A spool of thread

Threads

c++ threads

- #include <thread>
- http://www.cplusplus.com/reference/thread/thread/
- standard library in c++11

thread

```
#include <iostream>
#include <thread>
using namespace std;

void func(){
   //do some work
   cout << 5 << endl;
}

void fund( int me ){
   //do some work
   cout << me*2 << endl;
}

int main(){
   thread thd(func);
   thread the(fund, 4);

   thd.join();
   the.join();

   cout << “done” << endl;
   return 0;
}
```

thread

```
#include <iostream>
#include <thread>
using namespace std;

void func(){
   //do some work
   cout << 5 << endl;
}

void fund( int me ){
   //do some work
   cout << me*2 << endl;
}

int main(){
   thread thd(func);
   thread the(fund, 4);

   thd.join();
   the.join();

   cout << “done” << endl;
   return 0;
}
```

Which will print first?

Constructor makes thread ready to run

main will wait until *thd* is done then wait until *the* is done

A problem

- A sunken treasure ship has just been found
- How to retrieve the golden doubloons?
    - How much treasure?
- Hire divers to retrieve the treasure
    - How much did a diver retrieve?

thread

```
#include <iostream>
#include <thread>
using namespace std;
int treasure = 1000;
int payment = 0;

void fetch( ){
    while(treasure > 0){
        treasure -= 13;
        payment += 13;
    }
}

int main(){
    thread diver(fetch);
    diver.join();
    cout << payment << endl;
    return 0;
}
```

thread

Can treasure become negative?

```
#include <iostream>
#include <thread>
using namespace std;
int treasure = 1000;
int payment = 0;

void fetch( ){
  while(treasure > 0){
    treasure -= 13;
    payment += 13;
  }
}

int main(){
  thread diver(fetch);
  diver.join();
  cout << payment << endl;
  return 0;
}
```

thread

```
#include <iostream>
#include <thread>
using namespace std;
int treasure = 1000;
int payment = 0;

void fetch( ){
    while(treasure > 0){
        int amt = min(treasure,13);
        treasure -= amt;
        payment += amt;
    }
}

int main(){
    thread diver(fetch);
    diver.join();
    cout << payment << endl;
    return 0;
}
```

thread

```
#include <iostream>
#include <thread>
#include <vector>
using namespace std;
int treasure = 1000;
int payment = 0;

void fetch( ){
    while(treasure > 0){
        int amt = min(treasure,13);
        treasure -= amt;
        payment += amt;
    }
}

int main(){
    vector<thread*> divers;
    for(int i = 0; i < 4; ++i){
        divers.push_back(new thread(fetch));
    }
    for(thread* t: divers){
        t->join();
    }
    cout << payment << endl;
    return 0;
}
```