Chandra X-ray Center

AHELP for CIAO 3.4   **array_map**   Context: slangrtl

Jump to: Description Example See Also

Synopsis

Apply a function to each element of an array

Syntax

```
Array_Type array_map (type, func, arg0, ...)
```

Description

```
    DataType_Type type;
    Ref_Type func;
```

The array_map function may be used to apply a function to each element of an array and returns the result as an array of a specified type. The type parameter indicates what kind of array should be returned and generally corresponds to the return type of the function. The arg0 parameter should be an array and is used to determine the dimensions of the resulting array. If any subsequent arguments correspond to an array of the same size, then those array elements will be passed in parallel with the first arrays arguments.

Example

The first example illustrates how to apply the strlen function to an array of strings:

```
    S = ["", "Train", "Subway", "Car"];
    L = array_map (Integer_Type, &strlen, S);
```

This is equivalent to:

```
    S = ["", "Train", "Subway", "Car"];
    L = Integer_Type [length (S)];
    for (i = 0; i < length (S); i++) L[i] = strlen (S[i]);
```

Now consider an example involving the strcat function:

```
    files = ["slang", "slstring", "slarray"];

    exts = ".c";
    cfiles = array_map (String_Type, &strcat, files, exts);
    % ==> cfiles = ["slang.c slstring.c slarray.c"];

    exts =  [".a",".b",".c"];
```

```
    xfiles = array_map (String_Type, &strcat, files, exts);
    % ==> xfiles = ["slang.a", "slstring.b", "slarray.c"];
```

Many mathemetical functions already work transparantly on arrays. For example, the following two statements produce identical results:

```
    B = sin (A);
    B = array_map (Double_Type, &sin, A);
```

See Also

slangrtl

isnull, reshape, typeof, array_info, bstrlen, create_delimited_string, get_struct_field, getenv, init_char_array, length, reshape, set_struct_field, sin, strcat, strjoin, strlen, strncmp, strsub, substr, transpose, typeof, where

---