Passing more parameters in C function pointers

Asked 16 years, 4 months ago Modified 8 years ago Viewed 16k times

9

Let's say I'm creating a chess program. I have a function

```
void foreachMove( void (*action)(chess_move*), chess_game* game);
```

which will call the function pointer action on each valid move. This is all well and good, but what if I need to pass more parameters to the action function? For example:

```
chess_move getNextMove(chess_game* game, int depth){
  //for each valid move, determine how good the move is
  foreachMove(moveHandler, game);
}

void moveHandler(chess_move* move){
  //uh oh, now I need the variables "game" and "depth" from the above function
}
```

Redefining the function pointer is not the optimal solution. The foreachMove function is versatile and many different places in the code reference it. It doesn't make sense for each one of those references to have to update their function to include parameters that they don't need.

How can I pass extra parameters to a function that I'm calling through a pointer?

c architecture pointers

Share
Improve this question
Follow

edited Sep 9, 2008 at 21:13
Community Bot
1 ● 1

asked Aug 14, 2008 at 16:51
andrewrk
31.1k ● 28 ● 95 ● 117

9 Answers

Sorted by: Highest score (default)

13

Ah, if only C supported closures...

Antonio is right; if you need to pass extra parameters, you'll need to redefine your function pointer to accept the additional arguments. If you don't know exactly what parameters you'll need, then you have at least three choices:

1. Have the last argument in your prototype be a void*. This gives you flexibility of passing in anything else that you need, but it definitely isn't type-safe.
2. Use variadic parameters (...). Given my lack of experience with variadic parameters in C, I'm not sure if you can use this with a function pointer, but this gives even more flexibility than the first solution, albeit still with the lack of type safety.
3. Upgrade to C++ and use function objects.

Share Improve this answer Follow

answered Aug 14, 2008 at 17:19
user316

7

You'd probably need to redefine the function pointer to take additional arguments.

```
void foreachMove( void (*action)(chess_move*, int), chess_game* game )
```

Share Improve this answer Follow

answered Aug 14, 2008 at 16:58
Antonio Haley
4,790 ● 1 ● 28 ● 19

4

If you're willing to use some C++, you can use a "function object":

```
struct MoveHandler {
    chess_game *game;
    int depth;

    MoveHandler(chess_game *g, int d): game(g), depth(d) {}

    void operator () (chess_move*) {
         // now you can use the game and the depth
    }
};
```

and turn your `foreachMove` into a template:

```
template <typename T>
void foreachMove(T action, chess_game* game);
```

and you can call it like this:

```
chess_move getNextMove(chess_game* game, int depth){
    //for each valid move, determine how good the move is
    foreachMove(MoveHandler(game, depth), game);
}
```

but it won't disrupt your other uses of `MoveHandler`.

Share Improve this answer Follow

answered Aug 14, 2008 at 17:32

Jesse Beder
34k ● 22 ● 110 ● 146

2

If I'm reading this right, what I'd suggest is to make your function take a pointer to a struct as an argument. Then, your struct can have "game" and "depth" when it needs them, and just leave them set to 0 or Null when you don't need them.

What is going on in that function? Do you have a conditional that says,

```
if (depth > -1) //some default
  {
  //do something
  }
```

Does the function always REQUIRE "game" and "depth"? Then, they should always be arguments, and that can go into your prototypes.

Are you indicating that the function only sometimes requires "game" and "depth"? Well, maybe make two functions and use each one when you need to.

But, having a structure as the argument is probably the easiest thing.

Share

Improve this answer

Follow

edited Nov 24, 2016 at 22:22

Adam Tyszecki
49 ● 1 ● 5

answered Aug 14, 2008 at 17:24

Baltimark
9,272 ● 12 ● 38 ● 35

1

I'd suggest using an array of void*, with the last entry always void. say you need 3 parameters you could do this:

```
void MoveHandler (void** DataArray)
{
    // data1 is always chess_move
    chess_move data1 = DataArray[0]? (*(chess_move*)DataArray[0]) : NULL;
    // data2 is always float
    float data1 = DataArray[1]? (*(float*)DataArray[1]) : NULL;
    // data3 is always char
    char data1 = DataArray[2]? (*(char*)DataArray[2]) : NULL;
    //etc
}
```

```
void foreachMove( void (*action)(void**), chess_game* game);
```

and then

```
chess_move getNextMove(chess_game* game, int depth){
    //for each valid move, determine how good the move is
    void* data[4];
    data[0] = &chess_move;
    float f1;
    char c1;
    data[1] = &f1;
    data[2] = &c1;
    data[3] = NULL;
    foreachMove(moveHandler, game);
}
```

If all the parameters are the same type then you can avoid the void* array and just send a NULL-terminated array of whatever type you need.

Share Improve this answer Follow

answered Aug 14, 2008 at 18:15

Nathan Fellman
127k ● 104 ● 264 ● 326

0

+1 to Antonio. You need to change your function pointer declaration to accept additional parameters.

Also, please don't start passing around void pointers or (especially) arrays of void pointers. That's just asking for trouble. If you start passing void pointers, you're going to also have to pass some kind of message to indicate what the pointer type is (or types are). This technique is *rarely* appropriate.

If your parameters are always the same, just add them to your function pointer arguments (or possibly pack them into a struct and use that as the argument if there are a lot of parameters). If your parameters change, then consider using multiple function pointers for the multiple call scenarios instead of passing void pointers.

Share Improve this answer Follow

answered Aug 14, 2008 at 18:39

Derek Park
46.8k ● 16 ● 59 ● 76

0

If your parameters change, I would change the function pointer declaration to use the "..." technique to set up a variable number of arguments. It could save you in readability and also having to make a change for each parameter you want to pass to the function. It is definately a lot safer than passing void around.

http://publications.gbdirect.co.uk/c_book/chapter9/stdarg.html

Just an FYI, about the example code in the link: some places they have “n args” and others it is “n_args” with the underscore. They should all have the underscore. I thought the syntax looked a little funny until I realized they had dropped the underscore in some places.

Share Improve this answer Follow

answered Aug 14, 2008 at 20:22
Dale Ragan
18.3k ● 3 ● 55 ● 71

0

Use a typedef for the function pointer. See my answer for this question

Share
Improve this answer
Follow

edited May 23, 2017 at 11:53
Community Bot
1 ● 1

answered Sep 16, 2008 at 14:57
roo
7,196 ● 9 ● 42 ● 45

0

Another option would be to modify the `chess_move` structure instead of the function prototype. The structure is presumably defined in only one place already. Add the members to the structure, and fill the structure with appropriate data before any call which uses it.

Share Improve this answer Follow

answered Apr 8, 2013 at 5:08
luser droog
19.5k ● 3 ● 56 ● 106