<gbdirect>

Search Go

Training | Contact

Publications
The C Book
Preface
Introduction
Variables & arithmetic
Control flow
Functions
Arrays & pointers
Opening shots
Arrays
Pointers
Character handling
Sizeof & malloc
Function pointers
Pointer expressions
Arrays & address-of
Summary
Exercises
Structures
Preprocessor
Specialized areas
Libraries
Complete Programs
Answers
Copyright

Contact Us

Publications > The C Book > Arrays & pointers > Function pointers

5.6. Pointers to functions

A useful technique is the ability to have pointers to functions. Their declaration is easy: write the declaration as it would be for the function, say

```
int func(int a, float b);
```

and simply put brackets around the name and a * in front of it: that declares the pointer. Because of precedence, if you don't parenthesize the name, you declare a function returning a pointer:

```
/* function returning pointer to int */
int *func(int a, float b);

/* pointer to function returning int */
int (*func)(int a, float b);
```

Once you've got the pointer, you can assign the address of the right sort of function just by using its name: like an array, a function name is turned into an address when it's used in an expression. You can call the function using one of two forms:

```
(*func)(1,2);
/* or */
func(1,2);
```

The second form has been newly blessed by the Standard. Here's a simple example.

```
#include <stdio.h>
#include <stdlib.h>

void func(int);

main(){
        void (*fp)(int);

        fp = func;

        (*fp)(1);
        fp(2);
```

Printer-friendly version

```
        exit(EXIT_SUCCESS);
}

void
func(int arg){
        printf("%d\n", arg);
}
```

Example 5.16

If you like writing finite state machines, you might like to know that you can have an array of pointers to functions, with declaration and use like this:

```
void (*fparr[])(int, float) = {
                              /* initializers */
                      };
/* then call one */

fparr[5](1, 3.4);
```

About Us | Training | Consultancy | Software | Publications | Open Source | Support | Open Standards | FAQ | Jobs

Example 5.17

But we'll draw a veil over it at this point!

Previous section | Chapter contents | Next section