B2 • Discussion #11 • 2021-04-09

{{Remind us @ 10:38 am EDT to stop so that you can take the quiz!}}

Announcements

- Next week we will do dynamic memory allocation, and then a new type of data structure: linked lists, in which structures are dynamically allocated and linked together using pointers.
- This material is dense. You must really work hard to do the pre-class activities, and participate fully in class (audio and video on to get the attendance grade but more important to LEARN!)
- Final project due the last class day (4/28) so you will have a lot of time to work on it. However, do not procrastinate!
- Exam #3 (just on C material): two weeks from today.

Review Material

Camera on! don't forget :)

FREE

- functions that return one value
- void functions
- program organization; function prototypes
- pointers
- call-by-reference
- use of * to declare pointers, and also as the dereferencing operator
- NOTE: use call-by-reference only when a function is calculating or initializing more than one value; if it is doing only one, use return instead
- ANOTHER NOTE: arrays (including character arrays) cannot be returned from a function

FUNCTION PROTOTYPES

```
#include <stdio.h>

typedef struct
 {
   int streetno;
   char streetname[15];
 }  street_t;


/* Fill in the function prototypes */
```

type: street_t

float calcstuff(int, char);

void dostuff(street_t, int *, float *);

← this is the answer!

```
int main()
{
   street_t mystreet,
            avenues[20];

   float value;

   int count = 0;

   char myname[20];

   /* Assume that EVERYTHING is initialized here, */
   /*  including ALL elements of all arrays and   */
   /*  ALL members of all structs                 */

   value = calcstuff(mystreet.streetno, myname[0]);

   dostuff(avenues[3], &count, &value);


   return 0;
}

/* Assume that both function definitions are here */
```

&count ← address of count
&value ← address of value

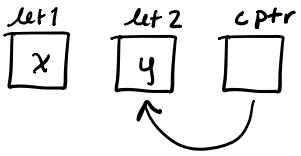
← example of call-by-value

← example of call-by-reference

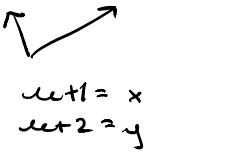
void dostuff(street_t strct, int *c, float *val)

VOID FUNCTIONS, POINTERS, AND CALL BY REFERENCE

```c
#include <stdio.h>

void dostuff(char *, char *);

int main( )
{
    char let1 = 'a',
         let2,
         *cptr;

    cptr = &let2;
    *cptr = 'e';
    printf("let1 is %c and let2 is %c\n", let1, let2);

    let2 = 'z';
    printf("let1 is %c and let2 is %c\n", let1, let2);
    printf("*cptr is %c\n", *cptr);

    dostuff(&let1, &let2);
    printf("let1 is %c and let2 is %c\n", let1, let2);

    return 0;
}

void dostuff(char *p1, char *p2)
{
    *p1 = 'x';
    *p2 = 'y';
}
```



let1 is a and let2 is e
let1 is a and let2 is z
*cptr is z
let1 is x and let2 is y



let1 = x
let2 = y