c loop stops when array value is zero not null

Asked 6 years, 5 months ago Modified 6 years, 5 months ago Viewed 10k times

5

```
int findMax(int*sums){
  int t = 0;
  int max = sums[0];
  while (sums[t] != '\0'){
    printf("current max: %d %d\n", max, sums[t]);
    if (sums[t] > max ){
      max = sums[t];
    }
    t ++;
  }
  return max;
}
```

This outputs:

```
current max: 7 7
current max: 7 4
current max: 7 2
```

And its ignoring the rest of the list, `sums`. I think this is because the next element in `sums` is `0`. But I can't see why it would treat `0` as `'\0'` (null).

c arrays while-loop int max

Share Follow

edited Jul 10, 2016 at 1:18

Cherubim
5,147 3 19 36

asked Jul 9, 2016 at 19:37

Karen McCulloch
99 1 1 9

3 `'\0'` is an `int` with the value `0`. There is no type difference. They are indistinguishable. – Weather Vane Jul 9, 2016 at 19:41

It coerces automatically - 0 becomes '\0' (and also becomes null, which is a different value of different type) – Elazar Jul 9, 2016 at 19:41

1 There is no such thing as `null int`, please see other comments. – user3078414 Jul 9, 2016 at 19:47

As mentioned, '\0' is actually just zero. You're treating sums[] as if it were a null-terminated array. Int arrays are not null-terminated in C (because it would be impossible to tell if a 0 in the array is a data element, or

@KarenMcCulloch are all your elements of array positive? Or is there any range to the elements you enter into array... – Cherubim Jul 9, 2016 at 22:35

6 Answers

Sorted by: Highest score (default)

3

`sums` is an array of integers (technically a pointer to integer). `'\0'` (the null byte) and 0 are the same value, so your loop will stop when it encounters a 0. There is no such thing as a null value as far as integers are concerned. The term "null" is used to refer to the value `NULL`, which is a pointer usually with the value 0 (i.e., a pointer that doesn't point to anything), and also the null (0) byte, such as the one that occurs at the end of a null-terminated string.

Share Follow

edited Jul 9, 2016 at 19:53

answered Jul 9, 2016 at 19:47

Andy Schweig
6,379 2 16 22

1 So can you iterate though an int list in C if you don't know how long the list is and zero could be in the list? – Karen McCulloch Jul 9, 2016 at 19:56

1 In C, there is no way to determine the length of an array represented by a pointer. In your function, you would have to pass the length as an additional parameter. – Andy Schweig Jul 9, 2016 at 20:00

1

- I do remember the time when I first encountered the same problem *(while I was trying to build a big number library using `int` arrays)*, and eventually I figured out pretty much the same as what other answers say that **technically `'\0'` and `0` have the same value.**
- Now here are `2` ways that I used to overcome this problem and these are only applicable under certain conditions

- **Case 1 :**

  Condition : When all your input elements are *positive*

- Now since all your input elements are *positive*, you can mark the end of the array by inserting a *negative* number
- Typically, I use `-1`, this way :

```
int a[] = {1, 2, 3, 4, -1}

for(int index = 0; a[index] != -1; index++)
```

- Instead you can enter any *negative* number and do it this way

```
for(int index = 0; a[index] >= 0; index++)
{
    //use the array element a[index] for desired purpose!
}
```

- **Case 2 :**

  Condition : When all your elements are bound within a certain *range*

- You might have got the idea by now :), lets say that all your elements belong to the *range* `[-100,100]`
- you can insert any number above or below the bounds of the range to mark the end... so in the above case I can mark the end by entering a number `< -100` and `>100` .
- And you can iterate the loop this way :

```
for(int index = 0; (a[index] > -100) && (a[index] < 100); index++)
{
    //use the array element a[index] for desired purpose!
}
```

  Generalizing both the cases, just place a value at the end of array which you know for sure is not equal to an array element

```
for(int index = 0; a[index] != value_not_in_array; index++)
{
    //use the array element a[index] for desired purpose!
}
```

So, now under **Case 1**, your while loop condition can be either of the following :

```
while(sums[t] != -1) //typically ended with `-1`
//(or)
while (sums[t] >= 0) //ended with any negative number
```

And under **Case 2** :

```
while ((sums[t] >min_range) && (sums[t] < max_range)) // when elements are bound
```

Or more generally :

```
while( sums[t] != value_not_in_array )
```

> **The underlying fact of both the cases is that I'm finding out a potential replacement for terminating `'\0'` character.**

Hope this helps, happy coding ;)

Share Follow

edited Jul 10, 2016 at 1:35

answered Jul 9, 2016 at 23:06

Cherubim
5,147 3 19 36

1

'\0' is a representation of a non-printable ASCII character. Specifically, it is the character 0 (as in, the zeroeth character, not the character '0', whichis 48. Look it up on an ASCII table).

'\0' is the same as 0 the same way 'A' is == 65. There is no difference as far as the compiler is concerned. '\0' == 0 will always evaluate as true.

Note that only strings are terminated with a '\0', unlike all other arrays.

Share Follow

edited Jul 9, 2016 at 19:55

answered Jul 9, 2016 at 19:50

Annonymus
194 1 1 10

0

In C, the character literal `'\0'` has the value `(int)0`, that's what the escape sequence translates to.

```
#include <stdio.h>
int main() {
    int i = 0;
    char c = '\0';

    printf("%s\n", (i == c) ? "same" : "different");
}
```

http://ideone.com/sYRbYZ

Share Follow

answered Jul 9, 2016 at 19:53

kfsone

1 @Annonymus See 6.4.4.4/2: "An integer character constant is a sequence of one or more multibyte characters enclosed in single-quotes, as in 'x'." – kfsone Jul 9, 2016 at 19:59

I see. Looks like I was misinformed. Thanks for clearing that up. – Annonymus Jul 10, 2016 at 5:15

0

I think you're confusing a pointer check for `NULL` vs a value check for zero.

Here are two slightly different variants of your function to illustrate the point:

```
#include <stdio.h>

int
findPtr(int **sums)
{
    int t = 0;
    int max = *sums[0];
    int val;

    while (sums[t] != NULL) {
        val = *sums[t];
        printf("current max: %d %d\n", max, val);
        if (val > max) {
            max = val;
        }
        t++;
    }

    return max;
}

int
findArr(int *sums,int count)
{
    int t = 0;
    int max = sums[0];

    while (t < count) {
        printf("current max: %d %d\n", max, sums[t]);
        if (sums[t] > max) {
            max = sums[t];
        }
        t++;
    }

    return max;
}
```

Since zero (either `0` or `\0` --which are equivalent) is a *valid* value in sums, it can't be used as a *sentinel* for end of array as your check was doing. You'll need to pass down the array count as in the latter example.

Craig Estey
28.3k 4 23 43

-1

In your code, you are taking a pointer to an array of integers as input in the findMax function. '\0' is a character. You are comparing integers to a character, causing the compiler to cast the character '\0' and use its integer equivalent NULL (or simply 0). Therefore the program stops when it comes to a 0 in the array. You might want to try :

```
int findMax(int*sums,int arraysize)
{
int t=0;
int max = sums[t];

while(t<arraysize)
{
    printf("current max: %d %d\n", max, sums[t]);
    if (sums[t] > max )
    {max = sums[t];}
 t++;
}
return max;
}
```

Share Follow

answered Jul 9, 2016 at 20:42

intersomnium
9 2