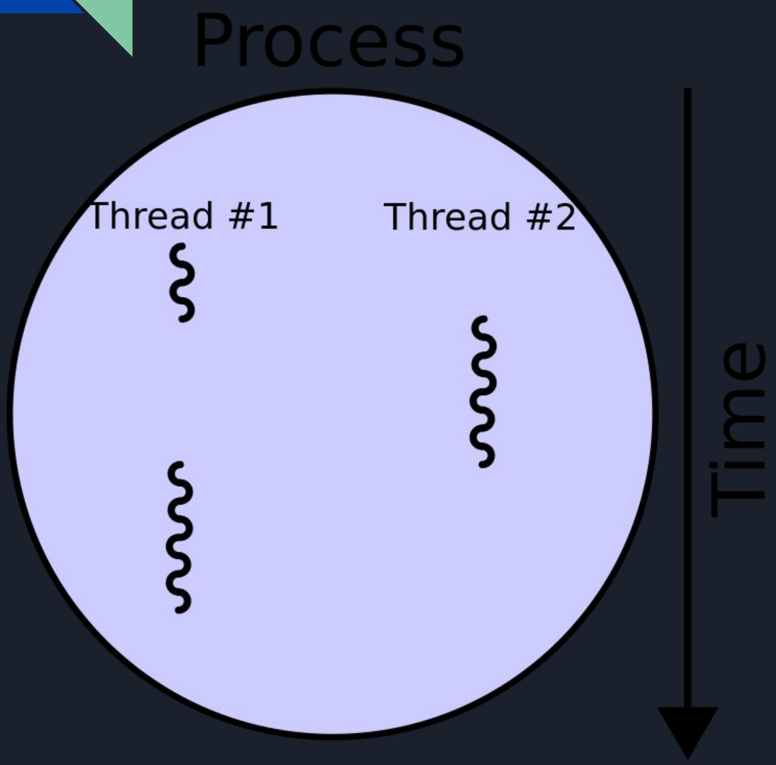
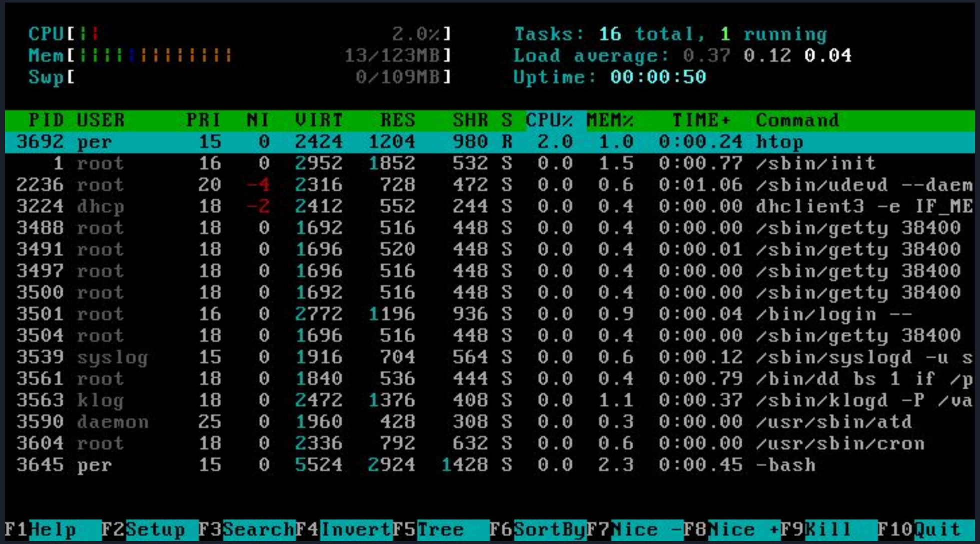
Wątki w C++ i nie tylko

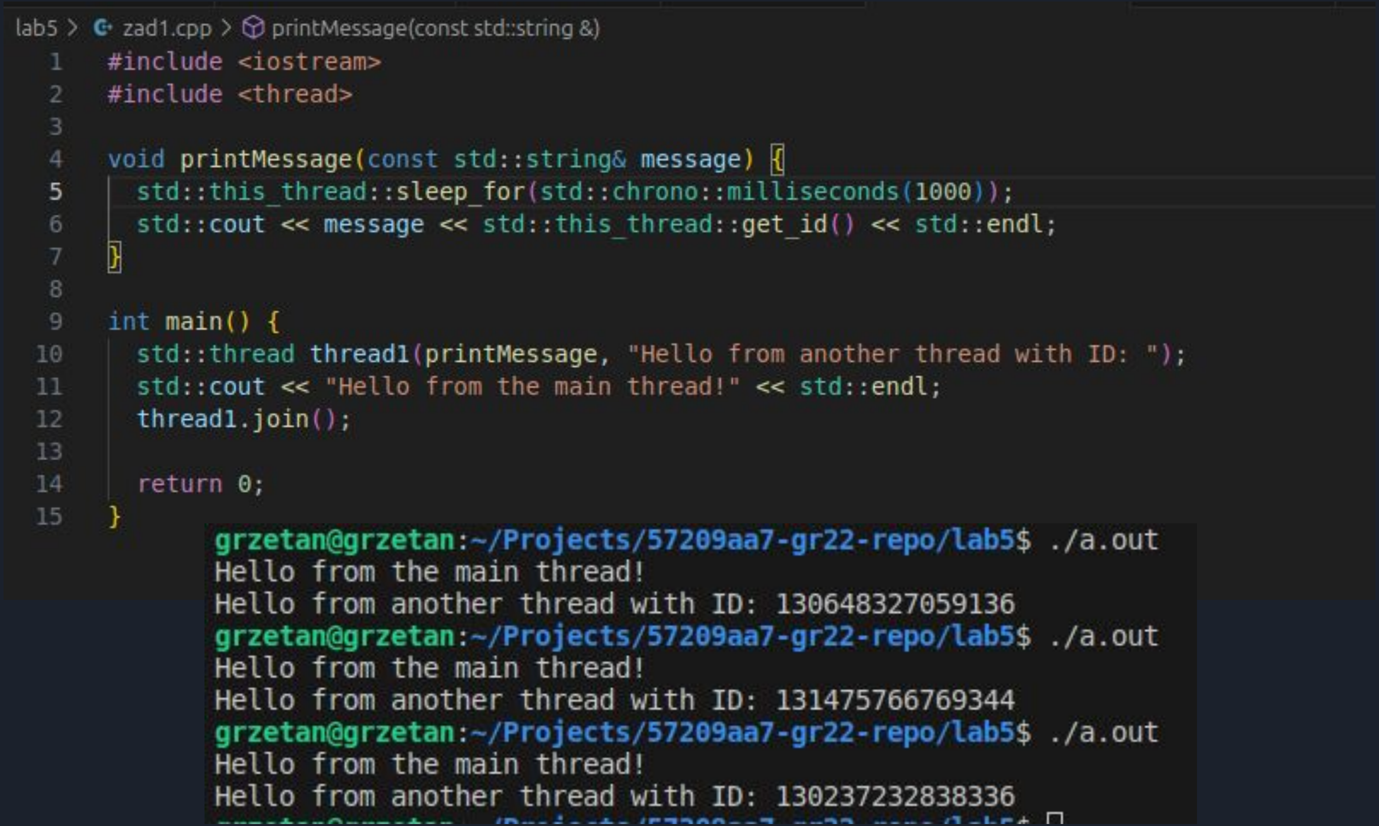
Czym jest wątek?





```
 CPU[||                              2.0%]   Tasks: 16 total, 1 running
 Mem[||||||||||||||||             13/123MB]   Load average: 0.37 0.12 0.04
 Swp[                              0/109MB]   Uptime: 00:00:50

  PID USER      PRI  NI  VIRT   RES   SHR S CPU% MEM%   TIME+  Command
 3692 per        15   0  2424  1204   980 R  2.0  1.0  0:00.24 htop
    1 root       16   0  2952  1852   532 S  0.0  1.5  0:00.77 /sbin/init
 2236 root       20  -4  2316   728   472 S  0.0  0.6  0:01.06 /sbin/udevd --daem
 3224 dhcp       18  -2  2412   552   244 S  0.0  0.4  0:00.00 dhclient3 -e IF_ME
 3488 root       18   0  1692   516   448 S  0.0  0.4  0:00.00 /sbin/getty 38400
 3491 root       18   0  1696   520   448 S  0.0  0.4  0:00.01 /sbin/getty 38400
 3497 root       18   0  1696   516   448 S  0.0  0.4  0:00.00 /sbin/getty 38400
 3500 root       18   0  1692   516   448 S  0.0  0.4  0:00.00 /sbin/getty 38400
 3501 root       16   0  2772  1196   936 S  0.0  0.9  0:00.04 /bin/login --
 3504 root       18   0  1696   516   448 S  0.0  0.4  0:00.00 /sbin/getty 38400
 3539 syslog     15   0  1916   704   564 S  0.0  0.6  0:00.12 /sbin/syslogd -u s
 3561 root       18   0  1840   536   444 S  0.0  0.4  0:00.79 /bin/dd bs 1 if /p
 3563 klog       18   0  2472  1376   408 S  0.0  1.1  0:00.37 /sbin/klogd -P /va
 3590 daemon     25   0  1960   428   308 S  0.0  0.3  0:00.00 /usr/sbin/atd
 3604 root       18   0  2336   792   632 S  0.0  0.6  0:00.00 /usr/sbin/cron
 3645 per        15   0  5524  2924  1428 S  0.0  2.3  0:00.45 -bash


F1Help  F2Setup F3SearchF4InvertF5Tree  F6SortByF7Nice -F8Nice +F9Kill  F10Quit
```

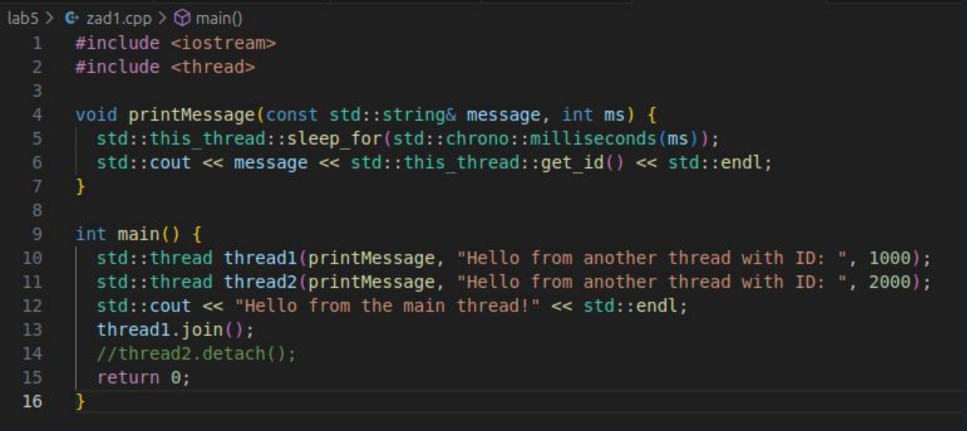
std::thread



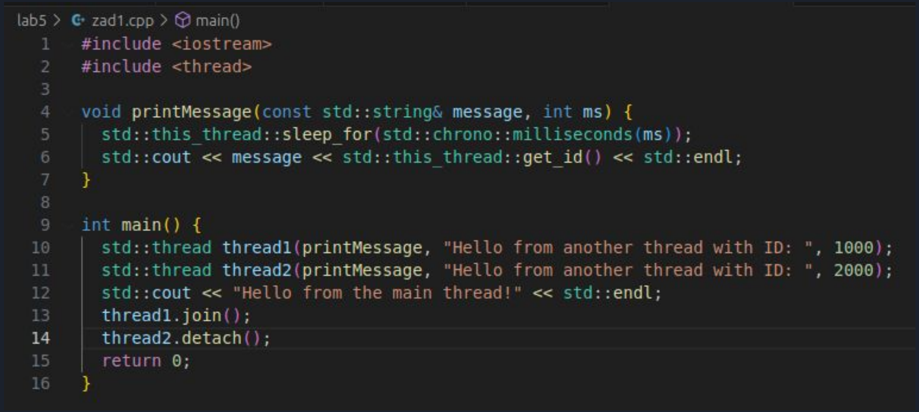
```cpp
#include <iostream>
#include <thread>

void printMessage(const std::string& message) {
    std::this_thread::sleep_for(std::chrono::milliseconds(1000));
    std::cout << message << std::this_thread::get_id() << std::endl;
}

int main() {
    std::thread thread1(printMessage, "Hello from another thread with ID: ");
    std::cout << "Hello from the main thread!" << std::endl;
    thread1.join();

    return 0;
}
```

```
grzetan@grzetan:~/Projects/57209aa7-gr22-repo/lab5$ ./a.out
Hello from the main thread!
Hello from another thread with ID: 130648327059136
grzetan@grzetan:~/Projects/57209aa7-gr22-repo/lab5$ ./a.out
Hello from the main thread!
Hello from another thread with ID: 131475766769344
grzetan@grzetan:~/Projects/57209aa7-gr22-repo/lab5$ ./a.out
Hello from the main thread!
Hello from another thread with ID: 130237232838336
grzetan@grzetan:~/Projects/57209aa7-gr22-repo/lab5$
```

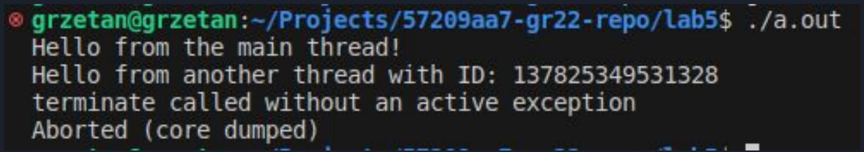
Join vs detach



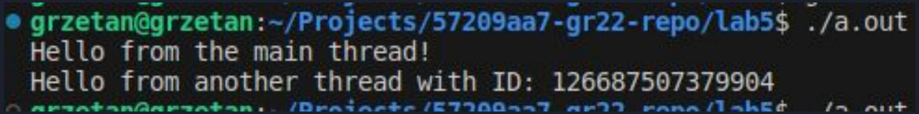
```cpp
#include <iostream>
#include <thread>

void printMessage(const std::string& message, int ms) {
    std::this_thread::sleep_for(std::chrono::milliseconds(ms));
    std::cout << message << std::this_thread::get_id() << std::endl;
}

int main() {
    std::thread thread1(printMessage, "Hello from another thread with ID: ", 1000);
    std::thread thread2(printMessage, "Hello from another thread with ID: ", 2000);
    std::cout << "Hello from the main thread!" << std::endl;
    thread1.join();
    //thread2.detach();
    return 0;
}
```



```cpp
#include <iostream>
#include <thread>

void printMessage(const std::string& message, int ms) {
    std::this_thread::sleep_for(std::chrono::milliseconds(ms));
    std::cout << message << std::this_thread::get_id() << std::endl;
}

int main() {
    std::thread thread1(printMessage, "Hello from another thread with ID: ", 1000);
    std::thread thread2(printMessage, "Hello from another thread with ID: ", 2000);
    std::cout << "Hello from the main thread!" << std::endl;
    thread1.join();
    thread2.detach();
    return 0;
}
```

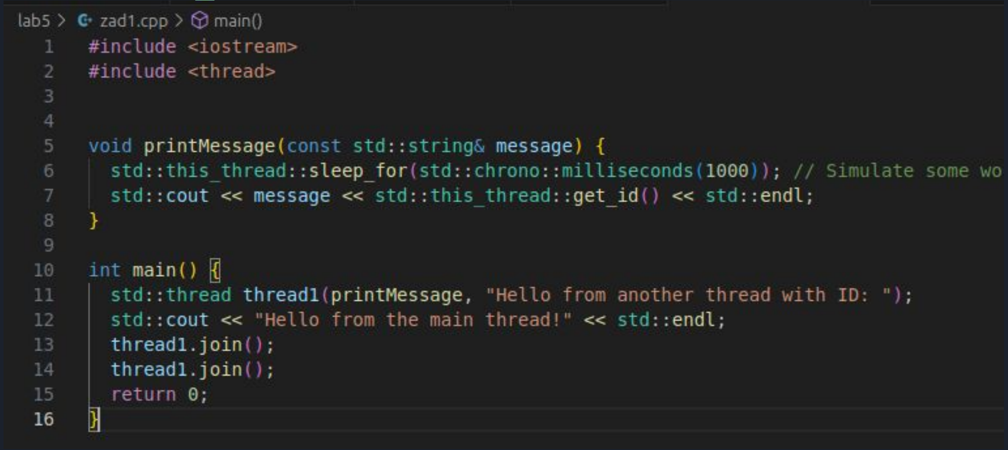


```
grzetan@grzetan:~/Projects/57209aa7-gr22-repo/lab5$ ./a.out
Hello from the main thread!
Hello from another thread with ID: 137825349531328
terminate called without an active exception
Aborted (core dumped)
```



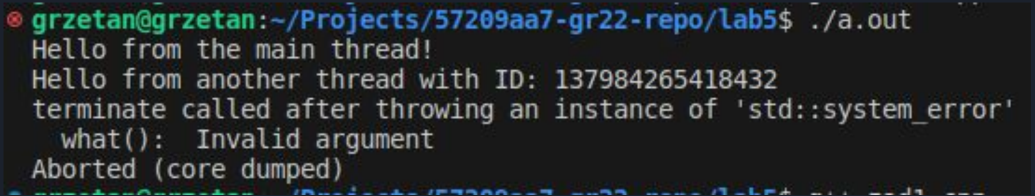
```
grzetan@grzetan:~/Projects/57209aa7-gr22-repo/lab5$ ./a.out
Hello from the main thread!
Hello from another thread with ID: 126687507379904
```

Joinable

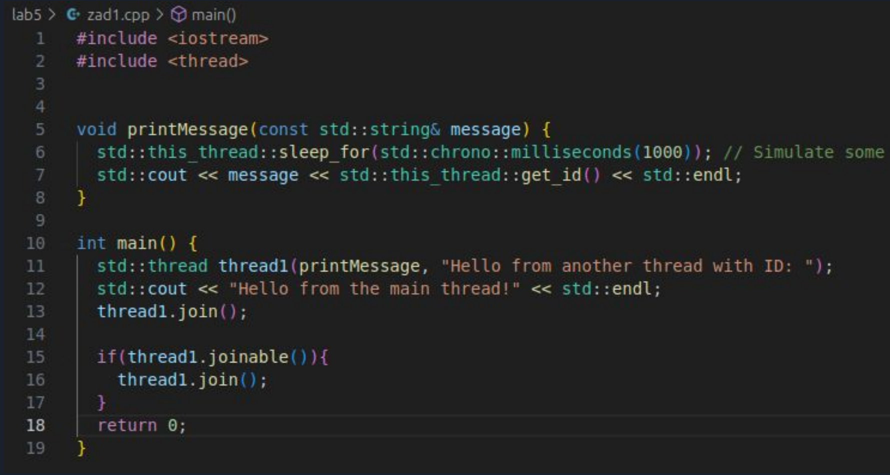


lab5 > zad1.cpp > main()

```cpp
#include <iostream>
#include <thread>


void printMessage(const std::string& message) {
  std::this_thread::sleep_for(std::chrono::milliseconds(1000)); // Simulate some wo
  std::cout << message << std::this_thread::get_id() << std::endl;
}

int main() {
  std::thread thread1(printMessage, "Hello from another thread with ID: ");
  std::cout << "Hello from the main thread!" << std::endl;
  thread1.join();
  thread1.join();
  return 0;
}
```



```
grzetan@grzetan:~/Projects/57209aa7-gr22-repo/lab5$ ./a.out
Hello from the main thread!
Hello from another thread with ID: 137984265418432
terminate called after throwing an instance of 'std::system_error'
  what():  Invalid argument
Aborted (core dumped)
```



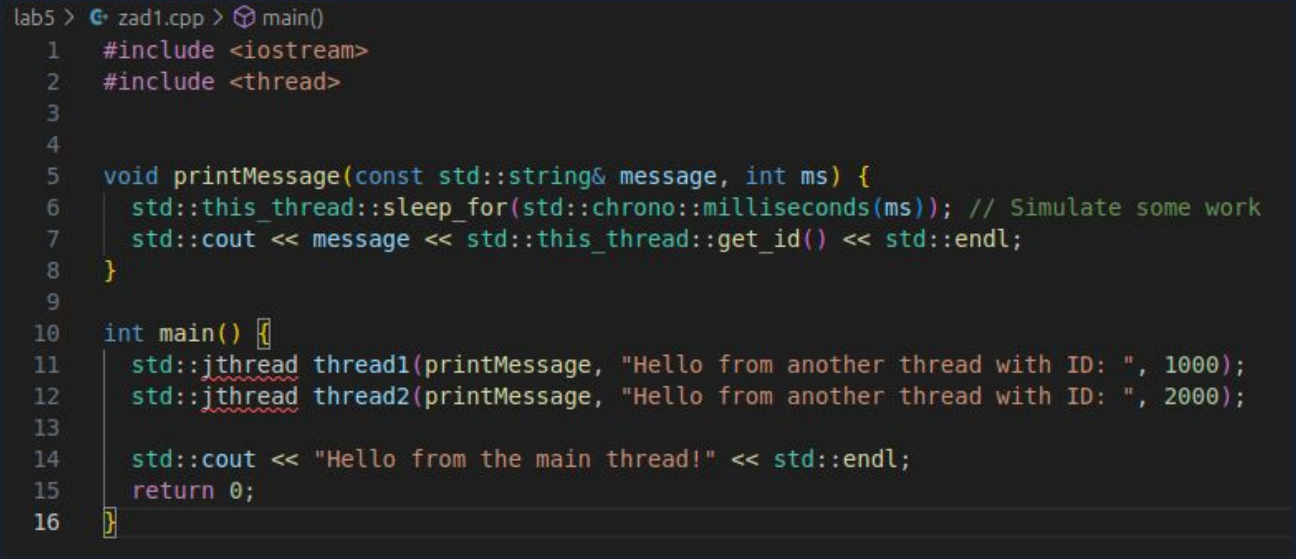
lab5 > zad1.cpp > main()

```cpp
#include <iostream>
#include <thread>


void printMessage(const std::string& message) {
  std::this_thread::sleep_for(std::chrono::milliseconds(1000)); // Simulate some
  std::cout << message << std::this_thread::get_id() << std::endl;
}

int main() {
  std::thread thread1(printMessage, "Hello from another thread with ID: ");
  std::cout << "Hello from the main thread!" << std::endl;
  thread1.join();

  if(thread1.joinable()){
    thread1.join();
  }
  return 0;
}
```

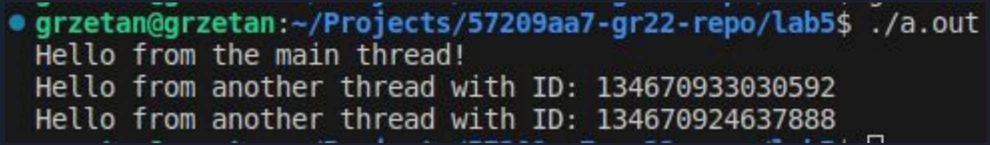
joinable == false gdy:

1. std::thread x;
2. Juz jest join() albo detach()
3. Gdy przeniesiemy wątek poprzez std::move

std::jthread (joining thread)



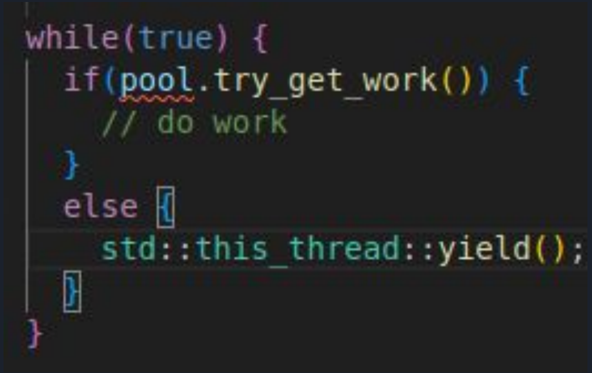
```cpp
#include <iostream>
#include <thread>


void printMessage(const std::string& message, int ms) {
  std::this_thread::sleep_for(std::chrono::milliseconds(ms)); // Simulate some work
  std::cout << message << std::this_thread::get_id() << std::endl;
}

int main() {
  std::jthread thread1(printMessage, "Hello from another thread with ID: ", 1000);
  std::jthread thread2(printMessage, "Hello from another thread with ID: ", 2000);

  std::cout << "Hello from the main thread!" << std::endl;
  return 0;
}
```



```
grzetan@grzetan:~/Projects/57209aa7-gr22-repo/lab5$ ./a.out
Hello from the main thread!
Hello from another thread with ID: 134670933030592
Hello from another thread with ID: 134670924637888
```

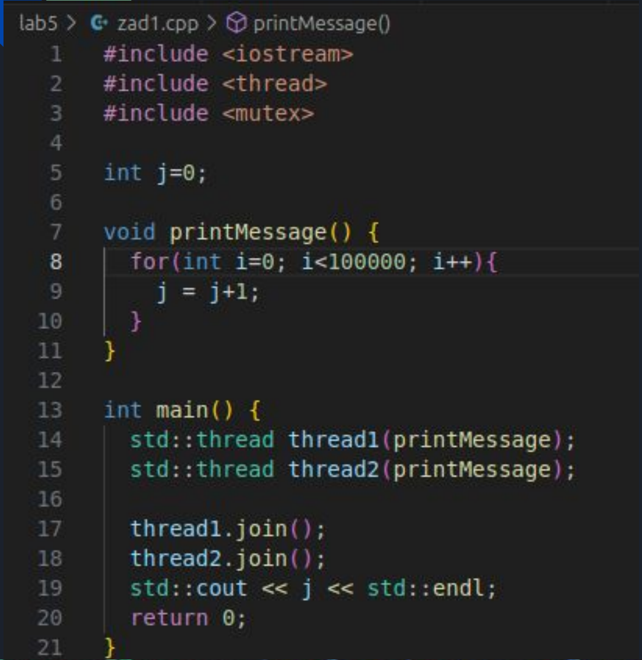
yield

- Nie czeka tylko daje szanse innym wątkom na wykonanie
- Wątek nadal jest responsywny

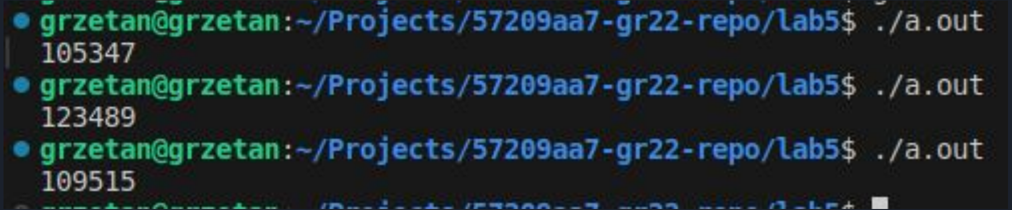


```
while(true) {
  if(pool.try_get_work()) {
    // do work
  }
  else {
    std::this_thread::yield();
  }
}
```

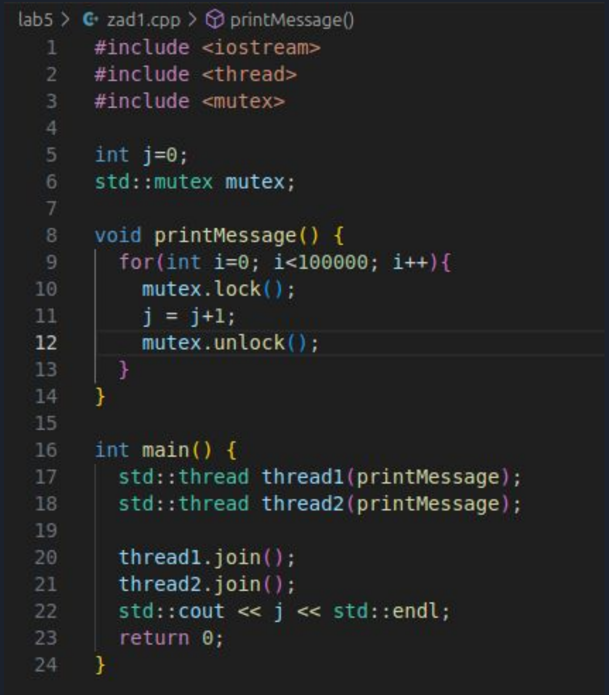
std::mutex



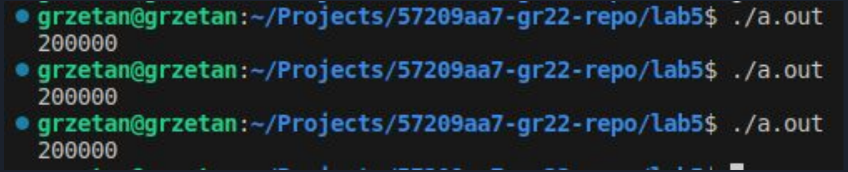
```cpp
#include <iostream>
#include <thread>
#include <mutex>

int j=0;

void printMessage() {
  for(int i=0; i<100000; i++){
    j = j+1;
  }
}

int main() {
  std::thread thread1(printMessage);
  std::thread thread2(printMessage);

  thread1.join();
  thread2.join();
  std::cout << j << std::endl;
  return 0;
}
```



```
grzetan@grzetan:~/Projects/57209aa7-gr22-repo/lab5$ ./a.out
105347
grzetan@grzetan:~/Projects/57209aa7-gr22-repo/lab5$ ./a.out
123489
grzetan@grzetan:~/Projects/57209aa7-gr22-repo/lab5$ ./a.out
109515
```

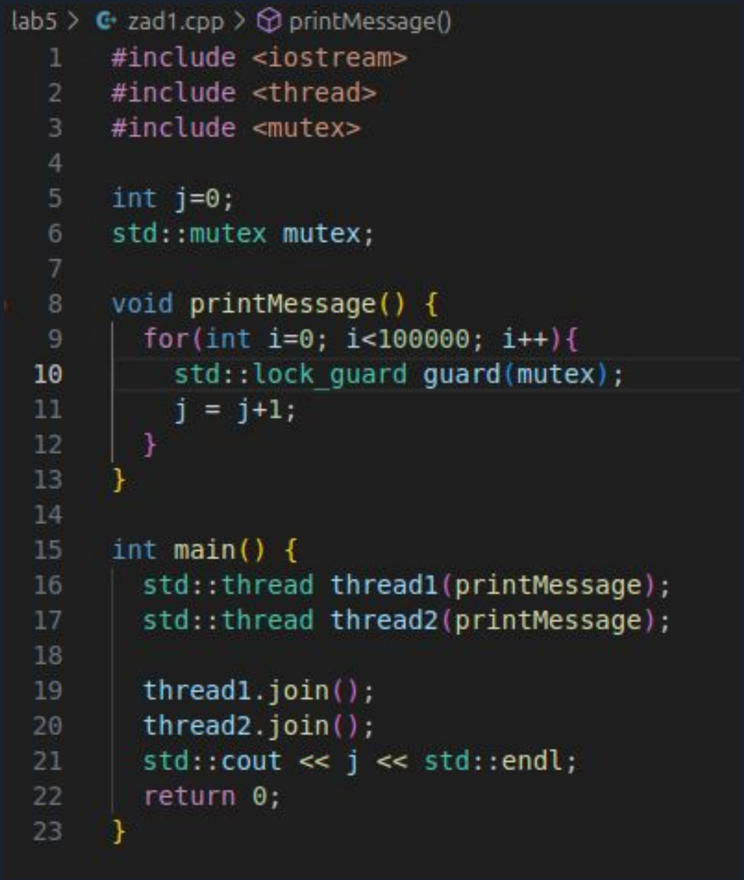


```cpp
#include <iostream>
#include <thread>
#include <mutex>

int j=0;
std::mutex mutex;

void printMessage() {
  for(int i=0; i<100000; i++){
    mutex.lock();
    j = j+1;
    mutex.unlock();
  }
}

int main() {
  std::thread thread1(printMessage);
  std::thread thread2(printMessage);

  thread1.join();
  thread2.join();
  std::cout << j << std::endl;
  return 0;
}
```



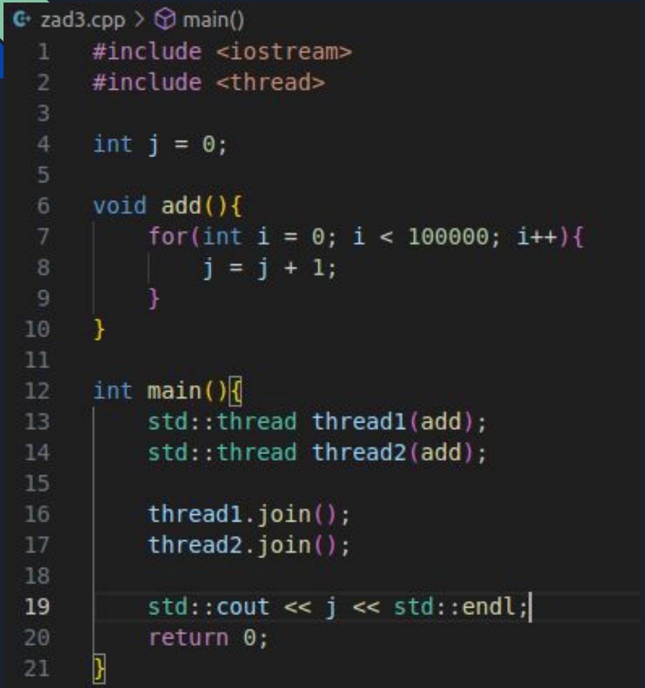
```
grzetan@grzetan:~/Projects/57209aa7-gr22-repo/lab5$ ./a.out
200000
grzetan@grzetan:~/Projects/57209aa7-gr22-repo/lab5$ ./a.out
200000
grzetan@grzetan:~/Projects/57209aa7-gr22-repo/lab5$ ./a.out
200000
```

Locks

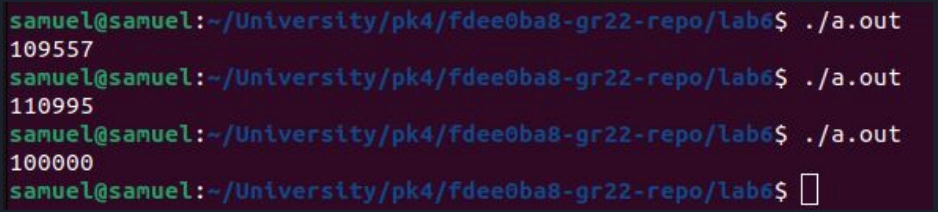


```cpp
#include <iostream>
#include <thread>
#include <mutex>

int j=0;
std::mutex mutex;

void printMessage() {
  for(int i=0; i<100000; i++){
    std::lock_guard guard(mutex);
    j = j+1;
  }
}

int main() {
  std::thread thread1(printMessage);
  std::thread thread2(printMessage);

  thread1.join();
  thread2.join();
  std::cout << j << std::endl;
  return 0;
}
```

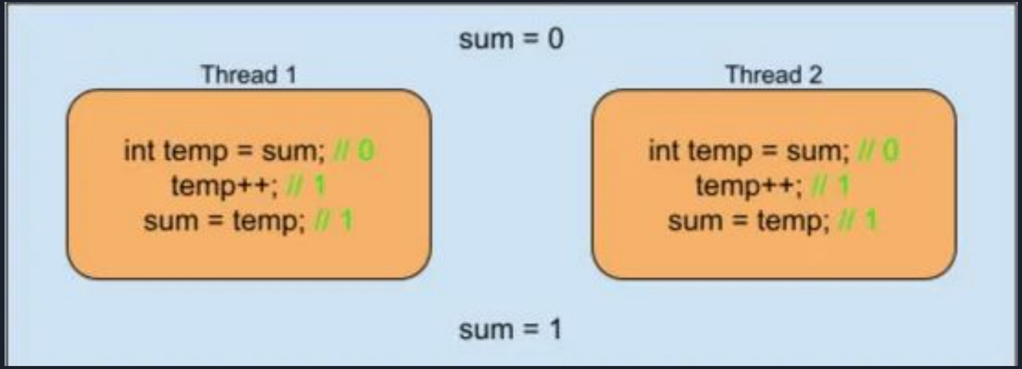
std::atomic



```cpp
#include <iostream>
#include <thread>

int j = 0;

void add(){
    for(int i = 0; i < 100000; i++){
        j = j + 1;
    }
}

int main(){
    std::thread thread1(add);
    std::thread thread2(add);

    thread1.join();
    thread2.join();

    std::cout << j << std::endl;
    return 0;
}
```



```
samuel@samuel:~/University/pk4/fdee0ba8-gr22-repo/lab6$ ./a.out
109557
samuel@samuel:~/University/pk4/fdee0ba8-gr22-repo/lab6$ ./a.out
110995
samuel@samuel:~/University/pk4/fdee0ba8-gr22-repo/lab6$ ./a.out
100000
samuel@samuel:~/University/pk4/fdee0ba8-gr22-repo/lab6$ 
```

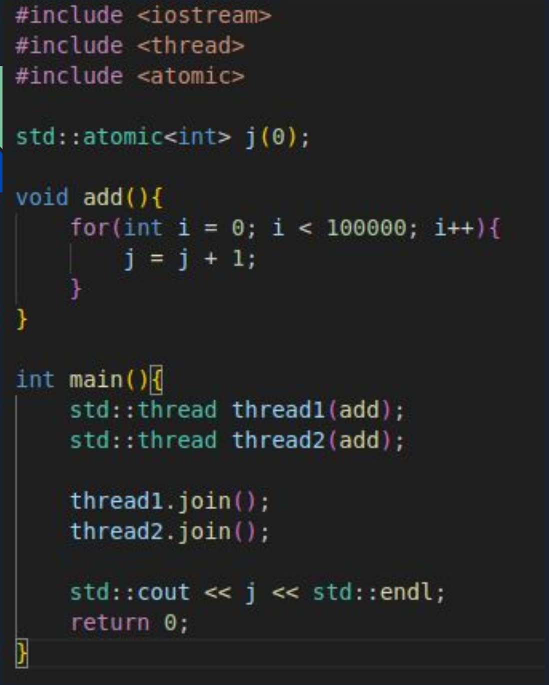


Thread 1

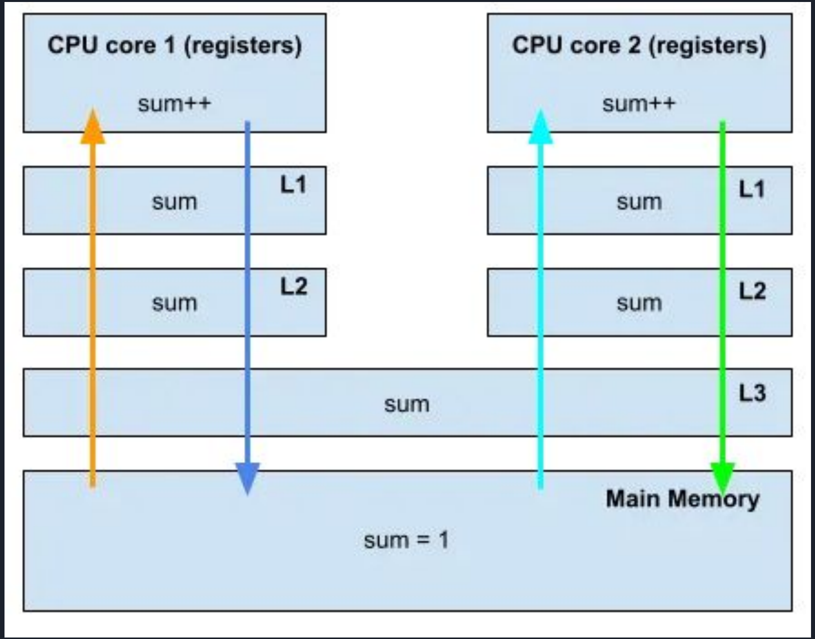
```
int temp = sum; // 0
temp++; // 1
sum = temp; // 1
```

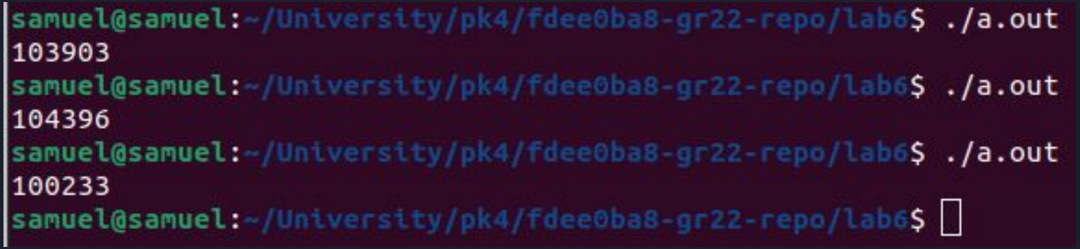
Thread 2

```
int temp = sum; // 0
temp++; // 1
sum = temp; // 1
```

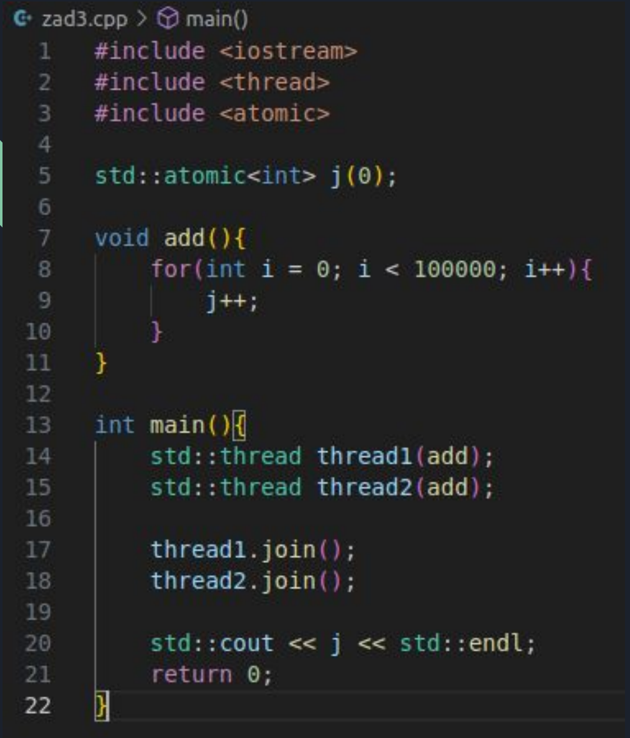


```cpp
#include <iostream>
#include <thread>
#include <atomic>

std::atomic<int> j(0);

void add(){
    for(int i = 0; i < 100000; i++){
        j = j + 1;
    }
}

int main(){
    std::thread thread1(add);
    std::thread thread2(add);

    thread1.join();
    thread2.join();

    std::cout << j << std::endl;
    return 0;
}
```

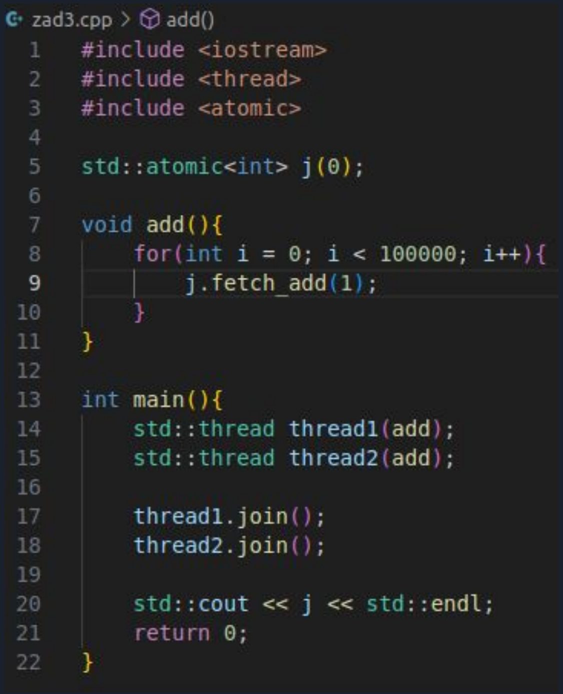




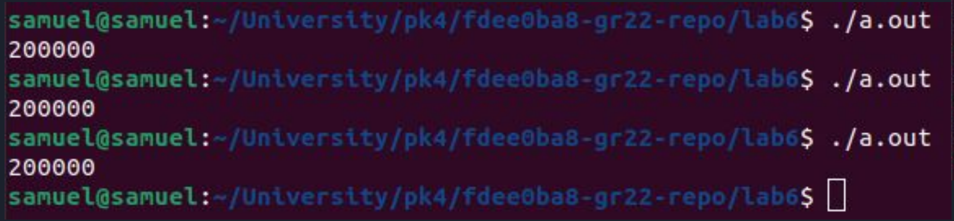
```
samuel@samuel:~/University/pk4/fdee0ba8-gr22-repo/lab6$ ./a.out
103903
samuel@samuel:~/University/pk4/fdee0ba8-gr22-repo/lab6$ ./a.out
104396
samuel@samuel:~/University/pk4/fdee0ba8-gr22-repo/lab6$ ./a.out
100233
samuel@samuel:~/University/pk4/fdee0ba8-gr22-repo/lab6$ 
```



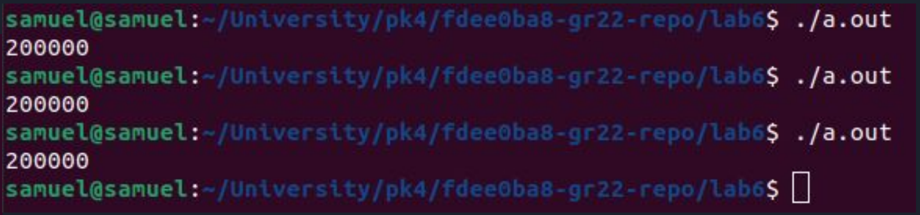
```cpp
#include <iostream>
#include <thread>
#include <atomic>

std::atomic<int> j(0);

void add(){
    for(int i = 0; i < 100000; i++){
        j++;
    }
}

int main(){
    std::thread thread1(add);
    std::thread thread2(add);

    thread1.join();
    thread2.join();

    std::cout << j << std::endl;
    return 0;
}
```



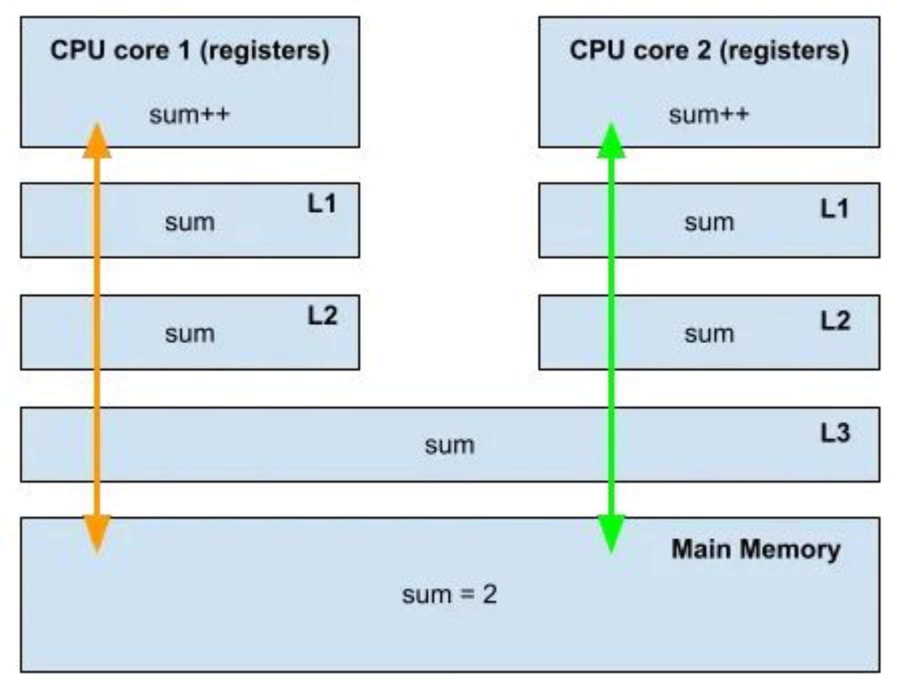
```cpp
#include <iostream>
#include <thread>
#include <atomic>

std::atomic<int> j(0);

void add(){
    for(int i = 0; i < 100000; i++){
        j.fetch_add(1);
    }
}

int main(){
    std::thread thread1(add);
    std::thread thread2(add);

    thread1.join();
    thread2.join();

    std::cout << j << std::endl;
    return 0;
}
```



```
samuel@samuel:~/University/pk4/fdee0ba8-gr22-repo/lab6$ ./a.out
200000
samuel@samuel:~/University/pk4/fdee0ba8-gr22-repo/lab6$ ./a.out
200000
samuel@samuel:~/University/pk4/fdee0ba8-gr22-repo/lab6$ ./a.out
200000
samuel@samuel:~/University/pk4/fdee0ba8-gr22-repo/lab6$
```



```
samuel@samuel:~/University/pk4/fdee0ba8-gr22-repo/lab6$ ./a.out
200000
samuel@samuel:~/University/pk4/fdee0ba8-gr22-repo/lab6$ ./a.out
200000
samuel@samuel:~/University/pk4/fdee0ba8-gr22-repo/lab6$ ./a.out
200000
samuel@samuel:~/University/pk4/fdee0ba8-gr22-repo/lab6$
```



CPU core 1 (registers)

sum++

CPU core 2 (registers)

sum++

L1

sum

L1

sum

L2

sum

L2

sum

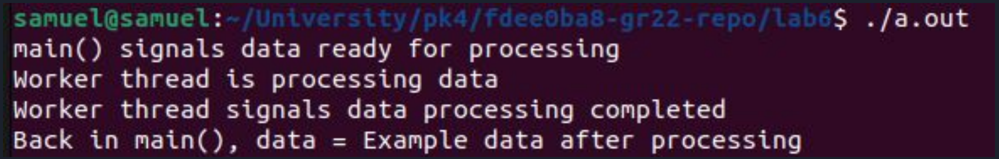
L3

sum

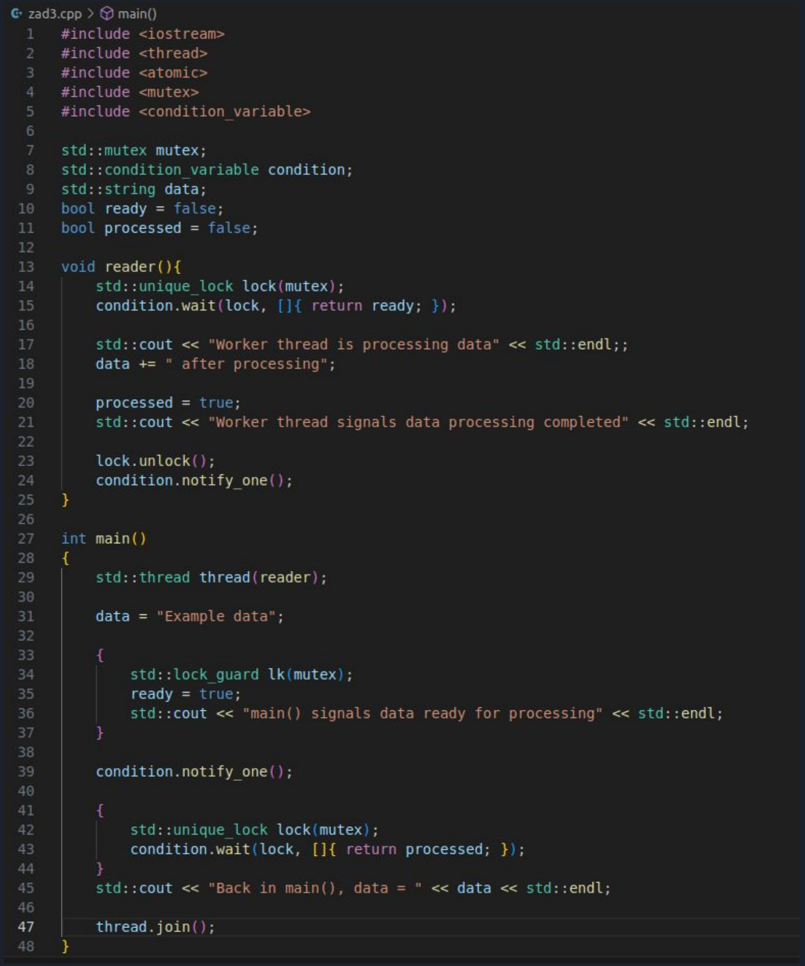
Main Memory

sum = 2

std::condition_variable



```
samuel@samuel:~/University/pk4/fdee0ba8-gr22-repo/lab6$ ./a.out
main() signals data ready for processing
Worker thread is processing data
Worker thread signals data processing completed
Back in main(), data = Example data after processing
```



```cpp
#include <iostream>
#include <thread>
#include <atomic>
#include <mutex>
#include <condition_variable>

std::mutex mutex;
std::condition_variable condition;
std::string data;
bool ready = false;
bool processed = false;

void reader(){
    std::unique_lock lock(mutex);
    condition.wait(lock, []{ return ready; });

    std::cout << "Worker thread is processing data" << std::endl;;
    data += " after processing";

    processed = true;
    std::cout << "Worker thread signals data processing completed" << std::endl;

    lock.unlock();
    condition.notify_one();
}

int main()
{
    std::thread thread(reader);

    data = "Example data";

    {
        std::lock_guard lk(mutex);
        ready = true;
        std::cout << "main() signals data ready for processing" << std::endl;
    }

    condition.notify_one();

    {
        std::unique_lock lock(mutex);
        condition.wait(lock, []{ return processed; });
    }
    std::cout << "Back in main(), data = " << data << std::endl;

    thread.join();
}
```

Zadania

1. Napisz wątek który co sekunde wypisuje zinkrementowaną liczbę aż do zamknięcia programu. Co sie stanie jak uzyjemy join a co jak detach.
2. Napisz program który ma 2 wątki które dodają 1mln razy dowolną liczbę do wektora a po ich zakończeniu dwa kolejne wątki zabierają po 500tyś z wektora. Na koniec wektor ma miec 1mln elementow.
3. Napisz program ktory bedzie wykorzystywal 3 watki, pierwsze 2 inkrementuja zmienna 1mln razy, trzcie czeka az zostanie spelniony warunek a nastepnie dekrementuje zmienna 100tys razy. Do pierwszych dwoch watkow uzyj std::atomic, trzeci watek czeka ma byc synchronizowany z reszta za pomoca std::condition_variable, nastepnie w glownym watku wyswietl zmienna.
4. Napisz program producer (pierwszy watek) - consument (drugi watek) gdzie producer zbiera inputy usera w postaci integerów a consument mnoży se ze sobą. Wątki mają sie komunikować za pomocą kolejki std::vector. Gdy user wpisze “KONIEC”, program powinien wypisać końcowy wynik i sie poprawnie zakończyć. Watek producera ma zbierac inputy i dodawac do kolejki a konsumer ma brac z kolejki i updejtowac wynik

Zrodla

- https://ryonaldteofilo.medium.com/atomics-in-c-what-is-a-std-atomic-and-what-can-be-made-atomic-part-1-a8923de1384d
- https://en.cppreference.com/w/cpp/thread/condition_variable
- https://www.geeksforgeeks.org/multithreading-in-cpp/
- https://www.geeksforgeeks.org/std-mutex-in-cpp/
-