

Re: Passing an array of structures from a pointer?

Source: <http://www.tech-archive.net/Archive/VC/microsoft.public.vc.language/2008-09/msg00899.html>

- *From:* Robby <Robby@xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx>
 - *Date:* Mon, 29 Sep 2008 10:27:01 -0700
-

Hello Barry,

I really must thank you for your much valued feedback. Now I see the reason why we should only declare stuff in headers and only define stuff in .c files. Thank you for pointing it out with great scrutiny.

Too often I have been so caught up in just getting the project to work at all costs and without debugging errors which rendered me sort of absent minded about specific details that contribute in doing things the right way! And for this I apologize to you and the newsgroups.

It makes sense, to only declare structs in headers and then define the arrays, variables or pointers of that struct within functions that require the use of that structure! In my example, the struct should be declared in the header and the array of that struct should be defined in the function. Also, this seems to suit me fine since my MCU will use its memory more efficiently as opposed to leaving all my structures global.

However, what if you have a simple array like this:

```
int xxx[5] = { 1, 3, 5, 7, 9};
```

In the header we would declare?

```
int xxx[5];
```

and in the .c file we define?

```
int xxx[0] = 1;  
int xxx[0] = 3;  
int xxx[0] = 5;  
int xxx[0] = 7;  
int xxx[0] = 9;
```

What is the right way? Its nice to just declare and define it in the header all in one line, but if you say its a nono! then is the above correct?

I re-wrote my example code the best I could so that it compiles.

However, for now there is too much to change to my code if I would define

Re: Passing an array of structures from a pointer?

the array of structures in a functions as opposed to defining it globally. Just to mention this, I did try to re-code my file using your recommendation of defining the array of struct in a function of .c file, but another function gave me an undefined identifier error in one of my other functions more precicely at a for loop:

```
for(i=0; i<DDLB_ITEM_LIST1; i++)
{ ... other code... }
```

where DDLB_ITEM_LIST1 is a global macro:

```
#define DDLB_ITEM_LIST1 (long) ((sizeof (DDLB_COLL1)/sizeof (struct
ddlbColl1)))
```

It will take me some time before I can modify my file to fully implement the declations and defines appropriately.

Any how the following compiles without errors!

```
=====Main Header
struct ddbColl1{
long LINK_ID;
int LIST_NUMBER;
} DDLB_COLL1[] = // Eventually Will be defined in a function in .c
file.
{
200, 4,
201, 8};

typedef struct ddListBox{
int x;
long y;
long *xxx;
} DDLB; // Eventually Will be defined in a function in .c file.
DDLB *obj_DDLB; // Eventually Will be defined in a function in .c file.

DDLB* ULC_DDLB_config_ddlb (struct ddbColl1 DDLB_COLL1[], int x, int y);
void ULC_DDLB_ddlb ( DDLB *obj_DDLB);
=====

=====Main

int main()
{

obj_DDLB = ULC_DDLB_config_ddlb (e_CREATE, e_I500, DDLB_COLL1,
pMN_DC->CURR_MENU_ITEM);
ULC_DDLB_ddlb (obj_DDLB);

return 0;
```

Re: Passing an array of structures from a pointer?

```
}  
  
DDLB* ULC_DDLB_config_ddlb  
( struct ddlbColl1 DDLB_COLL1[], int x, int y)  
{  
  obj_DDLB = NULL;  
  obj_DDLB = (DDLB*) malloc (sizeof (struct ddListBox));  
  
  obj_DDLB->x = x;  
  obj_DDLB->y = y;  
  obj_DDLB->xxx = DDLB_COLL1;  
  
  return obj_DDLB;  
}  
  
void ULC_DDLB_ddlb ( DDLB *obj_DDLB)  
{  
  // implementation file which uses the infos from DDLB_COLL1 array!  
  
}
```

=====

the "ddlbColl1" structure will have an array of itself declared as "DDLB_COLL1". In my example below I only declared one (1) such structure called "ddlbColl1", but someday I could end up having 2, 3 or even 5 of them!

I'm sorry but this makes no sense. A structure may not contain itself. The significance of the case change is not apparent.

I appologize Barry!

"I will declare a struct called "ddlbColl1" and then define an array of "ddlbColl1" structures...."

Do not define objects in your header. However, do declare obj_DDLB as an external pointer with
extern DDLB *obj_DDLB;
so that when you define it in a source file other source files will have access to it. (Ignoring the issue of whether a global pointer is

Re: Passing an array of structures from a pointer?

a good idea or not.)

I will have to work on this!

```
{
  200, 4,
  201, 8};

DDLB* ULC_DDLB_config_ddlb
```

Surely this function definition is not in a header but in a separate source file.

Yes it is, I just put it there so I can show my sample code in one page!

Since you don't need obj_DDLB to be global, you should define it here. You will end up with three objects with that name, one each in main, ULC_DDLB_config_ddlb, and ULC_DDLB_ddlb. Since they all have different scope, there is no problem, However, to avoid confusion, you might want to give the object in each function a slightly different name

For now, I prefer to just keep the obj_DDLB object global. When I re-structure the code according to your recommendations, I will define it as you say. However when I do this I will most probably use a new post since I will undoubtedly have problems!

The above code has been adapted to the full blown project and now compiles without errors. If I have problems accessing the informations in the DDLB_COLL1[] array while in the "void ULC_DDLB_ddlb (DDLB *obj_DDLB)" function, I will certainly get back to the newsgroups, under a new post though!

Barry, I thank you for your sincere help!

—

Best regards
Roberto

—

Re: Passing an array of structures from a pointer?

Best regards
Robert

"Barry Schwarz" wrote:

On Sun, 28 Sep 2008 15:51:00 -0700, Robby
<Robby@xxxxxxxxxxxxxxxxxxxxxxxxxxxx> wrote:

Hello,

I have a two part question. I do understand the principle of an array like so:

```
int xxx[10];
```

where xxx holds the address of the first item. But, when using this analogy

First problem: xxx does not hold the address of the first item. In many (but not all) contexts, the expression xxx is ***converted*** to the address of the first item, equivalent to the expression &xxx[0].

with arrays of structures, I feel a little unsure! Anyways, here is my problem!

-[Q#1]-

I am trying to pass the address of an array of structures, that have already been declared in one of my headers. I don't know... but I feel like I am

It is extremely undesirable that an array of struct be defined in a header. Headers contain typedef definitions, external declarations, function prototypes, and macro definitions but hardly ever object definitions.

drawing a blank ! I know I did similar stuff like this but not from the same exact perspective and the fact that I need to pass the address of an array of structures somehow confuses me! Basically I need to pass the address of the array of structures to a function that will then store this address in another structure for later use.

There is no difference in passing an array of struct to a function than in passing any other array (do you use strcpy or sprintf or sscanf). Arrays are passed by value and the actual value used is the address of the first element of the array.

Re: Passing an array of structures from a pointer?

If you have a declaration of the form
`struct x{...};`
and an array definition of the form
`struct x y[N];`
and a function that expects such an array (that is the function prototype specifies either
`void func(struct x [...]);`
or
`void func(struct x *...);`
which are equivalent as noted in my first comment and where the return type and other parameters are not part of this discussion), then you pass the array of struct to the function with
`func(y ...);`

Here's what I am trying to accomplish! In a header file I will be declaring several structures that have several members. Furthermore, an array for every one of these structures will be declared. For example, as shown below,

Don't define arrays in headers. How will you avoid the duplicate definition problem if your code has more than once source file? Arrays should be defined in the primary function that uses them and passed to any secondary functions as arguments. If you absolutely must have global arrays, declare them as external in the header and define them in the same source file as main.

the "ddlColl1" structure will have an array of itself declared as "DDL_COLL1". In my example below I only declared one (1) such structure called "ddlColl1", but someday I could end up having 2, 3 or even 5 of them!

I'm sorry but this makes no sense. A structure may not contain itself. The significance of the case change is not apparent.

The thing is, that in my main calling function I want to configure all the necessary components for my drop down list box by calling the `ULC_DDLB_config_ddlb ()` function which will setup informations in the "ddListBox" structure . One of the parameters of this function will be the desired array of structures collections such as the `DDLB_ARY[] !`

Then once back in main I would call the:

```
ULC_DDLB_ddlb (obj_DDLB);
```

Re: Passing an array of structures from a pointer?

function and pass the pointer to a ddListBox structure and manipulate its data, such as the data from the DDLB_COLL1 array within the function's implementation code.

–[Q#2]–

When I am in the void ULC_DDLB_ddlb () function, will I be able to reference data from the DDLB_COLL1 array? I believe that to retrieve or modify the data in the DDLB_COLL1 array I can do it like this: right?

```
DDLB_COLL1[1]. LIST_NUMBER;
```

I could be wrong, this is the first time I am passing arrays around this way!!!!

Here's the code sample, I really am unsure of the lines flagged with "[***)" !!!

All help is appreciated!

=====

```
typedef struct ddListBox{
int x;
long y;
struct ddlbColl1 DDLB_ARY[]; //Store address of DDLB_COLL1 here !
[***)
```

If you want to store an address here, use a pointer, not an array.

```
} DDLB;
DDLB *obj_DDLB;
```

Do not define objects in your header. However, do declare obj_DDLB as an external pointer with
extern DDLB *obj_DDLB;
so that when you define it in a source file other source files will have access to it. (Ignoring the issue of whether a global pointer is a good idea or not.)

```
struct ddlbColl1 {
```

This struct must be declared prior to struct ddListBox.

Re: Passing an array of structures from a pointer?

```
long LINK_ID;  
int LIST_NUMBER;  
} DDLB_COLL1[] =
```

Do not define any objects of type struct ddbColl1 in this header. Save the definitions for a source (.c or .cpp) file. Since you only use this object in main, you should define it there.

```
{  
200, 4,  
201, 8};
```

```
DDLB* ULC_DDLB_config_ddlb
```

Surely this function definition is not in a header but in a separate source file.

```
( struct ddbColl1 DDLB_ARY[], int x, int y)  
{  
obj_DDLB = NULL;
```

If you put the previous definition of obj_DDLB at file scope in this source file, it will be global.

```
obj_DDLB = (DDLB*) malloc (sizeof (struct ddListBox));
```

It would be a good idea to loose the cast since it is unnecessary in C (in C++ you would use new) and has undesirable side effects.

```
obj_DDLB->x = x;  
obj_DDLB->y = y;  
obj_DDLB->DDLB_ARRAY = DDLB_ARY; //Store address of  
DDLB_COLL1 ! [***]
```

obj_DDLB is a pointer to struct ddListBox. There is no member DDLB_ARRAY in that struct. Did you mean DDLB_ARY? Assuming yes, this is a syntax error. An array cannot be the left operand of the assignment operator. If you change the member to a pointer as suggested, then this will work.

It works because the right hand operand (the parameter DDLB_ARY – it

Re: Passing an array of structures from a pointer?

would have been nice if you had used a different name) is actually a pointer (even though it looks like an array). You can obviously assign a pointer value to a pointer.

```
return obj_DDLB;
```

Since you return this value, it seems to me that you really don't need obj_DDLB to be global. In that case, you should delete its declaration from your header and define it local to this function.

```
}

void ULC_DDLB_ddlb ( DDLB *obj_DDLB)
{
// This is the implementation file where I would like to use the
// informations from
// the DDLB_COLL1 array!
```

Go right ahead.

1 – The declaration of obj_DDLB as a parameter to this function will completely hide the global object with the same name. Further proof that you don't need the global object.

2 – You can refer to the elements of the array using normal subscript notation, obj_DDLB[0] is the first element. The fact that each element is a struct doesn't matter until you start to access the members of the struct. Then you have operands of the form obj_DDLB[3].x

```
}

int main()
{
```

Since you don't need obj_DDLB to be global, you should define it here. You will end up with three objects with that name, one each in main, ULC_DDLB_config_ddlb, and ULC_DDLB_ddlb. Since they all have different scope, there is no problem, However, to avoid confusion, you might want to give the object in each function a slightly different name.

Re: Passing an array of structures from a pointer?

```
obj_DDLB = ULC_DDLB_config_ddlb ( DDLB_COLL1, 10, 5);  
ULC_DDLB_ddlb (obj_DDLB);
```

```
return 0;  
}
```

=====

If this is too complicated I can simply find an easier more inefficient way,

It isn't complicated at all. Your only problem was trying to declare the struct member as an array instead of a pointer possibly compounded by defining objects in your header.

but I really find this would be a good fit for my project since all I have to do, is set my drop down list box once with the ULC_DDLB_config_ddlb() function and then just fetch the ULC_DDLB_ddlb(obj_DDLB) function and access all the info's via the obj_DDLB pointer using the points to operator "->".

I really would appreciate any help on how I can store the address of an array of structures in the respective places in the above example flagged by [***] !

Sincerely, thanking all in advance!

PS. Please note, I haven't shown the code for any testing of pointers returned by malloc or any free()'s in order to keep the code short and simple!

--
Remove del for email