C pointers practice

Table of Contents

README

- Practice workbook for pointers in C
- See GitHub for script/solutions

Emacs setup (optional)

Hide emphatic characters like ~, *

To **not** see the emphatic characters like ~ or * or / in the Org file text, run the following code chunk (or put the code in your `/.emacs` file): if successful, you should see `"t"` in the minibuffer.

```
Emacs Lisp
(setq-default org-hide-emphasis-markers t)
```

This will only work for new buffers. If you don't put it in your `/.emacs` file, the command will only work for the current Emacs session.

Close and reopen this file to see an effect.

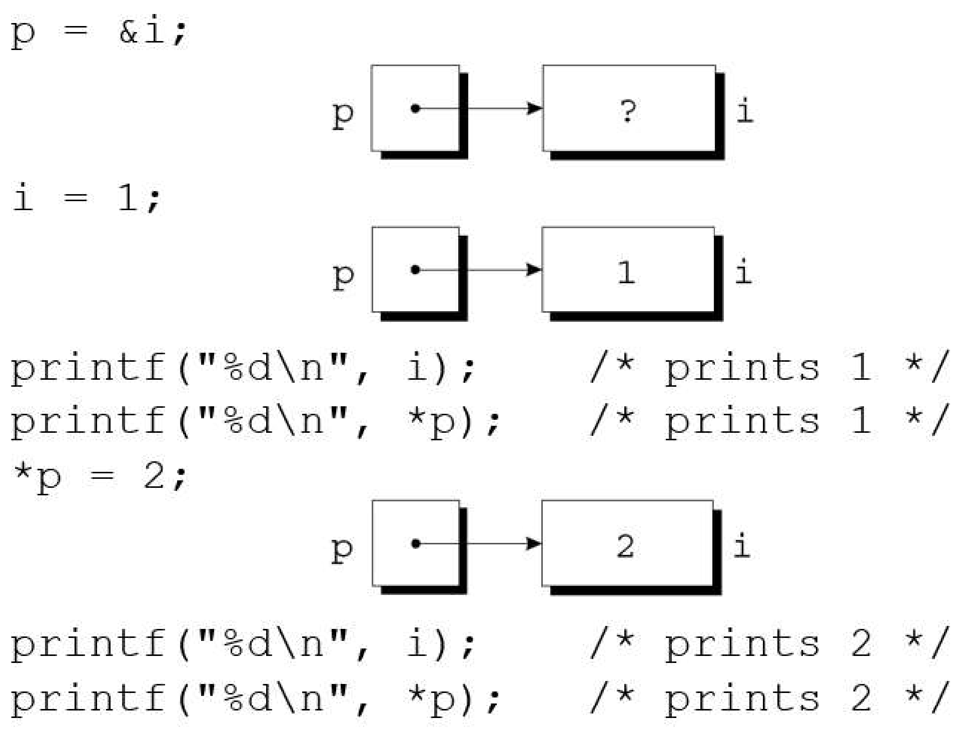
Change your theme

- In Emacs, type `M-x custom-themes`
- In the buffer that appears, select `Leuven`
- Select `Apply and Save Setting`
- This will work immediately

Indirection operator *

Put the code from the diagram into the code block below and run it to confirm the claims.

- Make sure you declare your variables!
- Comment your code to indicate you know what you're doing



```
p = &i;
i = 1;
printf("%d\n", i);     /* prints 1 */
printf("%d\n", *p);    /* prints 1 */
*p = 2;
printf("%d\n", i);     /* prints 2 */
printf("%d\n", *p);    /* prints 2 */
```

Figure 1: indirection operator (Source: King)

```
int i, *p;  // declare variable i, pointer variable p

p = &i; // initialize pointer with address of i

i = 1; // initialize integer variable with value

printf("%d\n", i); // prints value of i = 1

printf("%d\n", *p); // prints dereferenced pointer = 1

*p = 2; // initialize dereferenced pointer with value 2

printf("%d\n", i); // prints new value of i = 2

printf("%d\n", *p); // prints dereferenced pointer = 2
```

```
1
1
2
2
```

Initializing pointers

- [ ]

  The initialization of the pointer `iPtr` in the following code block went wrong:

- Fix the initialization so that the pointer is assigned an address, not a value
- Print the pointer variable `ptr`, the address and value of `x`

```
double x = 3.14159;
double *ptr;

// initialize pointer
ptr = x;      // ptr is assigned the address of x
ptr = 2.71828; // value of x is indirectly changed to e

// print pointer, address and value of i
...
```

Solution:

```
double x = 3.14159;
double *ptr;

// initialize pointer
ptr = &x;      // ptr is assigned the address of x
*ptr = 2.71828; // value of x is indirectly changed to e

// print pointer, address and value of i
printf("%p %p %g\n", ptr, &x, x);
```

```
0061FEC0 0061FEC0 2.71828
```

Fix the program

- The following function supposedly computes the sum and average of the numbers in the array `a`, which has length `n`. The variables `avg` and `sum` *point* to variables that the function should modify.

  Unfortunately, the function contains several errors:

  - find and correct them so that the code compiles

```
void avg_sum (double a[], int n, double *avg, double *sum) {
  int i;
  sum = 0.0;
  for (i = 0; i < n; i++) {
    sum += a[i];
  }
  avg = sum / n;
} // end of function
```

```
void avg_sum (double a[], int n, double *avg, double *sum) {
  int i;
  *sum = 0.0;
  for (i = 0; i < n; i++) {
   *sum += a[i];
  }
  *avg = *sum / n;
} // end of function
```

Created: 2022-04-27 Wed 10:03