```
/* The following code example is taken from the book
 * "The C++ Standard Library - A Tutorial and Reference, 2nd Edition"
 * by Nicolai M. Josuttis, Addison-Wesley, 2012
 *
 * (C) Copyright Nicolai M. Josuttis 2012.
 * Permission to copy, use, modify, sell and distribute this software
 * is granted provided this copyright notice appears in all copies.
 * This software is provided "as is" without express or implied
 * warranty, and with no claim as to its suitability for any purpose.
 */
#include <future>
#include <thread>
#include <chrono>
#include <random>
#include <iostream>
#include <exception>
using namespace std;

void doSomething (char c)
{
    // random-number generator (use c as seed to get different sequences)
    default_random_engine dre(c);
    uniform_int_distribution<int> id(10,1000);

    // loop to print character after a random period of time
    for (int i=0; i<10; ++i) {
        this_thread::sleep_for(chrono::milliseconds(id(dre)));
        cout.put(c).flush();
    }
}

int main()
{
    cout << "starting 2 operations asynchronously" << endl;

    // start two loops in the background printing characters . or +
    auto f1 = async([]{ doSomething('.'); });
    auto f2 = async([]{ doSomething('+'); });

    // if at least one of the background tasks is running
    if (f1.wait_for(chrono::seconds(0)) != future_status::deferred ||
        f2.wait_for(chrono::seconds(0)) != future_status::deferred) {
        // poll until at least one of the loops finished
        while (f1.wait_for(chrono::seconds(0)) != future_status::ready &&
               f2.wait_for(chrono::seconds(0)) != future_status::ready) {
            //...;
            this_thread::yield();  // hint to reschedule to the next thread
        }
    }
    cout.put('\n').flush();

    // wait for all loops to be finished and process any exception
    try {
        f1.get();
        f2.get();
    }
```

```
    catch (const exception& e) {
        cout << "\nEXCEPTION: " << e.what() << endl;
    }
    cout << "\ndone" << endl;
}
```