

# What is type safety and what are the “type safe” alternatives? [duplicate]

[Ask Question](#)

Asked 10 years, 6 months ago   Active 5 years, 6 months ago   Viewed 24k times



23



This question already has answers here:

Closed 10 years ago.

**Possible Duplicates:**

[What is Type-safe?](#)

[What is type-safety?](#)

I was reading about c++ vectors and it was mentioned that `memcpy` and `printf` functions from C are not type safe. Article here: [http://en.wikipedia.org/wiki/Vector\\_\(C%2B%2B\)](http://en.wikipedia.org/wiki/Vector_(C%2B%2B)).

Question: In simple English, what is type safety and what are the "type safe" alternatives?

`c++` `c` `type-safety`

[share](#) [improve this question](#) [follow](#)

edited May 23 '17 at 11:54



Community ♦

1 ● 1

asked Jan 26 '10 at 15:25



Dr Deo

4,595 ● 11 ● 35 ● 64

2

I'm sure we have answers to both of your questions on Stack Overflow already in separate questions. Will look up. In the meantime search for "type safety" and "memset memcpy std::fill" on stackoverflow. – [Mehrdad Afshari](#) Jan 26 '10 at 15:30

Voting to close as exact duplicate: [stackoverflow.com/questions/260626/what-is-type-safe](http://stackoverflow.com/questions/260626/what-is-type-safe)

Trying to check as exact definition: [https://en.cppreference.com/w/cpp/string/basic\\_string\\_view](https://en.cppreference.com/w/cpp/string/basic_string_view) [stackoverflow.com/questions/928275/what-is-type-safety](https://stackoverflow.com/questions/928275/what-is-type-safety) [stackoverflow.com/questions/1899906/...](https://stackoverflow.com/questions/1899906/...) – Sinan Ünür Jan 26 '10 at 15:52

add a comment

## 10 Answers

Active Oldest Votes



30

Type safety means that the compiler can check whether you're using the right types. For example, if you're using `printf`, you could accidentally crash your program by writing this:

```
printf("The meaning of life is %s", 42);
```



because 42 is an integer, not a string.

share improve this answer follow

edited Jan 26 '10 at 15:44



unwind

351k ● 58 ● 435 ● 566

answered Jan 26 '10 at 15:31



Chris Jester-Young

199k ● 43 ● 361 ● 408

[boost.org/doc/libs/1\\_45\\_0/libs/format/doc/format.html](http://boost.org/doc/libs/1_45_0/libs/format/doc/format.html) – Roman A. Taycher Feb 7 '11 at 4:13

6 Correct; C++ is a weakly typed system because you can essentially cast any type any other type, ints to bools and everything else. C++ gives the programmer complete control of the machine; memory is memory, and C++ will let you blow your leg off because it demands that you know precisely what you're doing every single step of the way. – Adam Miller Apr 1 '13 at 12:13

add a comment



12

Type safety means that the compiler will help check that you don't mix (incompatible) data types.

For instance, when you call `memcpy`, the function (and compiler) only sees two pointers in memory, and will happily start copying data. This means you can mix incompatible data types like this:



```
SomeClass a;  
AnotherClass b;  
memcpy((void*)&a, (void*)&b, sizeof(b));
```

There are many approaches to gaining type safety. You could use templates and make a wrapper around `mempcy()`, ensuring that the two pointers point to the same data type, or you could use other ways.

Since you are already using vectors from the STL, you are already using a more or less type safe implementation.

[share](#) [improve this answer](#) [follow](#)

[edited Jan 27 '10 at 7:32](#)

[answered Jan 26 '10 at 15:47](#)



[csl](#)

9,839 ● 4 ● 50 ● 83

[add a comment](#)



8



Type safety governs the usage of the compiler checking if the variable is of a right type. C is very loose on data type safety, for example, this is actually in the ANSI C standards, that states that type promotion will occur for data type `char`, an example in this assignment will explain this,

```
char ch = 32; /* that is a space character accordingly to ASCII */  
int n = ch + 3;
```

Notice how the `ch` variable gets 'promoted' to type `int`. That is legitimate but warrants closer inspection if that is what you are implying.

Compilers such as the C# compiler will not allow this to happen, that is the very reason why in C, there is a usage of cast's operator for example:

```
int n = (int)3.1415926535f;
```

Nit picky aside, that is a pi value, what happens, is that the value of `n` will be 3.

The above serves to illustrate the type safety and that C is very loose on this regard.

Type safety in modern languages is more strict, such as Java, C#, in order to constrain the usage and meaning of the variables. PHP is an excellent example of loose typing, where you could do this:

```
$myvar = 34;  
$myvar = $myvar + "foo";
```

is `$myvar` an integer, or is it a floating point or is it a string. The type safety here is not very clear on what is the intention which can lead to bugs and a happy debugging session trying to figure out what is happening.

Hope this helps

share improve this answer follow

edited Feb 18 '15 at 0:56



Chris Jester-Young

199k ● 43 ● 361 ● 408

answered Jan 26 '10 at 15:52



t0mm13b

32.2k ● 6 ● 66 ● 101

add a comment

Since you were on Wikipedia anyway: [Type safety](#).

2

Type safety means, roughly speaking, that the language prohibits you from accidentally mixing up your types.

`memcpy` is not type-safe because you can easily copy the memory of some `int` into a `char` array and end up with meaningless data. `printf` is not type safe because you can provide a `%i` format specifier with a string; again, the string will be interpreted as an `int` and you'll end up with garbage. (Incidentally, the VC++ compiler *does* check the format string in some situations.)

`std::vector<T>` is type-safe, because it only allows you to put values of the given type `T` into it. (You can do explicit typecasts, of course, but the point is that you have to be *explicit* about doing something that's not type safe).

share improve this answer follow

answered Jan 26 '10 at 15:42



Thomas

139k ● 40 ● 283 ● 396

5 I would like to know why this was downvoted. Did I say something wrong? – Thomas Jan 27 '10 at 10:30

Here's your upvote, mate ;) – Soup Endless May 15 at 16:48

add a comment



"Type safety" means that the compiler checks that you are doing the right things with the right types (e.g triggers a compiler error if you attempt to treat a Banana as an Orange, or give a string to a function expecting to output an integer).



Type safety (mostly) goes right out of the window when `void*` comes into the picture - it is a pointer that can point to anything (completely unaware of the types involved), and the language leaves going about with it completely in the programmers hands (for example, a `void*` isn't mostly good for anything except for being cast back to the original type; it can represent anything, but you have to know what it is before you can use it).

Type unsafety also comes to play with variadic functions like `printf` (the compiler doesn't care how many arguments there are and what their types are - again it is up to the caller to make sure that the format string matches the arguments and their types).

Type-safe alternative to `memcpy` (for arrays and containers) could be `std::copy` in `<algorithm>` - it may be implemented in terms of `memmove` if all involved types satisfy certain requirements, otherwise it performs assignments - with some classes you can break certain invariants if you bypass their public interface and just go and move / copy them around in memory (for example, I suppose any class with a non-trivial copy constructor is going to misbehave if you make copies of it with `memcpy`).

Type-safe alternative to C I/O routines are `iostreams` (and if you want the benefits of the format string, `boost::format`).

[share](#) [improve this answer](#) [follow](#)

edited Jan 26 '10 at 15:52

answered Jan 26 '10 at 15:43



UncleBens

37.5k ● 6 ● 51 ● 87

[add a comment](#)



"Type safety" is using a "type system" to ensure that errors are not propagated within programs. For example, without type safety, it might be possible to (silently) add a string type to floating point type in some undesirable way.



In the instances you're talking about, `memcpy()` and `printf()`, the lack of type safety is due to the how the functions treat their arguments. For example, with `memcpy(arg1, arg2, len)`, the `len` bytes starting at memory address `arg2` will be copied to memory address `arg1`, regardless of how many bytes `arg1` points to, potentially overwriting other portions of your program.

For type safe alternatives, look into [constructors](#) and [cout](#).

In fact, [look into the entire C++ FAQ Lite](#)

share improve this answer follow

edited Jan 26 '10 at 16:24

answered Jan 26 '10 at 15:44



Liz Albin

1,471 ● 8 ● 8

add a comment

▲  
1

It means that the compiler will generate no warning if you try to use a type in a way that doesn't make sense for that type. For example, the following is undefined behavior and in practice will copy the bits of a pointer into the bits of a float, where they make absolutely no sense. If `sizeof(char*) > sizeof(float)`, it will overwrite whatever memory locations happen to be just above where `f` lives.

▼  
🕒

```
float f;  
char *c = someString();  
memcpy(&f, &c, sizeof(char*));
```

share improve this answer follow

edited Jan 26 '10 at 17:12

answered Jan 26 '10 at 15:44



dsimcha

62.7k ● 42 ● 188 ● 316

add a comment

▲  
0

Type safety refers to a coding paradigm that enforces every variable to have a dedicated type at compilation time, for example `int a = 4; double d = 100.0; struct ms {char s;} mystruct;` The type of a variable is never 'lost'. If you want to change its type from `a` to `b`, an explicit or implicit conversion must be defined.



`printf` is *not* typesafe because you pass the arguments in a variadic argument list:



```
float f = 1.f;
printf("This is a float: %f\nAnd this is a string: %s", f, f);
```

`printf` does not know which kind of values she receives. The format string is used by the implementation to find out, but if the string is wrong, the implementation has no chance to find it out because there is no type information available at compile-time. The above `printf` call is most likely to end up catastrophic - `printf` expects a string as second parameter, but gets a floating-point number.

[share](#) [improve this answer](#) [follow](#)

answered Jan 26 '10 at 15:44



[Alexander Gessler](#)

41.5k ● 5 ● 73 ● 118

---

I just want to add that a type can be declared implicitly or explicitly. `a = 3;` clearly `a` is an `int`. – [Lay González](#) May 31 '13 at 4:37

---

[add a comment](#)



The signature of `memcpy` function is

0

```
void *memcpy (void* destination, const void* source, size_t num);
```



so as you can see it doesn't assume anything about pointers involved with the copy, they are just pointers. So if for example you want to copy a range of `ints` to a range of `floats` compiler won't complain about that.

**Type Safety** is a tool that helps developers to avoid certain errors by preventing some kind of erroneous code being compiled (and lately executed). It analyzes semantic aspect of source code to check if conversion between types and types in general are coherent.

What does that mean? It means that if your program passes the **type checking phase** you can be sure not to generate *CERTAIN KIND* of errors at run-time.

Of course sometimes you need to force this check not to be done, that's why you can use casts to force things to be what you want. Think about another example, `malloc`: it is defined to be

```
void* malloc (size_t size);
```

so when you want to allocate a pointer to `floats` for example you do:

```
float* ptr = (float*)malloc(sizeof(float)*COUNT);
```

You are forced to cast the result of the function to be `float*` otherwise the typecheck will find an assign of a `void*` to a `float*` but `void*` is too generic to be assigned so: **TYPE CHECK FAIL!**

That's why `memcpy` is not type-safe. It doesn't check anything, it just copy from a pointer to another pointer.

[share](#) [improve this answer](#) [follow](#)

answered Jan 26 '10 at 15:50



[Jack](#)

122k ● 26 ● 206 ● 310

[add a comment](#)

▲ A short version of the answer:

-1

```
class Person;
```

```
person.DoSomething(); // This is type safe.
```

```
void * p = &person; // You can now start doing unsafe things with p.
```

You can't pass a `Person` to `memcpy`. It only knows and cares about memory. Bytes.

[share](#) [improve this answer](#) [follow](#)

answered Jan 26 '10 at 15:46



[Daniel Daranas](#)


21.3k ● 9 ● 58 ● 104

You can pass a (pointer to a) `Person` to `memcpy` if `Person` is a POD class. I think what's more relevant to the issue of type-safety, is that you can't (for example) `memcpy` a `Person` to a destination that's not big enough. `std::copy`, which is type safe,

requires that the destination type be assignable from the source type which `memcpy` doesn't – [Steve Lesson](#) Jan 26 '10 at




requires that the destination type be assignable from the source type, which memcpy doesn't. [Steve Jessop](#) Jan 26 '10 at 17:08

- 1 @Steve Jessop: `std::copy` doesn't do any range-checking either (the destination buffer may be too small). IMO, the biggest issue with `memcpy` is that you treat a class as merely a collection of bytes (through the cast to `void*`), thus bypassing the copy constructor. Try copying a `std::string` with `memcpy` (run in debugger if it otherwise appears to work). - It would be complete impossible to use `memcpy` in template code, without somehow making sure you are only going to use it with POD types. – [UncleBens](#) Jan 26 '10 at 21:20 

[add a comment](#)


Not the answer you're looking for? Browse other questions tagged [c++](#) [c](#) [type-safety](#) or [ask your own question](#).


#### The Overflow Blog

 How we built it: our new Articles feature for Stack Overflow Teams

 Podcast 260: Silicon Valley Exodus

#### Featured on Meta

 CEO Blog: Some exciting news about fundraising

 Thank you, Geoff

#### Linked

260 [What is Type-safe?](#)

9 [What is type-safety?](#)

3 [Why is void\\* considered unsafe in C++?](#)

14 [NULL macro or NULL const](#)

16 Genericity vs type-safety? Using void\* in C

1 Cannot convert char(\*)[50] to char\* in assignment

## Related

2397 What is the difference between #include <filename> and #include "filename"?

3310 What are the differences between a pointer variable and a reference variable in C++?

2974 What does the explicit keyword mean?

1675 What is the effect of extern "C" in C++?

9045 What is the "-->" operator in C++?

3018 Improve INSERT-per-second performance of SQLite

1341 Why do we need virtual functions in C++?

2035 What is the copy-and-swap idiom?

2173 What is The Rule of Three?

2007 What does the ?!?! operator do in C?

## Hot Network Questions




















Does the halflife time of a radioactive material decrease if its temperature increases?







What are the differences between Q-Learning and A\*?



Why are the top application indicators (top bar icons) appearing black and white in Ubuntu 20.04?

-  What prevents people from using cash back credit cards to generate a revenue stream?
-  Property tax on land where you discover you can't build anything on?
-  Basing a civilization on maize, beans, and squash as staple foods?
-  Advice for first time mathematics TA during the COVID-19 pandemic
-  How can I communicate that the word limit prevents you from elaborating something in an essay?
-  Does Raspbian keep a security log file, and what is its location?
-  Why is the Star of the County Down referred to as Colleen?
-  Is it reasonable to use NFS on a production web server?
-  Why are American elections and politics receiving so much coverage in Canadian and European media?
-  Very precisely explaining when phase plays a role or doesn't play a role in QM
-  Can I enter the US if I am married to a US citizen and I have an ESTA?
-  Rotate a number
-  How to deal with this rejection email from mathematics of computation journal?
-  Does the idiom «to cross the pond» exist?
-  a utility to swap two files
-  Can anyone tell me which LEGO set this is?
-  When and where did the \$ convention for hexadecimal literals originate?

-  How long world it take to dismantle a city by hand?
-  Dealing with rejection anxiety as a reviewer
-  Does the monk's Step of the Wind ability allow them to double their jump distance without using Dash or Disengage as a bonus action?
-  Will SCOTUS be forced to rule on birthright citizenship soon?



## STACK OVERFLOW

Questions  
Jobs  
Developer Jobs Directory  
Salary Calculator  
Help  
Mobile  
Disable Responsiveness

## PRODUCTS

Teams  
Talent  
Advertising  
Enterprise

## COMPANY

About  
Press  
Work Here  
Legal  
Privacy Policy  
Contact Us

## STACK EXCHANGE NETWORK

Technology >  
Life / Arts >  
Culture / Recreation >  
Science >  
Other >

[Blog](#) [Facebook](#) [Twitter](#) [LinkedIn](#) [Instagram](#)

site design / logo © 2020 Stack Exchange Inc; user contributions licensed under cc by-sa.  
rev 2020.8.14.37407