Subject: Re: Pointer Help - Referencing/Dereferencing in Functions & Procedures
Posted by Chris[1] on Wed, 12 Mar 2003 20:50:55 GMT
View Forum Message <> Reply to Message

You need to pass an argument into your procedure; as far as it knows
"number_ptr" hasn't been declared.

Change the first line of number_proc to:

pro number_proc, number_ptr

and the call in $MAIN$ to

number_proc,number_ptr

and it should work.

Chris

Subject: Re: Pointer Help - Referencing/Dereferencing in Functions & Procedures
Posted by JD Smith on Wed, 12 Mar 2003 23:57:32 GMT
View Forum Message <> Reply to Message

On Wed, 12 Mar 2003 13:50:55 -0700, Chris wrote:

> You need to pass an argument into your procedure; as far as it knows
> "number_ptr" hasn't been declared.
>
> Change the first line of number_proc to:
>
> pro number_proc, number_ptr
>
> and the call in $MAIN$ to
>
> number_proc,number_ptr
>
>
>
> and it should work.
>
>
> Chris

Which is to say that, even though the heap of data to which a pointer points is available globally, the pointer itself is not. In fact, when you lose the pointer, but the heap data remains, this is a memory leak:

```
IDL> a=ptr_new(fltarr(1000))
IDL> a=1 ; uh oh, where's the pointer?
IDL> help,/heap
Heap Variables:
   # Pointer: 1
   # Object : 0

<PtrHeapVar1>   FLOAT     = Array[1000]
```

Here you see data on the global "pointer heap", but since you overwrote the pointer referring to it with "1", it's lost. It's still on the heap, but you just can't get to it (unless you know some arcane tricks). You can clean it up with:

```
IDL> heap_gc,/verbose
<PtrHeapVar1>   FLOAT     = Array[1000]
```

That got rid of it. So, in order to use the data a pointer points to, you need to pass the pointer in as an argument, or perhaps save it in a common block so you can get to it from anywhere.

JD
