Is there a decent wait function in C++?

Asked 14 years, 7 months ago Modified 4 years, 5 months ago Viewed 458k times

63

One of the first things I learned in C++ was that

```
#include <iostream>
int main()
{
    std::cout<<"Hello, World!\n";
    return 0;
}
```

would simply appear and disappear extremely quickly without pause. To prevent this, I had to go to notepad, and save

```
helloworld.exe
pause
```

ase

```
helloworld.bat
```

This got tedious when I needed to create a bunch of small test programs, and eventually I simply put `while(true);` at the end on most of my test programs, just so I could see the results. Is there a better wait function I can use?

c++ wait

Share Edit Follow

edited Apr 2, 2016 at 14:41
Lightness Races in Orbit
381k 77 650 1063

asked May 23, 2009 at 19:23
user98188

1 put a breakpoint on the return function – Mooing Duck Feb 20, 2013 at 20:01

1 It disappears because the OS (MS Windows, right?) opens a new window for the program's output and closes it when the program terminates. If you run the program from a command prompt the window won't go away. – Keith Thompson Dec 16, 2015 at 19:46

4 @KeithThompson: Or simply supply the `/K` switch to leave the prompt open after the requested program has terminated. No need for all these ugly (and, for some, 100% non-portable) abstraction leaks given down below. – Lightness Races in Orbit Apr 2, 2016 at 14:30

@BarryTheHatchet: Supply the `/K` switch to what? (I don't use Windows much.) – Keith Thompson Apr 2, 2016 at 18:54

1 @KeithThompson: Yes, exactly. The button isn't magical - at some point after you click the button, your program is launched, by invoking some sort of command. Typically it is a matter of configuration as to how that takes place. If it already is invoked as an argument to `cmd.exe`, you need only add `/K`. Otherwise you may be able to simply prepend `cmd.exe /K`. The only caveat, I concede, is that I don't know how well this plays with the VS debugger. – Lightness Races in Orbit Apr 2, 2016 at 22:23

14 Answers

Sorted by: Highest score (default)

79

you can require the user to hit enter before closing the program... something like this works.

```
#include <iostream>
int main()
{
  std::cout << "Hello, World\n";
  std::cin.ignore();
  return 0;
}
```

The cin reads in user input, and the .ignore() function of cin tells the program to just ignore the input. The program will continue once the user hits enter.

Link

Share Edit Follow

edited Aug 27, 2013 at 18:28
user283145

answered May 23, 2009 at 19:27
DeadHead
2,249 16 16

24 But **please *don't* do this**!! It's a *massive* abstraction leak and completely ruins the chainability of your program. Learn to properly operate your command-line environment instead. e.g. `cmd /K myprogram.exe` – Lightness Races in Orbit Apr 2, 2016 at 14:28

3 What do you mean by abstraction leak? @LightnessRacesinOrbit – Weston McNamara Apr 17, 2022 at 16:33

62

Please note that the code above was tested on Code::Blocks 12.11 and Visual Studio 2012 on Windows 7.

For forcing your programme stop or wait, you have several options :

- **sleep(unsigned int)**

The value has to be a positive integer in millisecond. That means that if you want your programme wait for 2 seconds, enter 2000.

Here's an example :

```
#include <iostream>     //for using cout
#include <stdlib.h>     //for using the function sleep

using namespace std;    //for using cout

int main(void)
{
   cout << "test" << endl;
   sleep(5000);         //make the programme waiting for 5 seconds
   cout << "test" << endl;
   sleep(2000);         // wait for 2 seconds before closing

   return 0;
}
```

If you wait too long, that probably means the parameter is in seconds. So change it to this:

```
sleep(5);
```

For those who get error message or problem using sleep try to replace it by _sleep or Sleep especially on Code::Bloks.
And if you still getting problems, try to add of one this library on the beginning of the code.

```
#include <stdio.h>
#include <time.h>
#include <unistd.h>
#include <dos.h>
#include <windows.h>
```

- **system("PAUSE")**

A simple "Hello world" programme on windows console application would probably close before you can see anything. That the case where you can use system("Pause").

```
#include <iostream>

using namespace std;

int main(void)
{
    cout << "Hello world!" << endl;

    system("PAUSE");
    return 0;
}
```

If you get the message "error: 'system' was not declared in this scope" just add the following line at the biggining of the code :

```
#include <cstdlib>
```

- **cin.ignore()**

The same result can be reached by using cin.ignore() :

```
#include <iostream>

using namespace std;

int main(void)
{
    cout << "Hello world!" << endl;

    cin.ignore();

    return 0;
}
```

- **cin.get()**

example :

```
#include <iostream>

using namespace std;

int main(void)
{
    cout << "Hello world!" << endl;

    cin.get();

    return 0;
}
```

- **getch()**

Just don't forget to add the library conio.h :

```
#include <iostream>
#include <conio.h>    //for using the function getch()
```

```
using namespace std;

int main(void)
{

    cout << "Hello world!" << endl;

    getch();

    return 0;
}
```

You can have message telling you to use _getch() insted of getch

Share Edit Follow

edited Oct 9, 2017 at 3:34
Caleb Sim
5 3

answered Feb 20, 2013 at 19:58
borderless
948 6 5

6 just to add, sleep is for linux while Sleep (Note the uppercase) is for Windows. – Ruchir Jul 31, 2015 at 6:50

10 But **please *don't* do this**!! It's a *massive* abstraction leak and completely ruins the chainability of your program. Learn to properly operate your command-line environment instead. e.g. `cmd /K myprogram.exe` – Lightness Races in Orbit Apr 2, 2016 at 14:29

`conio.h` is not part of standard c or c++ library.You cant use ut every where.Like in Unix. – user8435497 Sep 2, 2017 at 8:50

57

Lots of people have suggested POSIX `sleep`, Windows `Sleep`, Windows `system("pause")`, C++ `cin.get()` … there's even a DOS `getch()` in there, from roughly the late 1920s.

***Please* don't do any of these.**

None of these solutions would pass code review in my team. That means, if you submitted this code for inclusion in our products, your commit would be blocked and you would be told to go and find another solution. (One might argue that things aren't so serious when you're just a hobbyist playing around, but I propose that developing good habits in your pet projects is what will make you a valued professional in a business organisation, and keep you hired.)

Keeping the console window open so you can read the output of your program is *not* the responsibility of your program! When you add a wait/sleep/block to the end of your program, you are violating the single responsibility principle, creating a massive abstraction leak, and obliterating the re-usability/chainability of your program. It no longer takes input and gives output — it blocks for transient usage reasons. This is very non-good.

Instead, you should configure your *environment* to keep the prompt open after your program has finished its work. **Your Batch script wrapper is a good approach!** I can see how it would be annoying to have to keep manually updating, and you can't invoke it from your IDE. **You could**

make the script take the path to the program to execute as a parameter, and configure your IDE to invoke it instead of your program directly.

An interim, quick-start approach would be to **change your IDE's run command** from `cmd.exe <myprogram>` or `<myprogram>`, to `cmd.exe /K <myprogram>`. The `/K` switch to `cmd.exe` **makes the prompt stay open after the program at the given path has terminated**. This is going to be slightly more annoying than your Batch script solution, because now you have to type `exit` or click on the red 'X' when you're done reading your program's output, rather than just smacking the space bar.

I assume usage of an IDE, because otherwise you're already invoking from a command prompt, and this would not be a problem in the first place. Furthermore, I assume the use of Windows (based on detail given in the question), but this answer applies to any platform… which is, incidentally, half the point.

Share Edit Follow

answered Apr 2, 2016 at 14:37
Lightness Races in Orbit
381k 77 650 1063

7 This seems like overkill - I understand the general point, but this advice seems confusing (and, too early) to a new user who is just looking for the `sleep` function and integration in the IDE – Coruscate5 Sep 11, 2019 at 15:42

5 @Coruscate5 It's probably not necessary for a week-one programmer, no. The question wasn't constrained to such people. I'd certainly like to see people told not to use these "pause" tricks at least at *some* point during their education. I can't stress enough not only how they are inappropriate for the given task, but how choosing them for this task belies a misapprehension of the very concept of single-responsibility/abstraction, which is a fundamental flaw in any developer that must be overcome before one can produce quality software. – Lightness Races in Orbit Sep 11, 2019 at 15:51

4 *(cont.)* It's the "my car won't start; I fixed it by attaching a sail" approach. It's a mindset that *must* be overcome and/or untaught. – Lightness Races in Orbit Sep 11, 2019 at 15:54

27

The appearance and disappearance of a window for displaying text is a feature of how you are running the program, not of C++.

Run in a persistent command line environment, or include windowing support *in* your program, or use `sleep` or wait on input as shown in other answers.

Share Edit Follow

answered May 23, 2009 at 20:35
dmckee --- ex-moderator kitten
99.3k 25 144 235

2 Why is this not massively and superlatively upvoted :(– Lightness Races in Orbit Apr 2, 2016 at 14:26

4 Er ... presumably because most people coming to this questions are looking for a way to "*Make it stop doing that!*" rather than a discussion of why it's happening. Maybe that's a little sad, in a philosophical kind of way, but that's Q&A for you. – dmckee --- ex-moderator kitten Apr 2, 2016 at 14:46

It's very sad :(I've added my own answer to address this problem a little more. – Lightness Races in Orbit Apr 2, 2016 at 14:54

13

There is a C++11 way of doing it. It is quite simple, and I believe it is portable. Of course, as Lightness Races in Orbit pointed out, you should not do this in order to be able to see an *Hello World* in your terminal, but there exist some good reason to use a wait function. Without further ado,

```
#include <chrono> // std::chrono::microseconds
#include <thread> // std::this_thread::sleep_for
std::this_thread::sleep_for(std::chrono::microseconds{});
```

More details are available here. See also sleep_until.

Share Edit Follow edited Jul 15, 2019 at 10:57 answered Feb 4, 2017 at 21:37

Kevin
1,203 14 16

1 Hint: the C++ methods that allow you to give stuff like "2s" as argument, use some template code that create STB_GNU_UNIQUEs. If used in a shared lib, it will prevent the lib from being able to unload. See bugzilla.redhat.com/show_bug.cgi?id=1215452 – user1050755 Mar 20, 2020 at 5:21

13

the equivalent to the batch program would be

```
#include<iostream>
int main()
{
    std::cout<<"Hello, World!\n";
    std::cin.get();
    return 0;
}
```

The additional line does exactly what `PAUSE` does, waits for a single character input

Share Edit Follow answered May 23, 2009 at 19:27

SingleNegationEliminati
on
153k 33 267 306

7 But **please *don't* do this**!! It's a *massive* abstraction leak and completely ruins the chainability of your program. Learn to properly operate your command-line environment instead. e.g. `cmd /K`

`myprogram.exe` – Lightness Races in Orbit Apr 2, 2016 at 14:29

7

Actually, contrary to the other answers, I believe that OP's solution is the one that is most elegant.

Here's what you gain by using an external `.bat` wrapper:

1. The application obviously waits for user input, so it already does what you want.
2. You don't clutter the code with awkward calls. Who should wait? `main()` ?
3. You don't need to deal with cross platform issues - see how many people suggested `system("pause")` here.
4. Without this, to test your executable in automatic way in black box testing model, you need to simulate the `enter` keypress (unless you do things mentioned in the footnote).
5. Perhaps most importantly - should any user want to run your application through terminal (`cmd.exe` on Windows platform), they **don't** want to wait, since they'll see the output anyway. With the `.bat` wrapper technique, they can decide whether to run the `.bat` (or `.sh`) wrapper, or run the executable directly.

Focusing on the last two points - with any other technique, I'd expect the program to offer at least `--no-wait` switch so that I, as the user, can use the application with all sort of operations such as piping the output, chaining it with other programs etc. These are part of normal CLI workflow, and adding waiting at the end when you're already inside a terminal just gets in the way and destroys user experience.

For these reasons, IMO `.bat` solution is the nicest here.

Share Edit Follow

answered Oct 17, 2015 at 11:14
rr-
14.4k 6 45 67

4

What you have can be written easier. Instead of:

```
#include<iostream>
int main()
{
    std::cout<<"Hello, World!\n";
    return 0;
}
```

write

```
#include<iostream>
int main()
```

```
{
    std::cout<<"Hello, World!\n";
    system("PAUSE");
    return 0;
}
```

The system function executes anything you give it as if it was written in the command prompt. It suspends execution of your program while the command is executing so you can do anything with it, you can even compile programs from your cpp program.

Share Edit Follow

answered May 24, 2009 at 2:04

poliklosio
280 1 4

4 This solution is as unportable as it gets. – rr- Oct 17, 2015 at 9:18

2

Syntax:

void sleep(unsigned seconds);

sleep() suspends execution for an interval (seconds). With a call to sleep, the current program is suspended from execution for the number of seconds specified by the argument seconds. The interval is accurate only to the nearest hundredth of a second or to the accuracy of the operating system clock, whichever is less accurate.

This example should make it clear:

```
#include <dos.h>
#include <stdio.h>
#include <conio.h>

int main()
{
   printf("Message 1\n");
   sleep(2); //Parameter in sleep is in seconds
   printf("Message 2 a two seconds after Message 1");
   return 0;
}
```

Remember you have to #include dos.h

EDIT:

You could also use winAPI.

```
VOID WINAPI Sleep(
DWORD dwMilliseconds
);
```

Sleep Function(Windows)

Just a note,the parameter in the function provided by winapi is in milliseconds ,so the sleep line at the code snippet above would look like this "Sleep(2000);"

Share Edit Follow

answered May 23, 2009 at 19:28

Ivan Prodanov
34.9k 78 176 249

"sleep" and "wait" are not the same thing – Lightness Races in Orbit Apr 2, 2016 at 14:29

-1

getchar() provides a simplistic answer (waits for keyboard input). Call Sleep(milliseconds) to sleep before exit. Sleep function (MSDN)

Share Edit Follow

answered May 23, 2009 at 19:28

Eric Bainville
9,798 1 25 27

-2

Before the return statement in you main, insert this code:

system("pause");

This will hold the console until you hit a key.

```
#include<iostream>
#include<string>

using namespace std;

int main()
{
    string s;
    cout << "Please enter your first name followed by a newline\n";
    cin >> s;
    cout << "Hello, " << s << '\n';
    system("pause");
    return 0; // this return statement isn't necessary
}
```

Share Edit Follow

answered Feb 10, 2013 at 9:03

robert
149 1 4

-2

Well, this is an old post but I will just contribute to the question -- someone may find it useful later:

adding 'cin.get();' function just before the return of the main() seems to always stop the program from exiting before printing the results: see sample code below:

int main(){ string fname, lname;

```
    //ask user to enter name first and last name
    cout << "Please enter your first name: ";
    cin >> fname;

    cout << "Please enter your last name: ";
    cin >> lname;
    cout << "\n\n\n\nyour first name is: " << fname << "\nyour last name is: "
    << lname <<endl;

    //stop program from exiting before printing results on the screen
    cin.get();
    return 0;
```

}

Share Edit Follow

answered Jan 9, 2011 at 17:53
Phil Kelly
11

6 A lot of people have already answered the exact same thing before you. No need to clutter this page up. – wrongusername Aug 17, 2011 at 23:01

-2

You can use sleep() or usleep().

```
// Wait 5 seconds
sleep( 5 );
```

Share Edit Follow

answered May 23, 2009 at 19:27
jthompson
7,238 2 34 33

7 Note that sleep() isn't part of the C standard. It's defined by Posix, and I believe you have to use Sleep() on Win32. – Bastien Léonard May 23, 2009 at 19:53

The second thing to learn (one would argue that this should be the first) is the command line interface of your OS and compiler/linker flags and switches.

-3

Share Edit Follow

answered May 23, 2009 at 19:47

Nikolai Fetissov
82.8k 12 112 172

1 Providing an example of these flags would make this a good answer. Your approach is certainly optimal.
– Lightness Races in Orbit Apr 2, 2016 at 14:27