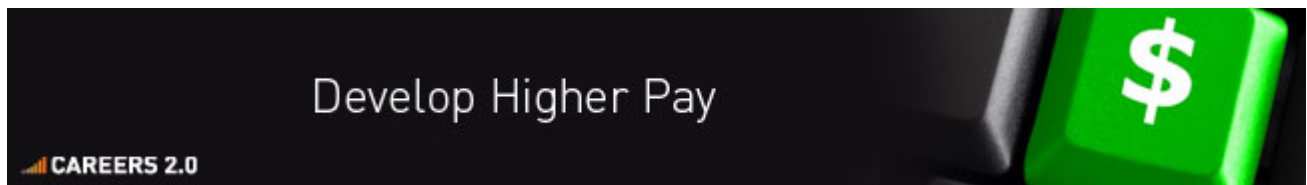


Stack Overflow is a question and answer site for professional and enthusiast programmers. It's free, no registration required.

What happens here? sizeof(short_int_variable + char



```
#include <stdio.h>
int main()
{
    short int i = 20;

    char c = 97;

    printf("%d, %d, %d\n", sizeof(i), sizeof(c), sizeof(c + i));
    return 0;
}
```

Could some one tell me what happens when sizeof(a+b) "a is short int type & b is char" Output is : 2, 1, 4

[c](#) [variables](#) [sizeof](#)

edited 42 mins ago



[mani](#)
1,074 1 12

asked 1 hour ago



[hem](#)
26 2

sizeof is a compile time construct. – [Alok Save](#) 1 hour ago

size of the result is calculated as int – [BLUEPIXY](#) 1 hour ago

You didn't mention yet that sizeof(c+c) will also give 4, so will sizeof(c/2) or sizeof(1998) [Kartikya](#) 36 mins ago

5 Answers

Because of C's standard [integral promotion rules](#), the type of the expression `c + i` is `int`, so that's why you're getting the equivalent of `sizeof (int)`.

Note that `sizeof` is not a function, the parentheses are only needed when naming and to resolve precedence conflicts. Your code could be written:

```
printf("%d, %d, %d\n", sizeof i, sizeof c, sizeof (c + i));
```

The final use of `sizeof` has parentheses since `sizeof` binds tighter than `+`.

Also note that it's a compile-time construct most of the time, the expression is never evaluated. All that happens is that the compiler "pretends" to evaluate it, to figure out the type of the expression, and then gives you the size of a value of that type. The actual value needs to exist, which is sometimes neat.

answered 57 mins ago



[unwind](#)
125k 16

And as you have already stated elsewhere, to avoid confusion, always put a whitespace between `sizeof` and the opening parenthesis. (I am just making this witty convention explicit for the reader) – [Rerito](#) 46 mins ago

Nice share. Thank you :) – [hem](#) 44 mins ago

CAREERS 2.0
by stackoverflow



+



Creating a profile just got easier
Import your LinkedIn to get started

1) `sizeof(i) ==> sizeof(short int) = 2`

2) `sizeof(c) ==> sizeof(char) = 1`

3) `sizeof(c + i [97+20]) ==> sizeof(int) = 4 // result in constant`

answered 59 mins ago



[mani](#)
1,074 1

downvoter. please explain what I understand wrong? I'll correct me.. – [mani](#) 57 mins ago

"result in constant value which is int as default" - Huh? I don't even understand what you are about. – [Christian Rau](#) 21 mins ago

@ChristianRau I talk about `(c+i)` which result in constant value which is taken as `int` default. – [mani](#) 9 mins ago

I don't why am I getting downvote again and again? For my `engl i sh` or Am I lead to wr answer? – [mani](#) 6 mins ago

As others have told, `sizeof` is computed at compile-time.

Here, value of the expression `c + i` integer, as `c` and `i` are promoted (integral promotion) and thus

```
sizeof( c + i )
```

gives you 4 bytes on 32-bit machine..

answered 54 mins ago



`sizeof` only works at compiletime to get the size of an expression.

The following won't actually increase 'c':

```
c = sizeof(++c);
```

The expression `sizeof(a + b)` will return the largest type with unsigned having precedence.

answered 1 hour ago



The data type of `a` is "short int". -32768 ~ +32767

The data type of `c` is "char". 0 ~ 255

When you add `a` to `c` it is not either **short int** nor **char**, it becomes **int**!

Here is an example code which help you to get the variable data type:

```
#include <typeinfo>
```

```
int main()
{
    short int a = 1;
    char c = 'A';
}
```

```
std::cout << typeid(a + c).name() << std::endl;  
  
return 1;  
}
```

edited 45 mins ago

answered 55 mir



ali-bah
29 2

A short has the range -32768 to 32767. – [Lindydancer](#) 54 mins ago

You are right ... I made a mistake. – [ali-bah](#) 46 mins ago

Not the answer you're looking for? Browse other questions tagged

[variables](#) [sizeof](#) or [ask your own question](#).