENDSWITCH (SWITCH 문의 끝)문이 있습니다.

본문에서 명령문과 내장 함수를 설명 시 **[ ]**은 생략 가능하다는 뜻이며 **|** 는 이들 중에 선택적으로 하나 이상 사용할 수 있다는 뜻입니다. 또한 **문장**은 하나 이상의 명령문 또는 내장 함수를 말합니다.

3.1. 지역 변수 선언문

지역 변수란 스크립트 내부에서만 사용하는 변수입니다. 이러한 지역 변수의 선언문에는 CHAR, DATE, NUM, TIME 이 있습니다.

3.1.1. CHAR 문

문법

CHAR 변수목록;

설명

변수형이 문자인 지역 변수를 선언합니다. 배열 변수를 선언할 수도 있습니다.

CHAR 형 변수는 보통 다른 언어에서 사용하는 character 형 변수 즉, 하나의 문자를 의미하는 것이 아니라 string 을 의미합니다.

예

```
CHAR GRADE[10];

CHAR LEVEL;

CHAR A, B, C, D;
```

3.1.2. DATE 문

문법

DATE 변수목록;

설명

변수형이 날짜인 지역 변수를 선언합니다. 배열 변수를 선언할 수도 있습니다.

예

```
DATE today;

DATE days[10];
```

3.1.3. NUM 문

문법

NUM 변수목록;

설명

변수형이 숫자인 지역 변수를 선언합니다. 배열 변수를 선언할 수도 있습니다.

예

```
NUM 점수;

NUM 나이[12];

NUM rank_a, rank_b, rank_c;
```

3.1.4. TIME 문

문법

TIME 변수목록;

설명

변수형이 시간인 지역 변수를 선언합니다. 배열 변수를 선언할 수도 있습니다.

예

```
TIME now;

TIME event[10];
```

3.2. 대입문

대입문은 **어떤 연산의 결과를 변수**에 대입하는 명령문으로, 값을 가질 수 있는 모든 객체에 값을 부여하는 방법입니다. 예를 들어 변수명이 " 텍스트 상자 1" 이고 변수형이 문자열인 텍스트 상자 객체가 있다고 할 때, 텍스트 상자 1 = " 안녕하세요 " ; 라는 스크립트가 실행되게 되면 텍스트 상자 객체에 " 안녕하세요 " 가 입력되게 됩니다.

3.2.1. =

문법

변수 = 연산식|내장 함수;

설명

우변의 연산식 또는 내장 함수의 결과를 좌변 변수에 대입합니다.

예

회사명 = "엠투소프트";

number = 123;

greater = 성적 1 > 성적 2;

first = !first;

강$ = 0;

제일큰값 = MAX (값 1..값 10);

3.3. 제어문

제어문은 **조건에 따라 문장을 실행하거나 중단**하는 명령문을 말합니다. 제어문에는 BREAK, CONTINUE, IF-ENDIF, FOR-ENDFOR, RUN, SWITCH-ENDSWITCH, WHILE-ENDWHILE 문이 있습니다.

3.3.1. BREAK 문

문법

BREAK;

설명

스크립트의 실행을 중단할 때 또는 FOR/WHILE 문을 실행하는 도중에 조건절이 ' 참' 임에도 불구하고 FOR/WHILE 문을 멈추고 빠져나가려고 할 때 사용합니다. 또한, SWITCH 문에서는 해당되는 CASE 블록만을 수행하고 빠져 나오기 위해서 사용합니다.

예

```
char Msg;

FOR (Index = Last; Index >= Start; Index = Index - 1)
  IF (이름[Index] == "")
    CALL error( );
    BREAK;   // FOR 문을 빠져나간다.
  ENDIF
ENDFOR

IF (이름[Index] == "")
  Break;    // 실행중단
ENDIF
```

3.3.2. CONTINUE 문

문법

```
CONTINUE;
```

설명

반복 실행을 위한 제어문인 FOR-ENDFOR, WHILE-ENDWHILE 문의 내부에서 사용되며 반복 실행문에서 CONTINUE 문이 실행되면 CONTINUE 문 이하에 있는 문장을 수행하지 않고 반복 실행문의 최후로 이동합니다.

예

```
FOR (Index = 1; Index <= Last; Index = Index + 1)
  IF (이름[Index] == "")
    CONTINUE;
  ENDIF
ENDFOR

Index = 0;
WHILE ( Index < Last )
  Index = Index + 1;
  IF ( 이름[Index] == "")
    CONTINUE;
  ENDIF
ENDWHILE
```

3.3.3. IF-ENDIF 문

문법

```
IF (조건절)
   문장 1
[ELSE
   문장 2]
ENDIF [;]
```

설명

조건절이 ‘ 참’ 이면 문장 1 을 수행하고 ‘ 거짓’ 이면 문장 2 를 수행합니다.

조건절에는 논리 연산식 또는 관계 연산식을 사용할 수 있습니다.

문장 2 에는 IF 문을 계속해서 사용할 수 있으며 이 경우, 첫 번째 IF 문의 조건절을 만족하지 않으면 두 번째 IF 문의 조건절이 만족하는지 봅니다. 그리고 반드시 IF 문의 개수만큼 ENDIF 문이 와야 합니다.

예

```
IF ( 세금 > 100000 ) 등급 = "High";
  ELSE 등급 = "Low";
ENDIF

IF (점수 > 90) 학점 = "A";
  ELSE IF (점수 > 80) 학점 = "B";
    ELSE IF (점수 > 70) 학점 = "C";
      ELSE IF (점수 > 60) 학점 = "D";
        ELSE 학점 = "F";
      ENDIF;
    ENDIF;
  ENDIF;
ENDIF
```

3.3.4. FOR-ENDFOR 문

문법

```
FOR ([처음 실행절]; [조건절]; [나중 실행절])
   문장
ENDFOR [;]
```

설명

조건절이 ‘ 참’인 동안에 FOR 문 내부의 문장을 반복 실행합니다.

조건절에는 논리 연산식 또는 관계 연산식이 올 수 있으며 생략 가능합니다.

처음 실행절과 나중 실행절은 FOR 문이 반복되는 회수를 결정하는 조건이 됩니다.

처음 실행절과 나중 실행절에는 대입문, 제어문, 지역 변수 선언문 등이 올 수 있습니다.

예

```
FOR (직원 = 1; 직원 <= 총직원 ; 직원 = 직원+1)
  총액 = 총액 + 급여[직원];
ENDFOR

num i, j;
FOR (i = 1; ; i = i + 1)
  IF (:사원코드[i] == "")
    BREAK;
  ENDIF
  급여합 = 급여합 + 급여[i] + 상여금[i];
  FOR (j = 1; j <=사원가족수[i]; j = j + 1)
    가족수당 = 가족수당 + 일인당수당;
  ENDFOR
ENDFOR
```

3.3.5. SWITCH-ENDSWITCH 문

문법

```
SWITCH ( 연산식 )
CASE 상수:
   문장;
[CASE 상수:
   문장;]
[default:
   문장;]
ENDSWITCH [;]
```

설명

조건에 맞는 문장을 선택적으로 수행합니다. 즉, 연산식의 값이 CASE 의 상수와 일치하면 해당 CASE 절의 문장이 실행됩니다. IF 문은 조건에 따라 두개 중 하나를 선택하여 실행하는 것이지만 SWITCH 문은 연산식의 값에 따라 여러 개 중에서 하나를 선택하여 실행하는 것입니다.

예

```
SWITCH ( type )
CASE 101:
  total = (a+b) * 0.8;
  BREAK;
CASE 102:
  total = (a+b) * 0.9;
  BREAK;
DEFAULT:
   total = a + b;
ENDSWITCH
```

3.3.6. WHILE-ENDWHILE 문

문법

```
WHILE ( 조건절 )
    문장
ENDWHILE [;]
```

설명

조건절이 ‘ 참’ 인 동안에 WHILE 내의 문장을 반복 실행합니다. 조건절에는 논리 연산식 또는 관계 연산식이 올 수 있습니다.

예

```
직원 = 1;
총액 = 0;
WHILE ( 직원 <= 총직원 )
  IF ( 근무[직원] != 퇴직 )
    총액 = 총액 + 급여[직원];
  ENDIF
  직원 = 직원 + 1;
ENDWHILE
```

4. 내장 함수

내장 함수는 **명령문 내**에서 사용되며 **괄호 안에 인수를 포함**합니다. 인수는 내장 함수마다 각각 다르며 인수의 구별 표시는 **쉼표(,)**를 사용합니다. 내장 함수의 인수로서 내장 함수를 사용할 수도 있으며 **10 개까지 중복사용** 가능합니다.

4.1. 함수의 종류

Crownix Report 스크립트에서 제공하는 내장함수는 객체관련 함수, 숫자 관련 함수, 문자열 관련 함수, 날짜/시간 관련 함수, 데이터베이스 관련 함수, 객체그리기 및 속성관련함수 등이 있습니다. 이들 함수는 비정형적인 데이터의 편집, 계산 등을 쉽게 처리하여 개발자의 의도대로 프로그래밍된 보고서를 작성하도록 도와줍니다.

4.1.1. ADDDATABODY()

문법

ADDDATABODY(argstr);

인수

argstr
객체변수명을 포함하는 문자열, 반드시 “ ” 로 묶여있는 문자열 형태로 와야 합니다.
“ 객체변수명” : 일반적인 경우
“ EndAddDataBody” : 한 표에 대해서 표를 늘리는 것이 끝난 경우, 반드시 사용
“ 객체변수명(PageBreak)” : 반복부를 늘리면서 페이지도 넘기는 경우

반환값

숫자
1 : 항상 1 이 넘어옴

설명

스크립트 문서 즉, 스크립트 문서에서 반복지정된 행이 있는 표를 사용할 때, 표의 반복부 하나를 늘려 붙여주는 기능을 수행합니다.

보고서 형태 중 표 문서로 문서를 작성하는 경우에는 데이터베이스 서버로부터 데이터를 읽어올 때마다 표가 자동으로 커지면서 문서를 만들어 주게 됩니다. 스크립트를 사용하는 스크립트 문서는 반복지정된 행이 있더라도 표 문서처럼 표가 자동으로 커지는 형태가 아니라 스크립트 내에서 이 표를 키워주기 위한 함수가 필요하며 이것이 바로 ADDDATABODY( )입니다.

ADDDATABODY( )의 인수로는 반복지정된 행(반복부)에 부여된 객체 변수 중 하나를 사용합니다. 만일 한 페이지에 반복지정된 행이 있는 표가 2 개 이상 있고 이 함수를 사용하여 한 표의 행을 늘리다가 다른 표의 행을 늘려야 하는 경우에는 먼저 “ EndAddDataBody”를 인수로 넣어서 현재 늘리고 있는 표를 끝낸 다음에 다른 표의 변수 이름을 넣어주도록 합니다. 어떤 시점에서 조건을 만족하는 경우, 임의로 **페이지 넘기기**를 하기 위해서는 인수로 “ 변수명(PageBreak)”를 써줍니다.

예

```
num i, ret;
char oldee;

i = 1;
a = :a1;

cursor = "mycursor";
execfile declare :cursor cursor;
declareeor("\\eor");
```

```
declarefield(cursor, 1, 200);
declarefield(cursor, 1, 200);
declarefield(cursor, 1, 200);
declarefield(cursor, 1, 200);
declarefield(cursor, 1, 200);

while(1)
  ret = execfile fetch :cursor into :aa, :bb, :cc, :dd, :ee;
  if(ret != 1)
    AddDataBody("EndAddDataBody");   /* 현재 늘리고 있는 표 중지*/
    break;
  endif

  b = aa;
  c = bb;
  d = cc;
  e = dd;
  f = ee;
  if(i == 1)
    oldee = ee;
  endif

  if(i > 1 && strcmp(ee, oldee) != 0)
    AddDataBody("b(PageBreak)"); /* b 객체가 있는 반복부를  다음페이지로 넘겨서 출력 */
    oldee = ee;
  else
    AddDataBody("b");      /* b 객체가 있는 반복부 늘리기 */
  endif

  i = i+1;
endwhile

execfile close :cursor;
```

4.1.2. ADDDATALINE()

문법

ADDDATALINE(argstr);

인수

argstr
객체변수명을 포함하는 문자열
“ 객체변수명” : 반드시 “ ” 로 묶여있는 문자열 형태로 입력해야 합니다.

반환값

숫자형
1 : 항상 1 이 넘어옴

설명

스크립트 문서 즉, 스크립트 문서에서 반복지정된 행이 있는 표를 사용할 때, 반복이 적용된 행의 윗 선 속성으로 선을 그립니다.

표문서에서는 몇 행마다 선을 그리는 경우, 요약절을 사용합니다. 즉 요약절의 요약조건을 주고 요약작성을 주지 않으면 요약필드의 위의 선모양만 나옵니다. 그러나 스크립트를 사용하는 고정표 문서에서는 그러한 기능을 지원하기위해서 이 함수를 사용합니다. 반복속성이 있는 행 위의선을 적용시켜 원하는 기능을 사용할 수 있는 것입니다.

예

```
if(i == 5)
  AddDataBody("b");    /* b 객체가 있는 반복부를  출력 */
else
  AddDataBody("b");
  AddDataLine(“c”);  /* c 객체가 있는 반복절의 위의선을 출력 */
endif
```

4.1.3. AVG()

문법

AVG(arglist);

인수

arglist
평균값을 낼 변수들을 나타내는 인수 목록
a1,a2,a3 : a1, a2, a3 의 3 개의 변수
a1..a10 : a1 ~ a10 사이의 10 개의 변수

반환값

숫자형
넘어온 인수의 평균값

설명

인수 목록의 평균을 구합니다. 인수에는 상수, 변수 또는 연산식이 올 수 있으며 1 개 이상 올 수 있습니다.

예

```
ret = AVG (0,1,2,3,4,5,6,7,8,9);

ret = AVG (a, b, c);

ret = AVG (a + b, 36/5 - 1, c * d);

ret = AVG (a$, b?);

ret = AVG (a1..a10);

tot_avg =  AVG (SUM (국어$), SUM(영어$), SUM(수학$));
```

4.1.4. BLANK ()

문법

BLANK(varname);

인수

varname
서식이 있는 객체의 객체변수명

반환값

없음

설명

blank 는 미리 지정된 서식이 있는 객체 (예: 날짜를 표시하는 텍스트 상자 상자)의 서식을 제거하고 공란으로 만듭니다.

예

```
blank(date_var);
// 날짜 객체 date_var 의 "0000 년 00 월 00 일" 서식을 삭제하고 공란으로 만듭니다.
```

4.1.5. CHANGEATTR()

문법

CHANGEATTR(varname, code, value);

인수

varname
속성을 바꿀 객체변수

code
속성을 바꾸기위한 코드를 나타내는 문자열

value
속성을 바꾸기 위한 코드에 해당하는 수행값을 나타내는 문자열

반환값

없음

설명

텍스트 상자나 표의 셀의 속성(면속성, 폰트, 글자색상) 등을 수정합니다. 수행 코드에 따른 값을 속성으로 적용시킵니다. 보통 어떤 조건을 만족할 경우, 문자속성을 변경시키고자 할 때, 사용됩니다.

수행코드(code)가 여러 개라면 ' @' 문자를 구분자로 하여 각각의 코드를 이어주며 수행값(value) 또한 마찬가지입니다. 수행코드와 값은 다음과 같습니다.

code	설 명	Value	
FC	폰트색상	R,G,B(각각 0 ~ 255 사이의 숫자)	
FS	폰트크기	4 ~ 127 사이의 숫자	
FN	폰트명	폰트이름	
FA	폰트속성	B	굵게
		I	기울임
		U	밑줄
		S	가운데선
		R	반전
		DB	굵게 사용안함
		DI	기울임 사용안함
		DU	밑줄 사용안함
		DS	가운데선 사용안함
		DR	반전 사용안함
JP	장평	33 ~ 200 사이의 값, 기본크기는 100	
BC	배경색상	R,G,B(각각 0 ~ 255 사이의 숫자)	
BP	배경무늬	1	
		2	
		3	
		4	
		5	
		6	
		7	
LS	문단의 줄 수	문단의 줄 수를 나타내는 숫자	
HA	가로정렬	1	왼쪽 정렬
		2	오른쪽 정렬
		3	가운데 정렬
		4	양쪽 정렬
		5	배분 정렬
VA	세로정렬	1	상단 정렬
		2	중앙 정렬
		3	하단 정렬

[수행코드의 종류와 값]

예

```
changeattr (a, "FN", "굴림체");

changeattr (a, "BC", "256,256,0");

changeattr (a, "FA@FS", "B@15");
```

참고

CHANGEROWATTR()

4.1.6. CHANGEROWATTR()

문법

CHANGEROWATTR(varname, code, value);

인수

varname
속성을 바꿀 객체변수

code
속성을 바꾸기위한 코드를 나타내는 문자열

value
속성을 바꾸기 위한 코드에 해당하는 수행값을 나타내는 문자열

반환값

없음

설명

표의 행의 속성(면속성, 폰트, 글자색상) 등을 수정합니다.

특정값에 따라 표의 행 속성을 임의로 수정할 수 있도록 사용하는 함수입니다. 기본적으로는 changeattr( ) 함수와 사용하는 방법은 같지만 행 전체의 속성에 적용된다는 차이가 있습니다. 따라서 이 함수를 사용하면 varname 을 포함하는 표의 행 전체 속성을 바꾸어 주게 됩니다.

수행코드(code)가 여러 개라면 ' @' 문자를 구분자로 하여 각각의 코드를 이어주며 수행값(value) 또한 마찬가지입니다. 수행코드와 값은 CHANGEATTR() 함수를 참조하시길 바랍니다.

예

```
changerowattr (a, "FN", "굴림체");

changerowattr (a, "BC", "256,256,0");

changerowattr (a, "FA@FS", "B@15");
```

참고

CHANGEATTR(), CHANGETEXTATTR()

4.1.7. CHANGETEXTATTR()

문법

CHANGETEXTATTR(varname, inx, strsub, code, value, allflag)

인수

varname
적용할 객체 변수

inx
strsub 를 찾기 시작하는 varname 문자열내서의 위치를 나타내는 숫자값. 1 부터시작.

strsub
varname 문자열내에서 찾을 문자열 또는 문자열 변수

code
속성을 바꾸기 위한 코드를 나타내는 문자열

value
속성을 바꾸기 위한 코드에 해당하는 수행값을 나타내는 문자열

allflag
일치하는 모든 문자열을 바꿀것인지 처음 찾는 것만 바꿀것인지 나타내는 숫자
0: 처음 찾는 문자열만 속성을 바꿈
1: 일치하는 모든 문자열의 속성을 바꿈

설명

varname 문자열내에서 strsub 문자열을 찾아서 찾은 문자열의 문자 속성만 바꿔주는 함수입니다. changeattr 함수와 사용법이 유사하고 단지 찾을 문자열(str) 추가가 되어있습니다. str 값에 ""(빈문자열)을 주면 전체 문자열의 속성을 바꿉니다.

주의) varname 문자열 내에서 strsub 를 찾을 위치에 대한 값은 유니코드 버전인 경우와 일반 버전인 경우가 서로 다릅니다. 유니코드 버전에서는 문자의 위치 및 길이가 각 문자의 개수와 연관되며 일반 버전에서는 바이트 수와 연관됩니다. 한글 한 글자인 경우, 유니코드 버전에서는 길이가 1 이지만 일반 버전에서는 2 가 됩니다.

수행코드(code)가 여러 개라면 ' @' 문자를 구분자로 하여 각각의 코드를 이어주며 수행값(value) 또한 마찬가지입니다. 수행코드값은 다음과 같습니다.

code	설 명	Value	
FC	폰트색상	R,G,B(각각 0 ~ 255 사이의 숫자)	
FS	폰트크기	4 ~ 127 사이의 숫자	
FN	폰트명	폰트이름	
FA	폰트속성	B	굵게
		I	기울임
		U	밑줄
		S	가운데선
		R	반전
		DB	굵게 사용안함
		DI	기울임 사용안함
		DU	밑줄 사용안함
		DS	가운데선 사용안함
		DR	반전 사용안함
JP	장평	33 ~ 200 사이의 값, 기본크기는 100	

[수행코드의 종류와 값]

예

```
changetextattr(CustomerID, 1, "ATR", "FA", "B", 1);
// CustomerID 의 문자열에서 ATR 문자열을 모두 찾아서 글자를 굵게 바꿉니다.
```

참고

CHANGEATTR(), CHANGEROWATTR()

4.1.8. CONNECTDB()

문법

CONNECTDB (connectionid, ipaddr, port, service, dbname, userid, password);

인수

connectionid
connection identifier . ConnectDB() 함수 호출 후, 쿼리실행 등에서 사용할 id 문자열

ipaddr
연결할 데이터베이스 서비스의 IP 주소

port
연결할 데이터베이스 서비스의 포트번호

service
서비스의 이름

dbname
데이터베이스 서비스에서 생성된 DB 이름

userid
데이터베이스 서비스의 사용자 아이디

password
데이터베이스 서비스의 사용자 암호

반환값

숫자형
0 : 연결 실패
1 : 연결 성공

설명

한 문서에서 여러 개의 데이터베이스 서버에 접속할 수 있도록 합니다. 즉, 데이터베이스가 여러 곳의 서버로 분산되어 있을 때, 현재의 데이터베이스와 접속종료한 상태에서 다른 데이터베이스와 접속을 하는 함수입니다. 단, 데이터베이스 연결방법은 문서속성이 가지고 있는 연결방법과 동일해야 합니다.

CONNECTDB ( )를 사용하여 다른 데이터베이스 서버에 연결한 상태를 끊기 위해서는 DISCONNECT ( )함수를 사용하여 접속을 종료해야 합니다.

ODBC 를 이용하지 않고 Crownix Report 가 제공하는 데이터베이스(ORACLE, SYBASE, UNISQL 등) 직접 연결 라이브러리를 사용할 때만 이 함수를 사용할 수 있습니다.

모든 parameter 를 사용하는 것은 아니고 데이터베이스 서버의 특성에 따라 적절히 사용합니다.

데이터베이스 접속 시 사용되는 인자는 다음과 같습니다.

예

```
// ORACLE 직접접속
CONNECTDB (“orar”, “”, “”, “oraservice”, “”, “m2”, “m2soft”);

// SYBASE 직접접속
CONNECTDB(“syb”, “”, “”, “sybservice”, “m2db”, “m2”, “m2soft”);
```

구체적인 예는 EXECSQL 예제를 참조하시길 바랍니다.

참고

DISCONNECTDB ( ), EXECSQL

4.1.9. COS()

문법

COS(arg)

인수

arg
radian 형태의 숫자 또는 숫자형 변수

반환값

숫자형
코사인 값

설명

삼각함수 중에서 cosine(코사인)의 값을 구하는 함수입니다. 일반적인 사용보다는 수학적인 통계나 수치의 연산을 위해서 사용합니다. 인수는 radian 형태로 구성합니다.

예

```
num ret, PI;

PI = 3.141592;
ret = COS (PI/2);
ret = COS (SUM (a, b, c, d));
ret = COS (PI/3 – PI/6 + AVG (a, b, c, d));
```

참고

SIN(), TAN()

4.1.10. CURRENTDATE()

문법

CURRENTDATE ();

인수

없음

반환값

숫자형
현재날짜의 숫자로 변환한 줄리언 날짜값

설명

현재 시스템에 설정된 날짜를 줄리언 날짜 (Julian Date)로 돌려주는 함수입니다.

예

```
DATE  d;
num   cdate;

cdate = CURRENTDATE();
d = cdate + 10;  /* 10 일 이후의 날짜값 */
```

참고

CURRENTTIME(), DATE(), TIME()

4.1.11. CURRENTTIME()

문법

CURRENTTIME();

인수

없음

반환값

시간형
현재 시각의 시간값

설명

현재 시각의 시간값을 구하는 함수입니다.

예

```
time ctime

ctime = CURRENTTIME();      // ctime 은 TIME 형 변수
```

참고

CURRENTDATE(), DATE(), TIME()

4.1.12. DATE()

문법

DATE(julian, kind);

인수

julian
줄리언 날짜를 나타내는 숫자

kind
반환할 값의 종류
YEAR : 년
MONTH : 월
DAY : 일
WEEKDAY : 요일 (1: 일요일, 2: 월요일 ..., 7: 토요일)

반환값

숫자형
kind 값에 에 따라서 반환하는 값.

설명

줄리언 날짜값에서 원하는 값(년, 월, 일 또는 요일)을 추출합니다.

예

```
NUM Curdate;
NUM Day;

Curdate = CURRENTDATE();

iYear = DATE(curdate, YEAR);
iMonth = DATE(curdate, MONTH);
iDay = DATE(curdate, DAY);

Day = DATE(curdate, WEEKDAY);
if (Day == 1)
  sWeekDay = "Sunday";
endif
```

참고

CURRENTDATE(), CURRENTTIME(), TIME()

4.1.13. DECLAREEOR()

문법

DECLAREEOR(str);

인수

str
서브데이터의 구분을 나타내는 문자열. 기본값은 //EOR//

반환값

숫자형
0: 실패
1: 성공

설명

파일접속을 하면서 스크립트 사용 시 EOR (end of record)를 정의합니다.

파일접속 스크립트 문서에서 sub 필드를 지원하기 위해서는 DECLAREFIELD( )함수를 사용하여 필드에 대한 정의를 해주어야 합니다.

execfile fetch 함수로 데이터 값을 가져올 때, 각 sub 필드에 해당하는 레코드 셋의 끝을 나타내기 위해 데이터 파일에 end of record(EOR)를 표시해주어야 합니다. 이 때, 어떤 문자열을 EOR 로 표시할 것인지를 DECLAREEOR( )함수를 통해서 지정해줍니다. 이 함수를 호출하지 않아도 기본값으로 " //EOR//" 이 설정되어 있습니다.

예

DECLAREFIELD()의 예제 참조.

참고

DECLAREFIELD(), EXECFILE

4.1.14. DECLAREFIELD()

문법

DECLAREFIELD(cursorname, type, length) ;

인수

cursorname
서브데이터의 커서명

type
필드의 타입
1: CHAR
2: NUMERIC
9: DATE
10: TIME
12: VARCHAR (필드가 개행문자를 포함하고 있는 경우)

length
필드의 길이(해당 데이터가 표시될 수 있는 최대길이)

반환값

숫자형
0: 실패
1: 성공

설명

파일접속을 하면서 스크립트 사용 시 sub 필드에 해당하는 필드정의를 합니다.

파일 접속 스크립트 문서에서 표문서의 subpage 형식과 같은 문서를 작성할 때, sub 필드를 지원하기 위해서는 sub 필드에 해당하는 Field(필드)의 타입과 길이를 정의해야 합니다.

DECLAREFIELD 함수가 처음 호출되면 인자(커서명, 타입, 길이)를 가지고 첫 번째 필드의 정보를 커서에 알려줍니다. 두 번째 호출되면 두 번째 컬럼에 대한 정보를 알려주게 되며 fetch 함수를 호출하기 전에 모든 필드를 정의해주어야 합니다.

예

```
aaa = :a1;
bbb = :a2;
ccc = :a3;
ddd = :a4;
eee = :a5;

cursor = "mycursor";
```

```
execfile declare :cursor cursor;      // 커서 정의
declareeor("_EOR_") ;

declarefield(cursor, 1, 30);   // character 형 30 바이트 길이의 field 추가
declarefield(cursor, 1, 30);   // character 형 30 바이트 길이의 field 추가
declarefield(cursor, 1, 30);   // character 형 30 바이트 길이의 field 추가

while(1)
  // 필드를 3 개 선언했으므로
  ret = execfile fetch :cursor into :fff, :ggg, :hhh;
  if(ret != 1)   // end of record 를 만나면 return 0
    AddDataBody("EndAddDataBody");
    Break;
  endif
  AddDataBody("fff");
  xxx = :a2;
  yyy = :a3;
endwhile

// 마지막에는 커서를 닫는다.
execfile close :cursor;
```

참고

EXECFILE

4.1.15. DEFINECHART()

문법

DEFINECHART(chartname, axisname, dataname);

인수

chartname
차트 객체 변수명

axisname
차트의 축이름을 나타내는 문자열

dataname
축에 적용할 데이터를 나타내는 문자열

반환값

없음

설명

차트 객체에 필요한 정보를 정의합니다.

스크립트에서 차트객체에 대한 정의를 하기 위해서 사용합니다. 차트 객체 변수명으로 지정된 차트 객체의 X 축, Y 축, Z 축으로 들어갈 값에 해당하는 변수(데이터명)를 정의한 후, drawchart( )함수를 사용하여 각각의 변수값을 차트객체로 전달하고 차트를 그리도록 합니다.

definechart(), drawchart() 함수는 Report Designer 의 초기버전(3.0 이전)에서 제공하던 기능입니다. Crownix Report 에서 teechart 를 제공하면서 추가된 함수인 defineseries(), drawseries() 등의 함수를 사용하시길 바랍니다.

예

```
definechart(mychart, "X", "Products.ProductName");
definechart(mychart, "Y", "Categories.CategoryName");
definechart(mychart, "Z", "Products.UnitsInStock");
catename = :Categories.CategoryName;
cursor = "mycursor";

query = "SELECT C.CategoryName, P.ProductName, P.UnitPrice, P.UnitsInStock FROM
      Categories C, Products P
      WHERE C.CategoryID = P.CategoryID AND Categories.CategoryName = :catename
      ORDER BY C.CategoryName, P.ProductName";

execsql declare :cursor cursor for :query;
```

```
for(i=0; i<5; i=i+1)
   ret = execsql fetch :cursor into :tcname, :tpname, :tpunitprice, :tpstock;
   if (ret != 1)
      break;
   endif
   drawchart("mychart");  // mychart 객체를 그리기.
endfor

execsql close :cursor;
```

참고

DEFINESERIES(), DRAWSERIES(), SETSERIESTITLE()

4.1.16. DEFINESERIES()

문법

DEFINESERIES(chartname, seriesname, var1, var2, var3);

인수

chartname
차트 객체 변수명

seriesname
차트 객체가 가지고 있는 시리즈(Series)명을 나타내는 문자열. “ ” 로 묶여있어야 합니다.
Designer 에서 미리 생성해 놓은 Series. Viewer 가 실행되면서 생성되어 있는 Series 의 순서대로 “ S0”, “ S1”, ” S2” …로 이름이 재부여됩니다. Deleteseries() 함수의 호출로, Series 가 제거되면 Series 의 index 와 위의 이름에 있는 번호가 달라질 수 있습니다.

var1
Series 에 적용할 변수명을 나타내는 문자열. “ ” 로 묶여있어야 합니다.

var2
Series 의 가로 축에 적용할 변수명을 나타내는 문자열. “ ” 로 묶여있어야 합니다.

var3
보통은 “ ” (null string)을 입력하지만 변수명을 사용하는 경우에는 “ ” 로 묶여있어야 합니다.
변수명이 적용되어 있고 drawseries()가 호출될 때, 해당 변수로 이전에 나온 값과 다른 값이 나오면 Series 에 추가가 되며 이 값은 Legend 에 표시됩니다.

반환값

없음

설명

seriesname 시리즈에 적용할 데이터 변수명을 지정하는 함수입니다. var1 은 시리즈의 데이터 값, var2 는 시리즈가 참조하는 축의 데이터값입니다.

예

```
defineseries(mychart, "S0", "stock", "productname", "");
defineseries(mychart, "S2", "order", "productname", "");
deleteseries(mychart, "S1");

setseriestitle(mychart, "S0", "재고량");
```

```
setseriestitle(mychart, "S2", "주문량");

cid = :Categories.CategoryID;
category = :Categories.CategoryName;

cursor = "mycursor";
query = "select products.ProductName, products.UnitsInStock, products.UnitsOnOrder
        from Products where Products.CategoryID=:cid";

execsql declare :cursor cursor for :query;

first = 1;

while(1)
  ret = execsql fetch :cursor into :productname,:stock,:order;
  if(ret != 1 && first == 0)
    AddDataBody("EndAddDataBody");
    break;
  endif

  drawseries(mychart, "S0");
  drawseries(mychart, "S2");
  AddDataBody("productname");
  first = 0;
endwhile;

execsql close :cursor;
```

참고

DEFINESERIESXY(), DEFINESERIESXYS(), DRAWSERIES()

4.1.17. DEFINESERIESXY()

문법

DEFINESERIESXY(chartname, seriesname, x, y, label);

인수

chartname
차트 객체 변수명

seriesname
차트 객체가 가지고 있는 시리즈(Series)명을 나타내는 문자열. “ ” 로 묶여있어야 합니다.
Designer 에서 미리 생성해 놓은 Series. Viewer 가 실행되면서 생성되어 있는 Series 의 순서대로 “ S0”, “ S1”, ” S2” …로 이름이 재부여됩니다. Deleteseries() 함수의 호출로 Series 가 제거되면 Series 의 index 와 위의 이름에 있는 번호가 달라질 수 있습니다.

x
X 축에 적용할 변수명을 나타내는 문자열. “ ” 로 묶여있어야 합니다.

y
Y 축에 적용할 변수명을 나타내는 문자열. “ ” 로 묶여있어야 합니다.

label
label 로 사용할 변수명을 나타내는 문자열. “ ” 로 묶여있어야 합니다.

반환값

없음

설명

seriesname 시리즈에 적용할 x 축, y 축의 변수명을 지정하는 함수입니다. 특정 시리즈의 경우, Y 축의 데이터가 필요하기 때문에 defineseries() 함수로 지정할 수 없는 경우가 생깁니다. 이 때, defineseriesxy() 함수를 사용합니다.

예

```
char a1, a3;
char a2, a4;

defineseriesxy(mychart, "S0", "a1", "a2", "");
defineseriesxy(mychart, "S1", "a3", "a4", "");

a1 = 0;
a2 = 0;
```

```
a3 = 0;
a4 = 7.251;
drawseries(mychart, "S0");
drawseries(mychart, "S1");

a1 = 0;
a2 = 0;
a3 = 23.3;
a4 = 45.1;

drawseries(mychart, "S0");
drawseries(mychart, "S1");
```

참고

DEFINESERIES(), DEFINESEARIESXYZ()

4.1.18. DEFINESERIESXYZ()

문법

DEFINESERIESXYZ(chartname, seriesname, x, y, z, label);

인수

chartname
차트 객체 변수명

seriesname
차트 객체가 가지고 있는 시리즈(Series)명을 나타내는 문자열. “ ” 로 묶여있어야 합니다.
Designer 에서 미리 생성해 놓은 Series. Viewer 가 실행되면서 생성되어 있는 Series 의 순서대로 “ S0”, “ S1”, ” S2” …로 이름이 재부여됩니다. Deleteseries() 함수의 호출로 Series 가 제거되면 Series 의 index 와 위의 이름에 있는 번호가 달라질 수 있습니다.

x
X 축에 적용할 변수명을 나타내는 문자열. “ ” 로 묶여있어야 합니다.

y
Y 축에 적용할 변수명을 나타내는 문자열. “ ” 로 묶여있어야 합니다.

z
Z 축에 적용할 변수명을 나타내는 문자열. “ ” 로 묶여있어야 합니다.

label
label 로 사용할 변수명을 나타내는 문자열. “ ” 로 묶여있어야 합니다.

반환값

없음

설명

seriesname 시리즈에 적용할 x 축, y 축, z 축 변수명을 지정하는 함수입니다. 특정 시리즈의 경우, Y 축, Z 축의 데이터가 필요하기 때문에 defineseries() 함수로 지정할 수 없는 경우가 생깁니다. 이 때, defineseriesxyz() 함수를 사용합니다.

예

DEFINESERIESXY( )의 예제를 참조.

참고

DEFINESERIES(), DEFINESERIESXY()

4.1.19. DELETESERIES()

문법

DELETESERIES(chartname, seriesname);

인수

chartname
차트 객체 변수명

seriesname
차트 객체가 가지고 있는 시리즈(Series)명을 나타내는 문자열. “ ” 로 묶여있어야 합니다.
Designer 에서 미리 생성해 놓은 Series. Viewer 가 실행되면서 생성되어 있는 Series 의 순서대로 “ S0”, “ S1”, ” S2” …로 이름이 재부여됩니다. deleteseries() 함수의 호출로 Series 가 제거되면 Series 의 index 와 위의 이름에 있는 번호가 달라질 수 있습니다.

반환값

없음

설명

seriesname 시리즈를 제거하는 함수입니다. Designer 에서 보고서의 차트 디자인시에 다수의 시리즈를 만들어 놓았을 경우, Viewer 실행시에 사용하지 않는 시리즈를 없애는 기능을 합니다.

예

DEFINESERIES( )의 예제 참조.

참고

DEFINESERIES()

4.1.20. DELWSPACE()

문법

DELWSPACE(str, pos);

인수

str
공백을 제거할 문자열

pos
제거할 공백의 위치를 나타내는 숫자
-1: 문자열의 앞쪽에 있는 공백을 제거
-1 이 아닌경우: 문자열의 뒤쪽에 있는 공백을 제거

반환값

문자열
공백을 제거한 문자열

설명

문자열의 앞이나 뒤에 있는 공백 문자(탭, 스페이스, 리턴)를 제거합니다. 위치가 -1 이면 앞에서부터 위치가 그 외의 숫자이면 뒤에서부터 공백 문자를 제거합니다. 그러나 문자열의 중간에 위치한 공백 문자는 제거하지 못합니다.

예

```
a = "   White Space";
ret = DELWSPACE (a,-1);  // ret = "White Space"

a = "Blue Sky   ";
ret = DELWSPACE (a,1);   // ret = "Blue Sky"
```

4.1.21. DISCONNECTDB()

문법

DISCONNECTDB(connectionid);

인수

connectionid
ConnectDB() 함수에서 데이터베이스 접속을 위해 사용한 id 문자열

반환값

숫자형
0: 실패
1: 성공

설명

CONNECTDB() 함수를 사용해서 연결한 데이터베이스의 접속을 끊을 때, 사용하는 함수입니다. 데이터베이스가 여러 곳의 서버로 분산되어 있을 때, 다른 데이터베이스와 접속을 위해 현재 연결되어 있는 데이터베이스 서버와 접속을 종료하는 함수입니다.

CONNECTDB()를 사용하여 다른 데이터베이스 서버에 연결하려면 우선 기존에 연결되어 있던 서버를 DISCONNECT()함수를 이용하여 접속을 종료해야 합니다.

ODBC 를 이용하지 않고 Crownix Report 가 제공하는 데이터베이스(ORACLE, SYBASE, UNISQL 등) 직접 연결 라이브러리를 사용할 때만 이 함수를 사용할 수 있습니다.

예

```
DISCONNECTDB (“oras”);  // CONNECTDB( ) 함수에서의 첫 번째 parameter 를 사용
```

구체적인 예는 EXECSQL 의 예제를 참조하시길 바랍니다.

참고

CONNECTDB(), EXECSQL

4.1.22. DRAWCHART()

문법

DRAWCHART(chartname);

인수

chartname
    차트 객체 변수

반환값

없음

설명

차트 객체를 그리는 기능을 수행합니다.

각각의 X 축, Y 축, Z 축에 대해서 일치되는 변수를 definechart( ) 함수를 사용하여 정의한 후, drawchart( )함수를 사용하여 각각의 변수값을 차트객체로 전달하고 차트를 그리도록 합니다.

definechart(), drawchart() 함수는 Report Designer 의 초기버전(3.0 이전)에서 제공하던 기능입니다. Crownix Report 에서 teechart 를 제공하면서 추가된 함수인 defineseries(), drawseries() 등의 함수를 사용하시길 바랍니다.

예

DEFINECHART()의 예제 참조.

참고

definechart(), defineseries(), drawseries()

4.1.23. DRAWIMAGE()

문법

DRAWIMAGE(x, y, path, width, height, opt);

인수

x
이미지를 출력할 x 좌표를 나타내는 숫자.
좌표값의 단위 및 기본위치는 Crownix Report 가 사용하는 것을 따릅니다. 1 인치는 1000

y
이미지를 출력할 y 좌표를 나타내는 숫자

path
이미지의 경로를 나타내는 문자열

width
이미지의 폭을 나타내는 숫자

height
이미지의 높이를 나타내는 숫자

opt
옵션을 나타내는 숫자
0: 크기와 상관없이 이미지 생성
1: 크기에 비례하여 이미지 생성

반환값

없음

설명

path 의 경로에 있는 이미지파일을 보고서의 특정위치 x, y 에 특정한 크기 width, height 로 그려주는 함수입니다.

예

```
char path;

path = “http://www.m2soft.co.kr/testimage.jpg”;
drawimage(100, 100, path, 2000, 1000, 0);
```

4.1.24. DRAWLINE()

문법

DRAWLINE(x1, y1, x2, y2, command, value);

인수

x1
그리는 선의 시작위치 x 좌표를 나타내는 숫자.
좌표값의 단위 및 기본위치는 Crownix Report 가 사용하는 것을 따릅니다. 1 인치는 1000

y1
그리는 선의 시작위치 y 좌표를 나타내는 숫자

x2
그리는 선의 끝위치 x 좌표를 나타내는 숫자

y2
그리는 선의 끝위치 y 좌표를 나타내는 숫자

command
선의 화살표 모양 및 속성을 정의하는 문자열

value
command 에 따른 값을 나타내는 문자열

반환값

없음

설명

선 객체를 그리는 함수입니다.

이 함수를 호출 시에 적용하지 않는 속성에 대해서는 SETDRAWOBJATTR() 함수에서 정의한 속성을 따릅니다. 속성을 나타내는 명령(command)과 해당하는 명령에 대응하는 값(value)은 DRAWLINE() 함수를 참조하시길 바랍니다.

예

```
num x1, y1, x2, y2;

x1 = 1000;
y1 = 1000;
x2 = 5000;
y2 = 1000;
```

```
drawline(x1, y1, x2, y2, "LS@LT@LC", "2@1@255,0,0");
// 보고서의 상단에 길이 4 인치 정도의 빨간색 가로점선을 그린다.
```

4.1.25. DRAWSERIES()

문법

DRAWSERIES(chartname, seriesname);

인수

chartname
차트 객체 변수명

seriesname
차트 객체가 가지고 있는 시리즈(Series)명을 나타내는 문자열. “ ” 로 묶여있어야 합니다.
Designer 에서 미리 생성해 놓은 Series. Viewer 가 실행되면서 생성되어 있는 Series 의 순서대로 “ S0”, “ S1”, ” S2” …로 이름이 재부여됩니다. Deleteseries() 함수의 호출로 Series 가 제거되면 Series 의 index 와 위의 이름에 있는 번호가 달라질 수 있습니다.

반환값

없음

설명

defineseries() 함수에서 적용한 seriesname 시리즈에 적용되어 있는 데이터 변수명의 값을 가지고 시리즈를 그립니다. 한번 호출될 때마다 하나의 레코드가 차트에 적용됩니다.

예

DEFINESERIES( )의 예제 참조

참고

DEFINESERIES(), DELETESERIES()

4.1.26. DRAWTEXT()

문법

DRAWTEXT(x1, y1, x2, y2, text, command, value)

인수

x1
그리는 선의 시작위치 x 좌표를 나타내는 숫자.
좌표값의 단위 및 기본위치는 Crownix Report 가 사용하는 것을 따릅니다. 1 인치는 1000

y1
그리는 선의 시작위치 y 좌표를 나타내는 숫자

x2
그리는 선의 끝위치 x 좌표를 나타내는 숫자

y2
그리는 선의 끝위치 y 좌표를 나타내는 숫자

text
텍스트 상자의 내용 문자열

command
텍스트 상자의 속성을 정의하는 문자열

value
command 에 따른 값을 나타내는 문자열

반환값

없음

설명

텍스트 상자를 그리는 함수입니다. 이 함수를 호출 시에 적용하지 않는 속성에 대해서는 SETDRAWTEXTATTR()에서 정의한 속성을 따릅니다. 속성을 나타내는 명령(command)과 해당하는 명령에 대응하는 값(value)은 SETDRAWTEXTATTR()함수를 참조하시길 바랍니다.

예

```
num x1, y1, x2, y2;

x1 = 1000;
y1 = 1000;
x2 = 5000;
```

```
y2 = 1500;
drawtext(x1, y1, x2, y2, "TEST STRING", "FN@FS", "Tahoma@20");
// 보고서의 상단에 길이 4 인치 정도의 텍스트 상자를 그린다.
```

4.1.27. DRILLDOWN()

문법

DRILLDOWN(mrdpath, param);

인수

mrdpath
drilldown 으로 나타날 양식을 구성하는 mrd 파일

param
mrd 를 열 때 사용하는 Crownix Report 파라미터 옵션
추가 파라미터
/rdrillright : 아래쪽에 붙지 않고 하이퍼링크 객체의 오른쪽에 나타납니다
/rdrillover : 아래쪽이나 오른쪽에 붙지 않고 호출하는 객체의 위쪽으로 나타납니다.

반환값

숫자형

설명

결과보고서에서 drilldown 기능을 필요로 할 때 사용합니다. 한번 DRILLDOWN() 한 객체에 다시 DRILLDOWN()이 호출되면 DRILLUP() 기능을 수행합니다.

스크립트 보고서에서의 스크립트에서는 사용할 수 없으며 하이퍼 객체의 스크립트 기능에서만 사용합니다. 하이퍼링크 객체의 스크립트에서만 사용할 수 있는 함수로는 DRILLDOWN(), DRILLUP(), SORTDATA() 함수 등이 있습니다.

예

```
drilldown("제품상세매출액.mrd", "/rp [1/4 분기] /rdrillright ");
/* 오른쪽으로 펼쳐지도록 */
```

참고

DRILLUP()

4.1.28. DRILLUP()

문법

DRILLUP();

인수

없음

반환값

없음

설명

DRILLDOWN()을 사용해서 삽입된 객체를 이 함수를 사용해서 삭제합니다. 한번 DRILLDOWN()한 객체에 다시 DRILLDOWN()이 호출되면 DRILLUP() 기능이 수행되므로 별도로 사용하지 않아도 됩니다.

스크립트 보고서에서의 스크립트에서는 사용할 수 없으며 하이퍼 객체의 스크립트 기능에서만 사용합니다. 하이퍼링크 객체의 스크립트에서만 사용할 수 있는 함수로는 DRILLDOWN(), DRILLUP(), SORTDATA() 함수 등이 있습니다.

예

DRILLDOWN() 함수 예제 참조

참고

DRILLDOWN()

4.1.29. EXECFILE

문법

EXECFILE DECLARE <cursor name> cursor

EXECFILE FETCH <cursor name> INTO <comma separated host variable list>;

EXECSQL CLOSE <cursor name>;

인수

<cursor name>
커서의 이름을 나타내는 문자열 변수.
DELARE 문에서 커서를 지정하여 CLOSE 문에서 커서를 닫습니다.

<comma separated host variable list>
주 데이터라면 필드정의에 선언된 필드 개수만큼의 데이터 변수 리스트
서브데이터라면 declarefield() 함수가 수행된 개수만큼의 데이터 변수 리스트

반환값

숫자형
1 : 성공
그외 : 실패

설명

스크립트 문서에서 파일접속을 하였을때, 접속한 파일의 데이터를 스크립트에서 처리하기 위하여 사용합니다.

DECLARE 문에서 쿼리 문장과 일치시키는 부분이 없다는 것을 제외하면 DB 접속 문서의 EXECSQL 과 완전히 일치합니다. EXECSQL 이 직접 데이터베이스에서 자료를 가져오고자 한다면 EXECFILE 은 어떤 텍스트 파일로부터 자료를 바로 읽어 들이기 때문에 쿼리 문장이 필요가 없는 것입니다.

EXECFILE 을 사용하여 텍스트 파일로부터 자료를 가져오는 로직은 다음과 같습니다.

DECLARE 문은 커서를 텍스트 파일(데이터 파일)의 첫 레코드로 위치시킵니다. FETCH 문은 커서를 한 행씩 내리면서 데이터 파일로부터 레코드를 하나씩 가져오고 커서가 데이터 파일의 마지막 레코드를 만날 때 까지 내려가서 FETCH 문의 실행이 끝나게 되면 CLOSE 문은 커서를 종료시키는 것이 됩니다.

EXECSQL 은 여러 개의 커서를 동시에 가지고 있을 수 있으나(multi-query), EXECFILE 을 사용하는 문서는 하나의 텍스트 파일의 첫 레코드에서 마지막 레코드까지 순서대로 가져오기 때문에 한 번에 하나의 커서만 존재하게 됩니다.

Master-detail 구조나 서로 다른 필드 구조를 갖는 여러 레코드 셋을 출력하는 문서(표 문서에서의 서브페이지 문서)를 작성하는 경우에는 declarefield( )함수를 사용해서 각각의 레코드 셋에 대한 필드정의(sub 필드정의 에 해당)를 해 주어야 합니다. 이렇게 선언한 커서는 FETCH 문장에서 EOR(end of record) 를 만날 때 까지 데이터를 가져옵니다.

예

DECLAREFIELD()의 예제 참조

참고

EXECSQL, DECLAREFIELD(), DECLAREEOR()

4.1.30. EXECSQL

문법

EXECSQL DECLARE <cursor name> CURSOR FOR <query> [ON <db server>];

EXECSQL FETCH <cursor name> INTO <comma separated host variable list>;

EXECSQL CLOSE <cursor name>;

EXECSQL COMMAND <query> [ON <db server>];;

인수

<cursor name>
커서의 이름을 나타내는 문자열 변수.
DELARE 문에서 커서를 지정하여 CLOSE 문에서 커서를 닫습니다.

<query>
데이터를 가져오기 위한 쿼리문을 포함하는 문자열 변수

<db server>
ConnectDB() 함수를 사용하여 다른 데이터베이스 서비스에 연결한 connection id

<comma separated host variable list>
<query>문의 select 문에 있는 필드 개수만큼의 데이터 변수 리스트

반환값

숫자형
1: 성공
그외: 실패

설명

SQL 문장의 실행을 위해서 커서의 선언, 데이터의 fetch, 커서의 종료 등을 수행합니다. 이 구조는 오라클의 PL/SQL 의 구문과 흡사합니다.

DECLARE 문은 쿼리 문장을 커서(cursor)에 일치시킵니다. 이 문장을 선언하지 않으면 FETCH 문장을 실행할 수가 없습니다.

FETCH 문은 Query 문의 실행을 통해 데이터베이스로부터 데이터를 변수 (객체 변수, 지역 변수, 임시 변수)로 가져옵니다. 하나의 FETCH 문장이 실행될 때마다 하나의 레코드만 가져오므로 다음 데이터를 가져오고자 할 때에는 FETCH 문을 한번 더 실행해야 합니다.

CLOSE 문은 DECLARE 문장에서 선언한 커서를 종료합니다.

SQL 문을 실행시킨 결과 레코드 테이블을 놓고 다시 한 번 설명하면 다음과 같습니다.

DECLARE 문은 커서를 결과 레코드 테이블의 첫 레코드에 위치시킵니다. FETCH 문은 커서를 한 행씩 내리면서 쿼리결과 테이블로 부터 레코드를 하나씩 가져오도록 합니다. 커서가 쿼리결과 테이블의 마지막 레코드까지 내려가서 FETCH 문의 실행이 끝나게 되면 CLOSE 문은 커서를 종료시킵니다.

FETCH 문의 수행 결과 성공이면 1, 더 이상 가져올 데이터가 없으면 0, 오류가 발생했으면 -1 을 각각 반환합니다. DECLARE 와 CLOSE 문은 수행 결과가 성공이면 1, 오류가 발생했으면 0 을 각각 반환합니다.

COMMAND 문은 쿼리문장을 바로 실행시킵니다. Insert, update 등의 쿼리문을 실행할 때 사용하며 객체의 하이퍼링크 속성을 이용해서 사용하면 유용합니다. 수행결과가 성공이면 1, 오류가 발생했으면 0 을 반환합니다.

분산된 데이터베이스 서버 환경에서 다른 데이터베이스 서버의 데이터 값을 읽어오기 위해서는 DECLARE, COMMAND 문의 맨 끝에 ON <db server>를 붙여주어서 특정 데이터베이스 서버와 연결하는 cursor 를 만들어 주어야 합니다. 여기에서 <db server>는 CONNECTDB( )를 사용하여 만들어진 connection handle 입니다.

주의) SQL 문에서 host variable 에 색인을 사용할 수 있습니다. 그러나 현재는 num 형의 숫자와 변수로 쓰여지는 identifier 만을 인식하며 색인에 expression 이나 function 을 사용해서는 안됩니다.

예

□ 예제 1

```
num employee, num I, tord;
date tgord;
num ret_m, ret_s, tunit, tquan, tprod;

kname = :Employees.FirstName;
ename = :Employees.LastName;
sex = :Employees.TitleOfCourtesy;
phone = :Employees.HomePhone;
address = :Employees.Address;
employee = :Employees.EmployeeID;

/* 데이터베이스-Query 정의 메뉴를 통해 가져온 데이터를 대입문을 통해 변수로 입력하는 경우에는 위에서 보이는 것처럼 컬럼명 앞에 :(콜론)을 붙여줍니다. */

/* cursor, query 는 통신을 위한 SQL 의 호스트 변수로 사용 */
cursor = "mycursor";
query = "select orders.OrderID, orders.OrderDate, orders.CustomerID from orders
        where orders.EmployeeID = :employee";
```

```
/* 쿼리 문장의 조건절에 스크립트 내부 변수를  사용하는 경우에도 변수명 앞에 :(콜론)을 붙여줍니다. 쿼리문의
실행 시에는 employee 변수가 가지고 있는 값으로 바꾸어서 실행하게  됩니다. */

execsql declare :cursor cursor for :query;

for(i=1; i<11; i=i+1)
  ret_m = execsql fetch :cursor into :tord, :tgord, :tcustomerid;

  if (ret_m != 1)
    break;
  endif

  /* host 변수인 :tord 를 이용하여 동적 질의어 생성이 가능합니다. */
  cursor1 = "mycursor1";
  query1 = "select O.UnitPrice, O.Quantity, O.ProductID from [order details] O
          where O.orderid = :tord";

  execsql declare :cursor1 cursor for :query1;
  ret_s = execsql fetch :cursor1 into :tunit, :tquan, :tprod;
  if (ret_s != 1)
    execsql close :cursor1;
    continue;
  endif

  /* 중요 : 색인을 이용한 변수를 sql 내부에 사용할 수 있지만 오류/자료의 끝에
     발생할 수 있는 잘못된 값을 출력하지 않기 위해 fetch 가 성공적으로 끝난
     직후 객체 변수를 갱신합니다. */
  orderid[i] = tord;
  gorderdate[i] = tgord;
  unitprice[i] = tunit;
  quantity[i] = tquan;
  product[i] = tprod;
  total[i] = tunit * tquan;

  /* loop 내에서 커서를 재사용하기 위해서는 반드시 execsql close 문으로
     닫아야 합니다. */
  execsql close :cursor1;
endfor

execsql close :cursor;
```

□ 예제 2

```
num I;
date tgord;
num ret_m, ret_s;
num count;

dbname1 = “mydb1”;

/* Oracle 직접접속에서는 connectdb 함수에서 ServiceName, UserID, Password 만
  입력하면 됩니다. */
Connectdb(dbname1, “ “, “ “, “halla”, “ “, “scott”, “tiger”);
Cursordb = “mycursordb”;
Querydb = “select kwnam, chasu, addr, kwcod, sogwan from cdkikw_f”;
/* 분산된 데이터베이스 접속을 위해서 DECLARE 문으로 쿼리에 대한 커서 생성 시에
  connectdb 함수에서 연결한 connection name 을 끝에다가 붙여줍니다.(on :dbname1)
*/
execsql declare : cursordb cursor for :squerydb on :dbname1;
dbname2 = “mydb2”;
connectdb(dbname2, ““, ““, “baekdu”, ““, “scott”, “tiger”);

cursordb2 = “mycursordb2”;
querydb2 = “select s_name, s_num, s_addr, s_code, s_birth from student”;
execsql declare :cursordb2 cursor for :querydb2 on :dbname2;

while(1)
  count  = count + 1;
  if(count == 10)
    break;
  endif

  ret_s = execsql fetch :cursordb into :f1, :f2, :f3, :f4, :f5;
  if (ret_s != 1)
    AddDataBody(“EndAddDataBody”);
    break;
  endif

  ret_s = execsql fetch :cursordb2 into :f6, :f7, :f8, :f9, :f10;
  if (ret_s != 1)
    AddDataBody(“EndAddDataBody”);
    break;
  endif
  AddDataBody(“f2”);
Endwhile

// cursor 를 닫습니다.
execsql close :cursordb;
execsql close :cursordb2;
```

```
// 연결된 데이터베이스를 닫습니다.
disconnectdb(dbname1);
disconnectdb(dbname2);
```

4.1.31. FORMAT()

문법

FORMAT(varname, str, opt)

인수

varname
특정 서식 문자열로 바꿀 변수. 변수형은 숫자, 날짜, 시간형만 지원합니다.

str
서식을 나타내는 문자열

opt
의미없는 숫자표시 여부를 나타내는 숫자
0 : 의미없는 숫자 표시. 예) 2005-12-04
1 : 의미없는 숫자 표시안합니다. 예) 2005-12- 4

반환값

문자열
서식형태로 변환한 문자열

설명

변수명의 값을 서식 문자열의 형태로 바꾸어 줍니다. 리턴되는 값은 서식 문자열 형식으로 변환된 문자열입니다. 보통 서식맞추기 대화상자에서 대부분의 서식은 모두 설정할 수 있으나 서식맞추기 대화상자에서 설정하지 않고 이 함수를 사용하는 경우는 특정 서식을 나타내는 데이터와 다른 데이터를 동시에 한 곳에 표시하고자 할 때입니다. 서식을 문자열로 바꾸어 다른 데이터와 & 연산자 등을 사용하여 문자열을 이어주는게 대표적인 예입니다.

opt 의 ‘ 의미없는 숫자’의 의미는 날짜형 데이터의 변환된 결과값이 “ 2005-09-10” 일 때, 의미없는 숫자를 표시 안하게 되면 “ 2005- 9-10”으로 결과값을 리턴하고 숫자인 경우에는 소수점이하의 맨 뒤에 나오는 ‘ 0’을 없애줍니다.

varname 이 숫자형인 경우, 서식 문자열인 str 은 서식맞추기 대화상자에서 설정할 수 있는 서식문자열만 사용할 수 있습니다.

예

```
ret = format(dateval, “yyyy-mm-dd”, 0)
// 날짜형 데이터 dateval 가 ‘2006 년 01 월 30 일’이라면
//결과값: ret == “2006- 1-30”

ret = format(numval, “#,###.###”, 1)
// 숫자형 데이터 numval 가 1234.56 라면 결과값: ret == “1,234.560”
```

```
ret = format(timeval, “hh:mm:ss”, 1)
// 시간형 데이터 timeval 가 ‘오후 12 시정각’ 이라면 결과값: ret = “12:00:00”
```

참고

4.1.32. GETGLOBAL()

문법

GETGLOBAL(varname);

인수

varname
global 변수이름을 나타내는 문자열. 반드시 “ ” 로 묶여있어야 합니다.

반환값

문자열
global 변수의 값

설명

특정 global 변수값에서 값을 읽어오는 함수입니다.

스크립트 보고서에서는 주 데이터의 각 레코드마다 스크립트가 실행됩니다. 실행되는 각각의 스크립트는 서로 독립적인 변수(객체, 지역, 임시변수)를 가지고 있습니다. 실행되는 스크립트의 특정한 결과값을 다음 레코드에 의해 실행되는 스크립트에 반영하고자 하는 경우에는 데이터를 전달하는 방법이 global 변수를 사용하는 것 밖에 없습니다. 데이터를 설정하는 것이 SETGLOBAL() 함수이고 데이터를 얻어오는 것이 GETGLOBAL() 함수입니다.

예

```
char dname, new_dname;

dname = getglobal(“g_dname”);

…
…
if (new_dname != “”)
  setglobal(“g_dname”, new_dname);
endif
```

참고

SETGLOBAL()

4.1.33. GETPARAM()

문법

GETPARAM (arg);

인수

arg
파라미터로 입력된 변수명을 나타내는 문자열

반환값

문자열
arg 변수의 값

설명

작성한 문서를 Viewer 로 실행시킬 때 /rp, /rv 파라미터로 넘겨주는 값을 스크립트 내로 읽어 들이는 기능을 수행합니다.

GETPARAM 의 인수는 Viewer 를 실행시킬 때, 넘겨받는 파라미터 변수가 됩니다. 즉, /rp 또는 /rpn 파라미터로 값을 넘겨 받을 때는 $1, $2,… 가 되고 /rv 파라미터로 값을 넘겨받을 때는 /rv 파라미터에서 정의한 변수명이 됩니다.

만일 해당 인수가 발견되지 않으면 수행중 오류가 발생합니다. 파라미터 정의는 ActiveX Viewer 를 사용하는 경우에는 FileOpen() 메소드의 두 번째 argument 에서 이루어지고 EXE Viewer 를 실행하는 경우에는 실행 파라미터에서 이루어집니다.

예

```
CHAR s1;

s1 = GETPARAM (“$1”);          /* s1 의 값은 $1 에 해당하는 문자열*/
```

4.1.34. GOTOPAGE()

문법

GOTOPAGE(pageno);

인수

pageno
    페이지번호를 나타내는 숫자

반환값

없음

설명

DRAWLINE(), DRAWTEXT() 함수를 가지고 보고서를 생성할 때, 현재페이지에서 다른 페이지로 이동하는 경우에 사용합니다.

참고

DRAWLINE(), DRAWTEXT()

4.1.35. LIBFREE()

문법

LIBFREE(filename);

인수

filename
LIBLOAD()에서 로딩한 DLL 파일명

반환값

없음

설명

Crownix Report 는 스크립트 내에서 외부 library (Dynamic Linking Library: 동적연결라이브러리, DLL)를 사용할 수 있는 기능을 제공하고 있습니다. 이 기능은 사용자가 만든 임의의 library 를 자유자재로 사용할 수 있다는 점에서 매우 유용하면서도 강력한 기능이라고 할 수 있습니다. 제공되는 함수로는 LIBLOAD(), LIBFREE ()가 있습니다.

LIBFREE()함수는 load 한 DLL (Dynamic Linking Library) 파일을 free 시킵니다.

filename 에는 파일명을 주어야 하는데 LIBLOAD 함수에서 인수로 쓰인 파일명과 반드시 일치해야 합니다. 일반적으로 사용되는 모든 DLL 파일을 수용할 수 있지는 않고 Crownix Report 에서 사용할 수 있는 형태로 변경해주어야 합니다.

예

□ **스크립트 예제**

```
num n;
num ret;
char libname;
char s;

libname = "e:\testdll\release\example.dll";
Libload(libname);

ret = Hello(s, n);
messagebox(s, "string", OK);
messagebox(n, "count number", OK);

ret = Hello(s, n);
messagebox(n, "count number", OK);

ret = Hello(s, n);
```

```
messagebox(n, "count number", OK);

LibFree(libname);
```

□ C 로 작성된 EXAMPLE.DLL 의 소스 (Example.c)

예로 나온 dll 은 Microsoft Visual C++ 5.0 에서 complile 한 것입니다.

```
/* ----------------------------------------------------------
*  Crownix Report  Example DLL  *
*  FILE : Example.C  *
*  Author : M2Soft  *
*  Date : 1998.05.01. by T.H. Kim  *
*  Compiler : Microsoft Visual C++ 5.0  *
---------------------------------------------------------- */

#include <windows.h>
#define DllExport __declspec( dllexport )

/* ----------------------------------------------------------
 *  << Script >>  << Example.c >>
 *  hello(str, num)  --->  _M_HELLO(int argc, char *argv [])
 *       ~~~  ~~~  argc == 2;
 *  argv [0] --> "rdviewer.exe"
 *  argv [1] --> str
 *  argv [2] --> num
---------------------------------------------------------- */

DllExport double _M_HELLO  (int argc, char *argv [])
{
  static int count = 0;
  double dret;
  double *pNum;

  if (argc != 3)
    dret = -1;
  else
  {
    strcpy(argv [1], "Hello World!");
    pNum = (double *)argv [2];
    *pNum = (double)count++;
    dret = 0;
  }
  return dret;
}
```

□ Example.def

```
;/* ------------------------------------------------------
;*  Crownix Report  Example DLL
;*  FILE : Example.C
;*  Author : M2Soft(Millenium Master)
;*  Date : 1998.05.01. by T.H. Kim
;*  Compiler : Microsoft Visual C++ 5.0
;------------------------------------------------------ */

LIBRARY Example

CODE   PRELOAD MOVEABLE DISCARDABLE
DATA   PRELOAD SINGLE

EXPORTS

_M_HELLO
```

1. 스크립트에서 LIBLOAD 를 실행한 후에 library 내의 함수를 호출할 때, 반드시 return 되어지는 값은 num 형 변수에 대입해주어야 합니다.

2. DLL 작성 시 스크립트에서 사용한 함수가 hello 라면 반드시 _M_HELLO 형태가 되어야 합니다. 즉, 모두 대문자가 되어야 하고 함수이름 앞에 " _M_" 가 있어야 합니다. 반드시 대문자가 되어야 하는 이유는 스크립트 상에서는 모든 이름에 대해서 대소문자 구분이 없으며 실제로 library 상에 존재하는 함수이름은 대소문자를 구별하기 때문에 혼용을 방지하기 위함입니다.

3. DLL 의 함수를 정의할 때에는 반드시 두개의 인수만 사용합니다.
int argc : 인수의 개수를 나타냅니다. 스크립트에서 호출할 당시의 인수의 개수보다 항상 1 이 큽니다.
char *argv[] : 스크립트에서 함수의 인수가 전달되는 방식으로 argv[0]은 항상 " rdviewer.exe" 이며 argv[1]부터 차례대로 인수가 전달됩니다.
모든 인수가 포인터(pointer) 형태로 전달되기 때문에 참조에 의한 호출(call by reference) 이 가능해 집니다.

4. DLL 함수로 부터 strcpy 등으로 string 을 넘겨 받을 경우에는 256 바이트를 넘지 않아야 합니다.

참고

LIBLOAD()

4.1.36. LIBLOAD()

문법

LIBLOAD(filename);

인수

filename
로딩할 DLL 파일명

반환값

없음

설명

Crownix Report 는 스크립트 내에서 외부 library (Dynamic Linking Library: 동적연결라이브러리, DLL)를 사용할 수 있는 기능을 제공하고 있습니다. 이 기능은 사용자가 만든 임의의 library 를 자유자재로 사용할 수 있다는 점에서 매우 유용하면서도 강력한 기능이라고 할 수 있습니다. 제공되는 함수로는 LIBLOAD(), LIBFREE ()가 있습니다.

LIBLOAD()함수는 DLL (Dynamic Linking Library) 파일을 load 합니다. load 해야할 DLL 을 작성하는 방법과 예제는 LIBFREE() 함수를 참조하시길 바랍니다.

예

LIBFREE() 함수 참조

참고

LIBFREE ( )

4.1.37. LOADIMAGE()

문법

LOADIMAGE(varname, path);

인수

varname
이미지를 삽입할 객체변수명

path
이미지 파일의 경로

반환값

없음

설명

객체에 이미지 파일을 읽어서 보여줍니다.

varname 의 객체변수명이 부여된 객체에 파일경로에 있는 이미지파일을 읽어서 보여줍니다. 이미지 파일의 경로는 로컬경로, HTTP URL 경로 등을 지원합니다.

예

```
char path;

path = “c:\temp\image.jpg”;
loadimage(a, path);  // a 객체에 이미지파일 읽기
```

4.1.38. LOG()

문법

LOG (num);

인수

num
자연로그 값을 구할 숫자
0 이면 안됨

반환값

숫자형
num 의 자연로그 값

설명

인수의 자연로그 값을 구합니다. 일반적인 사용보다는 수학적인 통계나 수치의 연산을 위해서 사용합니다. 인수는 0 이 될 수 없습니다.

예

```
ret = LOG (2.717);

ret = LOG (b/3.14);
```

참고

LOG10()

4.1.39. LOG10()

문법

LOG10(num);

인수

num
밑이 10 인 로그 값을 구할 숫자 또는 숫자형 변수
0 이 되면 안됨

반환값

숫자형
num 의 밑이 10 인 로그값

설명

인수의 밑이 10 인 로그 함수 값을 구합니다. 일반적인 사용보다는 수학적인 통계나 수치의 연산을 위해서 사용합니다. 인수는 0 이 될 수 없습니다.

예

```
y = LOG10 (2.717);

y = LOG10 (b/3.14);
```

4.1.40. MATCH()

문법

MATCH (str, strsub);

인수

str
문자열 또는 문자열 변수

substr
찾을 문자열 또는 문자열 변수

반환값

숫자형
찾은 문자열의 위치
-1 : 못 찾음
> 0 : 찾은 위치

설명

str 문자열 내에 strsub 와 동일한 문자열이 포함되어 있는지 조사합니다. 결과값은 str 에 strsub 가 포함되어 있지 않다면 -1, 포함되어 있으면 str 에서 strsub 의 문자열이 나타나는 첫 번째 위치가 됩니다. 이때 str 의 첫 번째 글자 위치를 1 로 합니다.

주의) 유니코드 버전에서 문자의 위치는 문자의 개수와 연관되며 일반버전에서는 바이트 수와 연관됩니다. 유니코드버전인 경우, 한글 하나의 글자가 1 자리를 나타내지만 일반버전에서는 2 자리를 나타냅니다.

예

```
ret = MATCH ("대한민국","대한");
// ret == 1
ret = MATCH (12345, 5);
// ret == 5

str1 = "우리나라 좋은 나라";
str2 = "나라";
ret = MATCH (str1, str2);
// 유니코드버전인 경우, ret == 3
// 일반버전버전인 경우, ret == 5
```

참고

STRSTR()

4.1.41. MAX()

문법

MAX(arglist);

인수

arglist
최대값을 구할 인수 목록
a1,a2,a3 : a1, a2, a3 의 3 개의 변수
a1..a10 : a1 ~ a10 사이의 10 개의 변수

반환값

숫자형
arglist 에 입력된 것 중 가장 큰 값을 반환

설명

인수 목록 중 최대값을 구합니다. 인수에는 상수, 변수 또는 연산식이 올 수 있으며 1 개 이상 올 수 있습니다.

예

```
ret = MAX (0,1,2,3,4,5,6,7,8,9);

ret = MAX (a, b, c);

ret = MAX (a + b, 36/5 - 1, c * d);

ret = MAX (a$, b?);

ret = MAX (a1..a10);
```

참고

MIN()

4.1.42. MESSAGEBOX()

문법

MESSAGEBOX (content, title, kind);

인수

content
메시지박스에 보일 내용을 나타내는 문자열

title
메시지박스의 제목을 나타내는 문자열

kind
메시지박스의 종류
OK : 확인 버튼만 있는 메시지박스
OKCANCEL : 확인, 취소 버튼이 있는 메시지박스

반환값

숫자형
kind 가 OKCANCEL 인 경우, 값을 반환
1 : 확인 버튼을 눌렀을 경우
2 : 취소 버튼을 눌렀을 경우

설명

메시지를 출력합니다.

원하는 내용과 제목의 윈도우 메시지 박스를 표시하는 함수입니다. 사용자에게 어떤 사항을 알리거나 스크립트의 디버깅 목적으로 스크립트 실행 중간에 어떤 값을 보기 위해서 사용됩니다.

예

```
IF (Result == 1)
  String = “TRUE”;
ELSE
  String = “FALSE”;
ENDIF

MESSAGEBOX(String, “결과”, OK);
```

4.1.43. MIN()

문법

MIN(arglist);

인수

arglist
최대값을 구할 인수 목록
a1,a2,a3 : a1, a2, a3 의 3 개의 변수
a1..a10 : a1 ~ a10 사이의 10 개의 변수

반환값

숫자형
arglist 에 입력된 것 중 가장 작은 값을 반환

설명

인수 목록 중 최소값을 구합니다. 인수에는 상수, 변수 또는 연산식이 올 수 있으며 1 개 이상 올 수 있습니다.

예

```
ret = MIN (0,1,2,3,4,5,6,7,8,9);

ret = MIN (a, b, c);

ret = MIN (a + b, 36/5 - 1, c * d);

ret = MIN (a$, b?);

ret = MIN (a1..a10);
```

참고

MAX()

4.1.44. PAGE()

문법

page()

인수

없음

반환값

숫자형
현재 스크립트를 처리하고 있는 페이지번호

설명

생성중인 보고서의 현재 페이지번호를 알려주는 함수입니다.

참고

GOTOPAGE()

4.1.45. PAGEBREAK()

문법

PAGEBREAK()

인수

없음

반환값

없음

설명

현재 페이지를 종료하고 다음페이지를 만듭니다.

drawline(), drawimage() 등의 함수를 이용해서 보고서를 작성할 때 사용합니다.

예

```
pagebreak();
```

참고

GOTOPAGE()

4.1.46. POW()

문법

POW(num, exp)

인수

num
지수값을 계산할 때, 밑(base)을 나타내는 숫자 또는 숫자형 변수

exp
지수값을 계산할 때, 지수(exponent)를 나타내는 숫자 또는 숫자형 변수

반환값

숫자형
반환할 지수값

설명

지수값을 계산합니다. 10 의 제곱과 같은 수치를 계산할 때 사용합니다. num 이 밑(base)이 되고 exp 가 지수(exponent)가 됩니다. num 이 0 이면서 exp 가 음수가 올 수 없습니다.

예

```
ret = POW (3,5);

ret = POW (b/3, a/4);
```

참고

SQRT()

4.1.47. RANDOM()

문법

RANDOM ([seed]);

인수

seed
난수 값을 얻기위한 씨드값. 생략가능합니다.

반환값

숫자형
난수 값

설명

난수 값을 얻는 함수입니다. 인수는 씨드(seed)이며 생략할 수 있습니다.

예

```
NUM seed, ret;

ret = RAMDOM( );

seed = RANDOM ( );

ret  = RANDOM (seed);
```

4.1.48. REPLACESTR()

문법

REPLACESTR (org, str1, str2);

인수

org
원본 문자열

str1
찾을 문자열

str2
변환할 문자열

반환값

문자열
변환한 문자열

설명

org 문자열내에서 str1 문자열에 해당하는 부분을 찾아서 str2 문자열로 변환합니다.

예

```
ret = REPLACESTR ("대한민국 서울","서울","부산");
// ret == "대한민국 부산"

ret = RMSTR ("ABCABCABC", "CA", "DD");
// ret == "ABDDBDDBC"
```

참고

RMSTR()

4.1.49. RMSTR()

문법

RMSTR (str1, str2);

인수

str1
원본 문자열

str2
원본 문자열 내에서 삭제할 문자열

반환값

문자열
str1 문자열에서 str2 문자열을 삭제하고 남은 문자열

설명

str1 내에 str2 와 동일한 문자열이 포함되어 있으면 포함되어 있는 부분을 삭제합니다. 결과 값은 str1 내에서 str2 에 해당하는 문자열을 모두 제거한 문자열이 됩니다.

예

```
ret = RMSTR ("대한민국","대한");
// ret == “민국”

ret = RMSTR ("ABCABCABC","CA");
// ret == “ABBBC”
```

참고

MATCH(), REPLACESTR()

4.1.50. ROUND()

문법

ROUND(num [,pos]);

인수

num
반올림 할 숫자 또는 숫자형 변수

pos
반올림 할 위치를 나타내는 숫자 (-9 ~ 9). 생략가능합니다.
생략 : 소수점 미만 반올림
-1 : 십자리 미만 반올림
0 : 소수점 미만 반올림
1 : 소수 1 자리 미만 반올림

반환값

숫자형
반올림한 결과 값

설명

인수의 값을 자릿수에 따라 반올림합니다. 자릿수는 -9 에서 9 까지 유효하며 생략할 수도 있습니다.

예

```
NUM i, ret;

i = 765.267;

ret = ROUND (i);          // ret 의 값은 765    (소수점 미만 반올림)

ret = ROUND (i, 0);     // ret 의 값은 765     (소수점 미만 반올림)

ret = ROUND (i, 2);     // ret 의 값은 765.27 (소수 2 자리 미만 반올림)

ret = ROUND (i,-1);     // ret 의 값은 770     (십 자리 미만 반올림)
```

참고

TRUNC()

4.1.51. SETAXISAUTO()

문법

SETAXISAUTO(chartname, axisname, opt);

인수

chartname
차트 객체 변수명

axisname
차트의 축이름을 나타내는 문자열. Designer 에서 미리 생성해 놓은 Axis.
보통 “ Left”, “ Right”, “ Top”, “ Bottom” 등의 이름을 가집니다.

opt
최대값, 최소값의 크기를 자동으로 설정할 지의 여부
0 : SETAXISMAX(), SETAXISMIN() 의 함수를 사용하여 최대값, 최소값을 설정
1 : 자동으로 설정

반환값

없음

설명

차트에서 특정 Axis 가 표현하는 최대값, 최소값의 크기를 자동으로 설정할지 여부를 결정합니다.

예

```
int autoopt;

…
…
if (autoopt == 0)
  setaxisauto(chart1, “Left”, 1);
  setaxismax(chart1, “Left”, 100);
  setaxismin(chart1, “Left”, 0);
endif
```

참고

SETAXISMAX(), SETAXISMIN();

4.1.52. SETAXISMAX()

문법

SETAXISMAX(chartname, axisname, value);

인수

chartname
차트 객체 변수명

axisname
차트의 축이름을 나타내는 문자열. Designer 에서 미리 생성해 놓은 Axis.
보통 “ Left”, “ Right”, “ Top”, “ Bottom” 등의 이름을 가집니다.

value
설정할 최대의 크기

반환값

없음

설명

차트에서 특정 Axis 가 표현하는 최대값을 결정합니다.

예

SETAXISAUTO() 함수 참조.

참고

SETAXISAUTO(), SETAXISMIN()

4.1.53. SETAXISMIN()

문법

SETAXISMIN(chartname, axisname, value);

인수

chartname
차트 객체 변수명

axisname
차트의 축이름을 나타내는 문자열. Designer 에서 미리 생성해 놓은 Axis.
보통 “ Left”, “ Right”, “ Top”, “ Bottom” 등의 이름을 가집니다.

value
설정할 최소의 크기

반환값

없음

설명

차트에서 특정 Axis 가 표현하는 최소값을 결정합니다.

예

SETAXISAUTO() 함수 참조.

참고

SETAXISAUTO(), SETAXISMAX()

4.1.54. SETCHARTTITLE()

문법

SETCHARTTITLE(chartname, title);

인수

chartname
차트 객체 변수명

title
차트 객체에서 보여줄 차트의 제목

반환값

없음

설명

차트의 타이틀을 설정합니다.

TITLE: 타이틀로 사용할 문자열입니다.

4.1.55. SETDRAWOBJATTR()

문법

SETDRAWOBJATTR (command, value)

인수

command
그리기 객체의 모양 및 속성을 정의하는 문자열

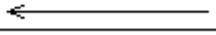
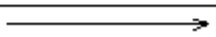
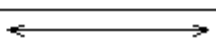
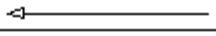
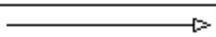
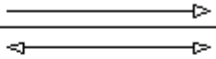
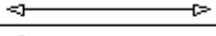
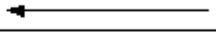
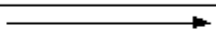
value
command 에 따른 값을 나타내는 문자열

반환값

없음

설명

DRAWLINE() 등으로 그리기 객체를 그릴 때 기본적으로 적용할 속성을 정의합니다. 속성을 나타내는 명령(command)과 해당하는 명령에 대응하는 값(value)은 다음과 같습니다.

command	설명	value	
LS	선모양	1	파선(dash line)
		2	점선(dot line)
		3	일점쇄선(dashdot line)
		4	이점쇄선(dashdotdot line)
		5	html 선(html line)
		그외	실선
LT	선굵기	0 ~ 255 사이의 숫자	
AS	화살표모양	1	
		2	
		3	
		4	
		5	
		6	
		7	
		8	
		9	
		그외	
LC	선색상	R,G,B (각각 0 ~ 255 사이의 숫자)	
BC	brush색상	R,G,B (각각 0 ~ 255 사이의 숫자)	
BS	brush모양	1	hollow
		그외	hatched

예

```
setdrawobjattr("LS@LT@LC", "2@1@255,0,0");
```

참고

DRAWLINE(), DRAWTEXTATTR()

4.1.56. SETDRAWTEXTATTR()

문법

SETDRAWTEXTATTR(command, value);

인수

command
텍스트 상자의 속성을 정의하는 문자열

value
command 에 따른 값을 나타내는 문자열

반환값

없음

설명

drawtext 로 텍스트 상자를 그릴때 기본적으로 적용할 속성을 정의합니다. 속성을 나타내는 명령(command)과 해당하는 명령에 대응하는 값(value)은 다음과 같습니다.

command	설 명	value	
FC	폰트색상	R,G,B(각각 0 ~ 255 사이의 숫자)	
FS	폰트크기	4 ~ 127 사이의 숫자	
FN	폰트명	폰트이름	
FA	폰트속성	B	굵게
		I	기울임
		U	밑줄
		S	가운데선
		DB	굵게 사용안함
		DI	기울임 사용안함
		DU	밑줄 사용안함
		DS	가운데선 사용안함
BC	배경색상	R,G,B(각각 0 ~ 255 사이의 숫자)	
HA	가로정렬	1	왼쪽 정렬
		2	오른쪽 정렬
		3	가운데 정렬
		4	양쪽 정렬
VA	세로정렬	1	상단 정렬
		2	중앙 정렬
		3	하단 정렬

예

setdrawtextattr("FN@FS@FA", "굴림체@20@B");

참고

DRAWTEXT(), SETDRAWOBJATTR()

4.1.57. SETFIXOBJECT()

문법

SETFIXOBJECT (varname, flag);

인수

varname
객체변수명. “ ” 로 묶여 있어야 합니다.

flag
객체가 밀리는 지를 나타내는 플래그
0 : 밀립니다.
1 : 고정되어 밀리지 않습니다.

반환값

없음

설명

AddDataBody() 함수를 사용할 때 기본적으로 반복되는 표의 아래가 자동으로 밀리게 되어 있지만 특정한 객체는 표가 반복되어 늘어나더라도 밀리지 않게 할 때 사용하는 함수 입니다.

varname 객체변수명을 가지고 있는 해당객체에 FLAG 를 설정합니다. 이 FLAG 가 1 이면 AddDAtaBody() 함수가 호출 되었을 때 뒤로 밀리지 않게 되고 0 이면 뒤로 밀리게 됩니다.

예

```
num i, ret;

i = 1;
a = :a1;

setfixobject(“a”, 1);  /* a 변수를 포함하는 객체는 b 가 늘어나도 밀리지 않음 */

cursor = "mycursor";
execfile declare :cursor cursor;
declarefield(cursor, 1, 200);

while(1)
 ret = execfile fetch :cursor into :aa
 if(ret != 1)
   AddDataBody("EndAddDataBody");   /* 현재 늘리고 있는 표 중지*/
   break;
 endif
```

```
  b = aa;
  AddDataBody("b");      /* b 객체가 있는 반복부 늘리기 */
  i = i+1;
endwhile

execfile close :cursor;
```

참고

ADDDATABODY()

4.1.58. SETGLOBAL()

문법

SETGLOBAL(varname, value);

인수

varname
global 변수이름을 나타내는 문자열. 반드시 “ ” 로 묶여있어야 합니다.

value
global 변수에 설정할 데이터 값

반환값

문자열
global 변수의 값

설명

특정 global 변수에 데이터값을 설정하는 함수입니다.

스크립트 보고서에서는 주 데이터의 각 레코드마다 스크립트가 실행됩니다. 실행되는 각각의 스크립트는 서로 독립적인 변수(객체, 지역, 임시변수)를 가지고 있습니다. 실행되는 스크립트의 특정한 결과값을 다음 레코드에 의해 실행되는 스크립트에 반영하고자 하는 경우에는 데이터를 전달하는 방법이 global 변수를 사용하는 것 밖에 없습니다. 데이터를 설정하는 것이 SETGLOBAL() 함수이고 데이터를 얻어오는 것이 GETGLOBAL() 함수입니다.

예

```
char dname, new_dname;

dname = getglobal(“g_dname”);

…
…
if (new_dname != “”)
  setglobal(“g_dname”, new_dname);
  setglobal(“g_ename”, “name data”);
endif
```

참고

GETGLOBAL()

4.1.59. SETHLDATA()

문법

SETHLDATA(varname, target, tooltip, targettype,paramlist, etclist, window);

인수

varname
하이퍼링크속성을 적용할 객체 변수명

target
연결할 대상이름
마우스가 객체위에 위치한 경우, 연결대상은 상태바에 표시됩니다.

tooltip
객체위에 마우스가 위치한 경우에 나타날 풍선도움말

targettype
연결대상 유형을 나타내는 숫자
0 : RD 문서
1 : 웹페이지 문서
2 : 기타문서
3 : 전자메일주소
4 : 현재문서
5 : 스크립트실행. 이 경우, target 이 실행할 스크립트 문장이 됩니다.

paramlist
연결대상유형이 RD 문서인 경우에 사용할 " /rp" 파라미터 리스트. 공백을 포함한 데이터는 " [" 과 "]" 으로 묶어서 넘깁니다.

etclist
연결대상유형이 RD 문서인 경우에 사용할 " /rp" 를 제외한 파라미터 리스트

window
연결대상을 어디에 표시할 지 나타내는 숫자
0 : 현재 창에 표시
1 : 새 창에 표시

반환값

없음

설명

하이퍼링크 정보를 설정합니다.

하이퍼링크로 표현할 텍스트 상자나 표의 셀에 이 함수를 사용해서 하이퍼링크와 관련된 정

보를 설정하도록 합니다.

연결대상(target), 풍선도움말(tooltip), 연결시사용파라미터리스트(paramlist, etclist)는 그 값이 필드명인 경우에 DB 에서 가져온 값으로 매핑되어 최종값이 됩니다.

예

```
SetHLData(hlink,“http://m2soft.co.kr/test/test.mrd”,“툴팁입니다.”,
        0,“1 [죽은 시인의 사회]”, “”, 1);
/* 변수이름 hlink 객체를 마우스 클릭하면 연결대상 문서인
http://m2soft.co.kr/test/test.mrd 에 파라미터 정보
- /rp 1 [죽은 시인의 사회] - 를 적용해서 오픈한다. */

SetHLWinHTML(hlink, "c:\test\default_myocx.html");

SetHLWinOpt(hlink, 0,0,0,0);
/* 새창 옵션인 적용된 하이퍼링크 수행시에 새 창이 메뉴, 툴바, 상태바를 없이
  뜨게 하고 윈도 크기 조정을 할 수 없게 합니다. */

SetHLWinPos(hlink, 20, 20);
SetHLWinSize(hlink, 500, 700);
```

참고

SETHLWINHTML(), SETHLWINOPT(), SETHLWINPOS(), SETHLWINSIZE()

4.1.60. SETHLWINHTML()

문법

SETHLWINHTML(varname, htmlpath);

인수

varname
하이퍼링크 새창옵션을 적용할 객체변수 이름

htmlpath
연결시 사용할 템플릿 HTML 문서 경로명
이 파라미터를 사용 안하면 기본 템플릿을 이용합니다.

반환값

없음

설명

Crownix Report ActiveX Viewer 에서 하이퍼링크 수행시, 브라우저가 이용할 템플릿 HTML 문서이름을 지정합니다.

SETHLDATA() 에서 window(연결대상을 어디에 나타낼지를 표시) 인수 값을 새창에 표시(1)로 지정한 경우에 사용할수 있습니다. ActiveX Viewer 모듈에서만 이용 가능합니다.

예

SETHLDATA() 함수 예제 참조

참고

SETHLDATA(), SETHLWINOPT(), SETHLWINPOS(), SETHLWINSIZE()

4.1.61. SETHLWINOPT()

문법

void SetHLWinOpt (varname, bMenu, bToolbar, bStatus, bChangeSize);

인수

varname
하이퍼링크 새창옵션을 적용할 객체변수 이름

bMenu
메뉴보이기 (0|1) - 숨기기-0 보이기-1
0 : 숨기기
1 : 보이기

bToolbar
툴바보이기 (0|1) - 숨기기-0 보이기-1
0 : 숨기기
1 : 보이기

bStatus
상태바보이기 여부
0 : 숨기기
1 : 보이기

bChangeSize
크기 조정 가능 여부
0 : 크기조정 막음
1 : 크기조정 가능

반환값

없음

설명

하이퍼링크 수행 시 열리는 윈도우의 리소스 표시 여부와 윈도우 크기조절 여부를 지정합니다.

SETHLDATA() 에서 window(연결대상을 어디에 나타낼지를 표시) 인수 값을 새창에 표시(1)로 지정한 경우에 사용할수 있습니다. ActiveX Viewer 모듈에서는 툴바보이기, 상태바보이기만 이용 가능합니다.

예

SETHLDATA() 함수 예제 참조

참고

SETHLDATA(), SETHLWINHTML(), SETHLWINPOS(), SETHLWINSIZE()

4.1.62. SETHLWINPOS()

문법

SETHLWINPOS(varname, left, top);

인수

varname
하이퍼링크 새창옵션을 적용할 객체변수 이름

left
새창의 가로 위치

top
새창의 세로 위치

반환값

없음

설명

하이퍼링크 수행 시 열리는 윈도우의 가로, 세로 위치를 지정합니다.

SETHLDATA() 에서 window(연결대상을 어디에 나타낼지를 표시) 인수 값을 새창에 표시(1)로 지정한 경우에 사용할 수 있습니다. EXE Viewer 모듈에서만 이용 가능합니다.

예

SETHLDATA() 함수 예제 참조

참고

SETHLDATA(), SETHLWINHTML(), SETHLWINOPT(), SETHLWINSIZE()

4.1.63. SETHLWINSIZE()

문법

SETHLWINSIZE(varname, width, height);

인수

varname
하이퍼링크 새창옵션을 적용할 객체변수 이름

width
새창의 가로 크기

height
새창의 세로 크기

반환값

없음

설명

하이퍼링크 수행시 열리는 윈도우의 크기를 지정합니다.

SETHLDATA() 에서 window(연결대상을 어디에 나타낼지를 표시) 인수 값을 새창에 표시(1)로 지정한 경우에 사용할수 있습니다. EXE Viewer 모듈에서만 이용 가능합니다.

예

SETHLDATA() 함수 예제 참조

참고

SETHLDATA(), SETHLWINHTML(), SETHLWINOPT(), SETHLWINPOS()

4.1.64. SETSERIESTITLE()

문법

SETSERIESTITLE(chartname, seriesname, title);

인수

chartname
차트 객체 변수명

seriesname
차트 객체가 가지고 있는 시리즈(Series)명을 나타내는 문자열. “ ” 로 묶여있어야 합니다.
Designer 에서 미리 생성해 놓은 Series. Viewer 가 실행되면서 생성되어 있는 Series 의 순서대로 “ S0”, “ S1”, ” S2” …로 이름이 재부여됩니다. Deleteseries() 함수의 호출로 Series 가 제거되면 Series 의 index 와 위의 이름에 있는 번호가 달라질 수 있습니다.

title
시리즈가 표시되는 이름을 나타내는 문자열

반환값

없음

설명

Series 의 Title 을 바꾸어 줍니다.

예

DEFINESERIES() 함수 예제 참조

참고

SETCHARTTITLE(), DEFINESERIES()

4.1.65. SIN()

문법

SIN(arg);

인수

arg
radian 형태의 숫자 또는 숫자형 변수

반환값

숫자형
arg 의 sine(사인) 값

설명

삼각함수 중에서 sine(사인) 값을 구하는 함수입니다. 일반적인 사용보다는 수학적인 통계나 수치의 연산을 위해서 사용합니다. 인수는 radian 형태로 구성합니다.

예

```
num ret, PI;

PI = 3.141592;

ret = SIN (PI/2);

ret = SIN (PI/6);

ret = SIN (PI/2 + PI/4);

ret = SIN (PI/3 – PI/6);
```

참고

COS(), TAN()

4.1.66. SORTDATA()

문법

SORTDATA(fieldname, order)

인수

fieldname
주쿼리의 필드이름

order
정렬 순서
0 : 내림차순
1 : 오름차순

반환값

숫자형
0 : 실패
1 : 성공

설명

주쿼리의 결과데이터를 특정필드명으로 재정렬하여 보고서를 다시 생성합니다.

스크립트 보고서에서의 스크립트에서는 사용할 수 없으며 하이퍼 객체의 스크립트 기능에서만 사용합니다. 하이퍼링크 객체의 스크립트에서만 사용할 수 있는 함수로는 DRILLDOWN(), DRILLUP(), SORTDATA() 함수 등이 있습니다.

주의) 서브페이지 문서와 스크립트 문서인 경우에는 사용하지 못합니다.

예

```
char name, order, gvar;
num ret;

name = getparam("$2"); /* 파라미터로 정렬할 변수이름을 넘겨받음 */
order = getparam("$3");
gvar = getglobal("nextorder");

//messagebox(gvar, "test", OK);
if (gvar == "오름차순")
  order = 1;
else
  if(gvar == "내림차순")
    order = 0;
```

```
  endif;
endif;

if (order == 1)
  gvar = "내림차순";
else
  gvar = "오름차순";
endif;

ret = sortdata(name, order); /* 넘겨받은 파라미터로 데이터를 정렬함 */
if(ret)
  setglobal("nextorder", gvar);
endif
```

참고

DRILLDOWN(), DRILLUP()

4.1.67. SQRT()

문법

SQRT(num);

인수

num
제곱근을 구할 숫자 또는 숫자형 변수.
num 은 0 보다 크거나 같음.

반환값

숫자형
num 의 제곱근 값

설명

인수의 제곱근(root)를 구하는 함수입니다. x 는 음수가 아니어야 합니다.

예

```
ret = SQRT (2.4);

ret = SQRT (SUM (a, b, c, 4));
```

참고

POW()

4.1.68. STRCAT()

문법

STRCAT(str1, str2);

인수

str1
이어진 문자열에서 앞 부분에 나올 문자열 또는 문자열 변수

str2
이어진 문자열에서 뒷 부분에 나올 문자열 또는 문자열 변수

반환값

숫자형
두 인수의 문자열을 이은 문자열

설명

두 개의 문자열을 이어 하나의 문자열로 만들어 줍니다. ‘ &’ 연산자와 같은 기능을 합니다. 여러 개의 문자열을 동시에 이어주는 경우, strcat() 함수를 사용하는 것보다 ‘ &’연산자를 사용하는 것이 더 편리합니다.

예

```
ret = STRCAT ("abc", b);

ret = STRCAT ("", "Crownix Report");

ret = STRCAT ("엠투소프트", STRCAT(b, STRCAT(c,"Crownix Report")));
```

참고

& 연산자

4.1.69. STRCMP()

문법

STRCMP(str1, str2);

인수

str1
비교할 첫 번째 문자열

str2
비교할 두 번째 문자열

반환값

숫자형
비교 결과값
0 : 두 문자열이 같음
> 0 : str1 이 str2 보다 큼
< 0 : str1 이 str2 보다 작음

설명

두 인수가 동일한지를 사전(dictionary)식으로 비교합니다. 인수는 상수, 변수 또는 연산식이 올 수 있으며 반드시 2 개 이어야 합니다. 결과값은 두 인수가 같으면 0, 인수 1 이 크면 양수, 인수 2 가 크면 음수입니다.

예

```
ret = STRCMP ("ABC","AB"); // ret > 0
ret = STRCMP ("157","75"); // ret < 0

a = "Crownix Report";
b = "Crownix Report";
ret = STRCMP (a, b);              // ret = 0
```

4.1.70. STRLEN()

문법

STRLEN(str);

인수

str
　　길이를 구할 문자열 또는 문자열 변수

반환값

숫자형
　　문자열의 길이

설명

문자열의 길이를 구합니다.

주의) 유니코드 버전에서는 각 문자의 개수로 문자열의 길이를 구하며 일반 버전에는 문자열의 바이트 수로 문자열의 길이를 구합니다. 따라서 같은 문자열이라도 유니코드 버전인지 아닌지에 따라서 길이가 달라질 수 있습니다

예

```
ret = STRLEN ("ABCDEFGHIKL");    // ret = 11
ret = STRLEN (298763);           // ret = 6
ret = STRLEN (a);

strlen("한국")
//유니코드 버전 결과값:  ret == 2
//일반 버전의 결과값:    ret == 4
```

4.1.71. STRRCHR()

문법

STRRCHR(str, character);

인수

str
문자열

character
str 문자열 내에서 찾을 문자

반환값

숫자형
str 문자열에서 끝에서부터 앞으로 character 문자를 검색해서 찾은 위치
0 : 문자 찾지 못함
> 0 : 찾은 위치

설명

str 의 문자열을 끝에서부터 앞쪽으로 검색해 가면서 character 와 동일한 문자를 찾습니다. 결과값은 character 에 해당하는 문자를 찾은 위치가 됩니다. 문자를 찾지 못하면 0 을 반환합니다. character 가 한 글자 이상의 문자열이라 하더라도 첫 번째 문자만 가지고 검색합니다.

주의) 유니코드 버전에서는 각 문자의 개수로 위치 등를 구하며 일반 버전에는 문자열의 바이트 수로 구합니다. 따라서 같은 경우라도 유니코드 버전인지 아닌지에 따라서 반환값이 달라질 수 있습니다

예

```
ret = STRRCHR ("ABCABCABC","A");                    // ret == 7
ret = STRRCHR ("ABCDEF","H");                 // ret == 0

ret = STRRCHR ("대한민국 ABC", "A");
// 유니코드 버전: ret == 5
// 일반버전:     ret == 9
```

참고

STRSTR()

4.1.72. STRSTR()

문법

STRSTR(str1, str2);

인수

str1
문자열

str2
str1 문자열내에서 찾을 문자열

반환값

숫자형
str1 에서 str2 와 동일한 문자열이 나타나는 첫 번째 위치
0 : 문자열 못찾음
> 0 : 문자열 찾은 위치

설명

str1 내에 str2 와 동일한 문자열이 포함되어 있는지 조사합니다. 결과값은 str1 에 str2 가 포함되어 있지 않으면 0, 포함되어 있으면 str1 에서 str2 의 문자열이 나타나는 첫 번째 위치가 됩니다.

주의) 유니코드 버전에서는 각 문자의 개수로 위치 등을 구하며 일반 버전에는 문자열의 바이트 수로 구합니다. 따라서 같은 경우라도 유니코드 버전인지 아닌지에 따라서 반환값이 달라질 수 있습니다.

예

```
ret = STRSTR ("대한민국 서울", "서울");
// 유니코드버전: ret == 6
// 일반버전:    ret == 10

ret = STRSTR ("ABCDEFG","FG");                    // ret == 6
```

참고

MATCH(), STRRCHR()

4.1.73. SUBSTR()

문법

SUBSTR(str, pos, num);

인수

str
일부 문자열을 뽑아낼 원본 문자열 또는 문자열 변수

pos
원본 문자열에서의 위치를 나타내는 숫자
첫 위치는 1 부터 시작

num
뽑아낼 문자열의 길이를 나타내는 숫자

반환값

문자열
str 의 특정부분을 뽑아내 문자열

설명

문자열내에서 특정위치의 문자열을 구합니다. str 문자열의 pos 위치에서부터 num 개수 만큼을 발췌합니다.

주의) 유니코드 버전에서는 문자의 위치 및 길이가 각 문자의 개수와 연관되며 일반 버전에서는 바이트 수와 연관됩니다. 한글 한 글자인 경우, 유니코드 버전에서는 길이가 1 이지만 일반 버전에서는 2 가 됩니다.

예

```
ret = substr("BlueSky",5,3)
//결과값: ret == "Sky"

substr("대한민국서울",5,2)
//유니코드 버전에서의 결과값: ret == "서울"
//일반 버전에서의 결과값:    ret == "민"
```

참고

STRSTR(), STRRCHR()

4.1.74. SUM()

문법

SUM(arglist);

인수

arglist
합계를 구할 인수 목록
a1,a2,a3 : a1, a2, a3 의 3 개의 변수
a1..a10 : a1 ~ a10 사이의 10 개의 변수

반환값

숫자형
인수 목록 변수의 합계 값

설명

인수 목록의 합을 구합니다.

예

```
ret = SUM (1,2,3,4,5,6,7,8,9);

ret = SUM (a, b, c);

ret = SUM (a + b, 36/5 - 1, c * d);

ret = SUM (a$, b?);

ret = SUM (a1..a10);
```

참고

AVG(), MAX(), MIN()

4.1.75. TAN()

문법

TAN(arg)

인수

arg
radian 형태의 숫자 또는 숫자형 변수

설명

삼각함수 중에서 tangent(탄젠트) 값을 구하는 함수입니다.

예

```
num ret, PI;

PI = 3.141592;

ret = TAN (PI/2);

ret = TAN (PI/6);

ret = TAN (PI/a + PI/4);

ret = TAN (PI/3 – PI/b);
```

참고

SIN(), COS()

4.1.76. TEXTREAD()

문법

TEXTREAD(filename, varname, mode);

인수

filename
텍스트 파일의 경로
파일의 크기는 60K 를 넘지 않아야 합니다.

varname
읽어들인 텍스트 데이터를 저장할 변수

mode
파일을 읽기 위한 모드
r : 읽기만을 위해 개설합니다.
t : 텍스트 상태로 파일을 읽습니다.
b : 이진 상태로 파일을 읽습니다.

반환값

없음

설명

텍스트 파일을 지정 변수로 읽어옵니다.

TEXTREAD ()는 반드시 글의 입력이 가능한 객체인 텍스트 상자와 표에서만 읽어오기가 가능하고 읽어올 파일은 60K 이내의 텍스트 파일이어야 합니다. 객체 내의 텍스트 문서 내용을 파일로 저장하기 위하여는 TEXTWRITE () 을 이용합니다.

예

```
char    var;

TextRead ("file1.txt", var, "rt");
/* 텍스트파일 file1.txt 를 텍스트 상태로 읽어 와서 문자형 변수 var 에 저장합니다.
즉, 아래 그림과 같이 var 텍스트 상자 객체에 file1.txt 파일을 읽어옵니다. */
```

참고

TEXTWRITE( )

4.1.77. TEXTWRITE ()

문법

TEXTWRITE (filename, varname, mode);

인수

filename
텍스트 파일의 경로
파일의 크기는 60K 를 넘지 않아야 합니다.

varname
파일에 저장할 텍스트 데이터를 가지고 있는 변수

mode
파일을 읽기 위한 모드
w, w+ : 쓰기를 위해 개설합니다.
a, a+ : 파일의 끝에 덧붙여 쓰기 위해 개설하거나 파일이 없는 경우, 쓰기를 위해 개설합니다.
t : 텍스트 상태로 파일을 씁니다.
b : 이진 상태로 파일을 씁니다.

반환값

숫자형

설명

객체 내의 텍스트 문서 내용을 파일에 씁니다.

TEXTWRITE ()는 지정 객체 내의 문서를 텍스트 문서로서 저장합니다. 그 문서를 다시 읽어 오려면 스크립트 내장 함수 TextRead () 를 이용합니다.

예

```
TEXTWRITE("file2.txt", var, "w+t");
/* 텍스트 상자(or 표) 객체 변수 var 의 내용을 "file2.txt"
이름의 텍스트 파일로서 저장합니다. */
```

참고

TEXTREAD()

4.1.78. TIME()

문법

TIME(timevalue, kind);

인수

timevalue
시간 값

kind
반환할 값의 종류
HOUR : 시(24 시간제)
MINUTE : 분
SECOND : 초

반환값

숫자형
kind 에 따라서 반환하는 값의 종류가 변경됨

설명

시간 값에서 원하는 단위 값(시, 분 또는 초)을 추출하는 함수입니다.

예

```
현재시 = time(currenttime(), HOUR);
// 현재 시각이 12 시 35 분 24 초이면 현재 시는 12 가 됩니다.
```

참고

CURRENTDATE(), CURRENTTIME(), DATE()

4.1.79. TOLOWER()

문법

TOLOWER(str);

인수

str
문자열

반환값

문자열
str 내의 알파벳 대문자를 모두 소문자로 변환한 문자열

설명

str 인수내의 알파벳 대문자를 모두 소문자로 변환합니다.

예

```
CHAR str;        // 문자형 지역 변수 str 을 선언

str="ABCEDRG";

OBJ=TOLOWER (str);
```

참고

TOUPPER()

4.1.80. TOUPPER()

문법

TOUPPER(str);

인수

str
문자열

반환값

문자열
str 내의 알파벳 소문자를 모두 대문자로 변환한 문자열

설명

str 인수내의 알파벳 소문자를 모두 대문자로 변환합니다.

예

```
CHAR str;          // 문자형 지역 변수 str 를 선언

str="abcdefg";

OBJ=TOUPPER (str);
```

참고

TOLOWER()

4.1.81. TRUNC()

문법

TRUNC(num [,pos]);

인수

num
버림을 할 숫자 또는 숫자형 변수

pos
버림을 할 위치를 나타내는 숫자 (-9 ~ 9). 생략 가능합니다.
-1 : 십자리 미만 버림
0 : 소수점 미만 버림
1 : 소수 1 자리 미만 버림

반환값

숫자형
num 숫자를 특정위치에서 버림하고 난 결과값

설명

num 인수에 해당하는 값을 자리 수에 따라 버림합니다. 자리 수는 -9 에서 9 까지 유효합니다.

예

```
NUM i, ret;

i = 765.267;

ret = TRUNC (i);        // ret 의 값은 765     (소수점 미만 버림)

ret = TRUNC (i, 0);     // ret 의 값은 765     (소수점 미만 버림)

ret = TRUNC (i, 2);     // ret 의 값은 765.26 (소수 2 자리 미만 버림)

ret = TRUNC (i,-1);     // ret 의 값은 760     (십 자리 미만 버림)
```

참고

ROUND()

4.1.82. UPDATEEOR()

문법

UPDATEEOR();

인수

없음

반환값

없음

설명

스크립트 문서가 아닌 문서(일반문서, 표문서, 라벨문서)에서 스크립트를 사용하고자 할 때 사용합니다. 서브페이지 필드정의 데이터를 저장하는 함수인 UPDATESUBVALUES() 를 한 이후에 서브페이지 데이터간의 구분을 위해서 서브페이지 데이터의 종료를 표시해야 합니다. 이 때, 사용하는 함수가 UPDATEEOR() 함수입니다.

updatevalues( ) 함수의 설명 참조

예

updatevalues( ) 함수의 예제 참조

참고

UPDATEVALUES(), UPDATESUBVALUES()

4.1.83. UPDATESUBVALUES()

문법

UPDATESUBVALUES(subpageno);

인수

subpageno
서브페이지 번호

반환값

없음

설명

스크립트 문서가 아닌 문서(일반문서, 표문서, 라벨문서)에서 스크립트를 사용하고자 할 때 사용합니다. 이 함수는 서브페이지 필드정의 데이터를 저장하는 함수이고 기본 필드정의의 데이터를 저장하는 UPDATEVALUES() 함수와 같이 사용될 수 있습니다.

updatevalues( )의 설명 참조

예

updatevalues( )의 예제 참조

참고

UPDATEVALUES()

4.1.84. UPDATEVALUES()

문법

UPDATEVALUES();

인수

없음

반환값

없음

설명

스크립트 문서가 아닌 문서(일반문서, 표문서, 라벨문서)에서 스크립트를 사용하고자 할 때 사용합니다.

스크립트 문서가 아닌 문서에서 하나의 쿼리만으로 데이터베이스에서 얻어올 자료에 대한 정의가 어려운 경우가 생길 수 있습니다. 이 때에는 동시에 스크립트를 사용하여 문서에서 필요한 자료를 쉽게 만들어 줄 수가 있습니다. 그 이유는 스크립트를 사용하면 여러 개의 쿼리 문장이 사용 가능하고 또한, 이런 자료에 대한 변형을 쉽게 가할 수 있기 때문입니다.

일반문서, 표문서, 라벨문서는 스크립트를 사용하는 스크립트 문서와는 달리, 특정한 하나의 레코드 형태만을 필요로 하기 때문에 **파일접속**에서 처럼 레코드의 **필드 정의**가 필요합니다.

스크립트가 실행되는 도중에 UPDATEVALUES(), UPDATESUBVALUES()가 실행되면 필드 정의에서 정의한 필드이름에 해당하는 값을 가지고 하나의 레코드를 만들게 됩니다. 이 함수를 사용하는 문서는 레코드를 모두 생성하고 나서 즉, 스크립트가 실행 종료한 시점에서 폼이 만들어지기 때문에 다소 느리다는 단점이 있습니다.

예

```
num ret;

cursor = “mycursor”;
query = “select korean, english, math from courses”;

/* 학생”은 데이터베이스-필드정의메뉴에서
   기본 필드정의에 정의된 필드이름 입니다. */
학생 = “홍길동”;
updatevalues();
execsql declare :cursor cursor for :query;

/* while 내의 statement 들이 한번 실행될 때마다 하나의 레코드생성*/
while(1)
```

```
  /* 국어, 영어, 수학 이 데이터베이스-필드정의메뉴에서
     서브페이지에 정의된 필드이름 입니다. */
  ret = execsql fetch :cursor into :국어, :영어, :수학;
  if (ret_m != 1)
    break;
  endif

  /* 국어, 영어, 수학의 변수값을 가지고 하나의 레코드가 만들어집니다. */
  updatesubvalues(1);
endwhile

updateeor();

execsql close :cursor;
```

참고

UPDATEVALUES(), UPDATESUBVALUES()

4.1.85. RESIZEOBJECT()

문법

RESIZEOBJECT(varname, width, height);

인수

varname
크기를 변경할 객체의 객체변수명

width
객체의 폭을 나타내는 숫자

height
객체의 높이를 나타내는 숫자

반환값

없음

설명

varname 에 해당하는 변수명을 가지고 있는 객체의 크기를 지정한 width 와 height 대로 변경합니다.