How much memory do Enums take?

Asked 16 years, 2 months ago Modified 13 years, 3 months ago

Viewed 37k times

57

For example if I have an Enum with two cases, does it make take more memory than a boolean? Languages: Java, C++

java c++ memory enums

Share

Improve this question

Follow

edited Sep 27, 2008 at 13:11

Jonny Buchanan

62.7k ● 17 ● 145 ● 150

asked Sep 27, 2008 at 9:15

Jelo

12 Answers

Sorted by: Highest score (default)

55

In Java, an `enum` is a full-blown class:

> Java programming language enum types are much more powerful than their counterparts in other languages. The enum declaration defines a

> class (called an enum type). The enum class body can include methods and other fields.

In order to see the actual size of each `enum`, let's make an actual `enum` and examine the contents of the `class` file it creates.

Let's say we have the following `Constants` enum class:

```
public enum Constants {
  ONE,
  TWO,
  THREE;
}
```

Compiling the above `enum` and disassembling resulting `class` file with `javap` gives the following:

```
Compiled from "Constants.java"
public final class Constants extends java.lang.Enum{
    public static final Constants ONE;
    public static final Constants TWO;
    public static final Constants THREE;
    public static Constants[] values();
    public static Constants valueOf(java.lang.String);
    static {};
}
```

The disassembly shows that that each field of an `enum` is an instance of the `Constants` `enum` class. (Further analysis with `javap` will reveal that each field is initialized by creating a new object by calling the `new Constants(String)` constructor in the static initialization block.)

Therefore, we can tell that each `enum` field that we create will be at least as much as the overhead of creating an object in the JVM.

Share Improve this answer

Follow

edited Aug 28, 2011 at 4:52

answered Sep 27, 2008 at 9:23

coobird

161k ● 35 ● 214 ● 227

16 On 32-bit platforms, it'll be exactly the same size as an int - it's just a (32 bit) pointer to the enum instance. Only one instance of each enum value exists in memory.
– thenickdude Feb 3, 2010 at 23:42

4 -1 "a little bit more heavier than a plain integer"? What does that mean, and where do you get that from the text?
– Erick Robertson Aug 11, 2011 at 17:38

17

In Java, there should only be one instance of each of the values of your enum in memory. A reference to the enum then requires only the storage for that reference. Checking the value of an enum is as efficient as any other reference comparison.

Share Improve this answer

Follow

answered Sep 27, 2008 at 10:18

thenickdude

1,663 ● 1 ● 17 ● 18

8

`bool` might be implemented as a single byte, but typically in a structure it would be surrounded by other elements that have alignment requirements that would mean that the boolean would effectively be occupying at least as much space as an `int`.

Modern processors load data from main memory as a whole cache line, 64 bytes. The difference between loading one byte from L1 cache and loading four bytes is negligible.

If you're trying to optimise for cache lines in a very high-performance application, then you might worry about how big your enum is, but generally I'd say it's clearer to define an enum than to use a boolean.

Share Improve this answer

Follow

answered Sep 27, 2008 at 10:10

Mike Dimmick

9,792 ● 2 ● 26 ● 48

8

You would only worry about this when storing large quantities of enums. For Java, you may be able to use an EnumSet in some cases. It uses a bit vector internally which is very space efficient and fast.

http://java.sun.com/j2se/1.5.0/docs/api/java/util/EnumSet.html

Share Improve this answer

Follow

answered Sep 27, 2008 at 10:57

Tom

56.3k ● 3 ● 29 ● 35

4

In Java, it would take more memory. In C++, it would take no memory than required for a constant of the same type (it's evaluated at compile-time and has no residual significance at runtime). In C++, this means that the default type for an enum will occupy the same space as an int.

Share Improve this answer

Follow

answered Sep 27, 2008 at 9:27

Jeff Hubbard

9,882 ● 3 ● 32 ● 29

3

In ISO C++ there is no obligation for an enum to be larger than its largest enumerator requires. In particular, enum {TRUE, FALSE} may have sizeof(1) even when sizeof(bool)==sizeof(int). There is simply no requirement. Some compilers make the enums the same size as an int. That is a compiler feature, which is allowed because the standard only imposes a minimum. Other compilers use extensions to control the size of an enum.

Share Improve this answer

Follow

answered Sep 29, 2008 at 12:50

MSalters

179k ● 11 ● 164 ● 368

I think most use the native size for the CPU architecture. 32-bit aligned data is fast to retrieve for most 32-bit CPUs, so most types end up as multiples of 32 bits. – gbjbaanb Oct 19, 2008 at 18:48

2

```
printf("%d", sizeof(enum));
```

Share Improve this answer

Follow

answered Sep 27, 2008 at 9:30

Patrick

92.4k ● 11 ● 52 ● 61

1

In C++ an enum is typically the same size as an `int`. That said it is not uncommon for compilers to provide a command line switch to allow the size of the enum to be set to the smallest size that fits the range of values defined.

Share Improve this answer

Follow

answered Sep 27, 2008 at 23:29

Andrew Edgecombe

40.3k ● 3 ● 38 ● 63

0

No, an enum is generally the same size as an int, same as boolean.

Share Improve this answer

answered Sep 27, 2008 at 9:18

Follow

Serafina Brocious
30.6k ● 12 ● 91 ● 115

It could, hence 'generally'. If someone is asking such a question without further details, though, edge cases like that are rarely important. – Serafina Brocious Sep 27, 2008 at 9:24

The Java and C++ tags indicate that this comment contains the correct answer in this context. – Jeffrey L Whitledge Sep 27, 2008 at 13:22

OK, on further research, it appears the answer for Java is not correct. (OT: It is the same as an int in C#, though.)
– Jeffrey L Whitledge Sep 27, 2008 at 13:28

In C++ enum and int same. bool and char same. – Loki Astari Sep 27, 2008 at 18:50

0

If your enum will ever have only two cases, indeed using a boolean instead might be a better idea (memory size, performance, usage/logic), even more in Java.
If you are wondering about memory cost, it might imply you plan to use lot of them. In Java you can use BitSet class, or on a smaller scale, in both languages you can manipulate bits with bitwise operations.

Share Improve this answer

Follow

answered Sep 27, 2008 at 9:35

PhiLho
41.1k ● 6 ● 99 ● 136

0

sizeof(enum) depends upon what you have in the enum. I was recently trying to find the size of an ArrayList() with default constructor params and no objects stored inside (which means the capacity to store is 10). It turned out that ArrayList is not too big < 100 bytes.

So, sizeof(enum) for a very simple enum should be less than 10 bytes. you can write a small program, give it a certain amount of memory and then try allocating enums. you should be able to figure it out(that's how i found out the memory of ArrayList)

BR,
~A

Share Improve this answer

Follow

answered Sep 27, 2008 at 18:14

anjanb
13.8k • 19 • 80 • 106

0

In C/C++ an enum will be the same size as an int.

With gcc you can add **attribute**((packed)) to the enum definition to make it take the minimum footprint. If the largest value in the enum is < 256 this will be one byte, two bytes if the largest value is < 65536, etc.

```
typedef enum {
    MY_ENUM0,
    MY_ENUM1,
    MY_ENUM2,
    MY_ENUM3,
    MY_ENUM4,
```

```
    MY_ENUM5
} __attribute__((packed)) myEnum_e;
```

Share Improve this answer Follow

edited Sep 28, 2008 at 3:39

answered Sep 28, 2008 at 3:18

DGentry
16.3k ● 8 ● 53 ● 66

I am fairly sure __attribute__((packed)) is a GCC-only extension. If I remember correctly, C99 does not specify a standard pragma for "packing", so every compiler does it differently. – Michael Ratanapintha Sep 28, 2008 at 3:29

You're probably right. It was my recollection that packed was a C99 feature, but my memory is often cantankerous and uncooperative. I removed the C99 reference. – DGentry Sep 28, 2008 at 3:39