C++ PROGRAMMING

Lecture 13

Secure Software Engineering Group

Philipp Dominik Schubert

Fraunhofer IEM

HEINZ NIXDORF INSTITUT
UNIVERSITÄT PADERBORN

SECURE SOFTWARE ENGINEERING GROUP

CONTENTS

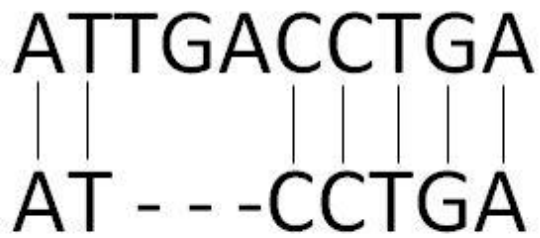
Introduction to the project

- Compare genome sequences to each other
- DNA sequencing machines
  - Decode DNA molecules
  - Produce massive sequence (text) files
  - E.g. ion torrent sequencer
    - Price: ~ $ 50.000

[Figure taken from http://gizmodo.com/5709604/got-50000-you-can-buy-yourself-a-personal-dna-sequencing-machine]

Introduction to the project

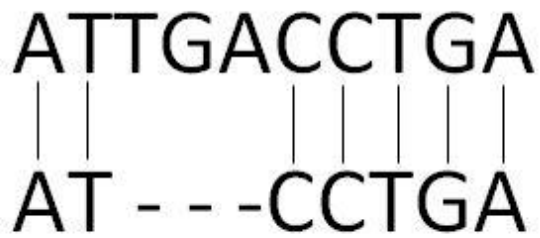
- Sequence alignment
  - ATTGACCTGA
  - ATCCTGA
  - How to find an optimal alignment?
- Smith-Waterman algorithm
  - Find optimal alignment score (similarity)
  - Find optimal alignment (according to the score)



[Figure taken from http://gizmodo.com/5709604/got-50000-you-can-buy-yourself-a-personal-dna-sequencing-machine]

Introduction to the project

- What is an alignment?
  - An alignment is a sequence of operations that transforms one sequence into another one
  - Allowed operations
    - Substitution
    - Copy
    - Deletion
    - Insertion



[Figure taken from http://gizmodo.com/5709604/got-50000-you-can-buy-yourself-a-personal-dna-sequencing-machine]

Input files

- Sequence files are in fasta format
  - .fasta
  - .fas
  - .fa

```
> A fasta example header
ATAAGGTACGACACACT
AGATacacacatgAAAG
AACAGACTTAtattTTT
```

- Sequence files can be huge
  - Reading line by line is usually too slow
  - Read file as one block
  - No need for memory mapped files

- Tasks to solve
  - Read files from disk
  - Remove the header line
  - Remove line breaks `'\n'`
  - Convert to upper case letters

Smith-Waterman algorithm

- Perform algorithm on
  - ACGA
  - TCCG
- Weights
  - $\omega_{mismatch} = \omega_{gap} = -1$
  - $\omega_{match} = 2$
- Create matrix
  - Initialize first row to 0
  - Initialize first column to 0
  - Fill matrix according to recurrence
  - Largest matrix entry is the score
- (Optimal alignment could be reconstructed from matrix)
- We are only interested in the score: Do we really need a matrix? → No!

\	ϵ	A	C	G	A
ϵ	0	0	0	0	0
T	0	0	0	0	0
C	0	0	2	1	0
C	0	0	2	1	0
G	0	0	1	4	3

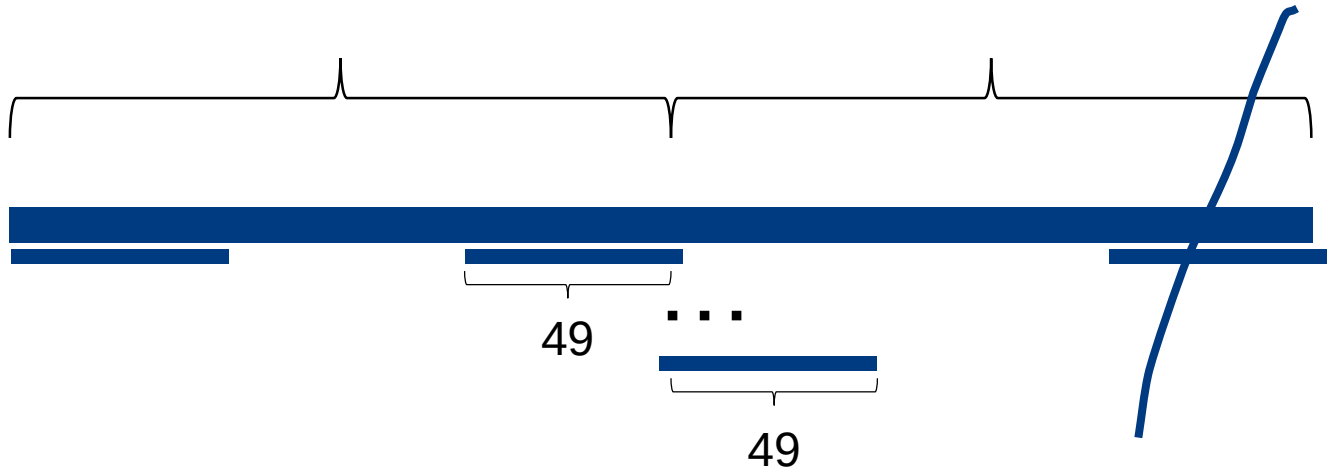
http://rna.informatik.uni-freiburg.de/Teaching/index.jsp?toolName=Smith-Waterman

$$H(i,j) = \max \begin{cases} 0 & \\ H(i-1,j-1) + \omega(a_i, b_j) & \text{match/ mismatch} \\ H(i-1,j) + \omega_{gap} & \text{deletion} \\ H(i,j-1) + \omega_{gap} & \text{insertion} \end{cases}$$

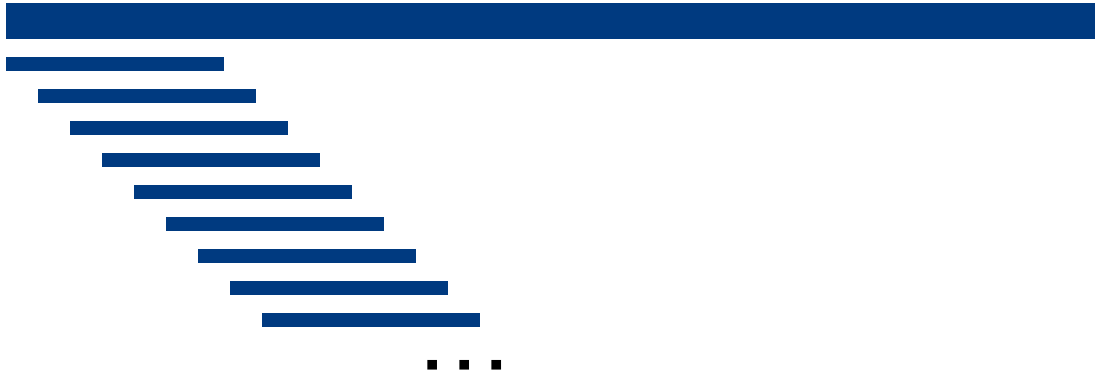
$$\omega(a,b) = \begin{cases} \omega_{match}, & a = b \\ \omega_{mismatch}, & a \neq b \end{cases}$$

Parallelization, calling the Smith-Waterman algorithm multiple times

- Compare each 50 character **segment** of sequence n to each 50 character segment of sequence m using the Smith-Waterman algorithm
- Split into **subtasks**
  - Suppose running two **threads**
  - Split sequence n into two parts
  - One thread compares every segment of first **part** to every segment of second sequence
  - Other thread compares segments of second part to every segment of second sequence
  - Caution at **borders** of parts
  - Caution for thread working on last part
- Both sequences are only read from (no need for synchronization)
  - Make both sequences global variables!



- You may wish to use **at()** rather than **operator[]** to ensure indices are within valid bounds or use Clang's sanitizers or Valgrind



How to model the tasks?

- Model a task as a class
- Provide member variables to capture all required information to solve the task
  - Start of its corresponding part in sequence n
  - End of its corresponding part in sequence n
  - …
- Provide a constructor to correctly initialize members and set up the task
- Implement the call operator to start the actual computations

- Example

```
/* this is not complete */
class SWDTask {
private:
    size_t seq_one_start;
    size_t seq_one_end;
    int smith_waterman_distance(…);

public:
    SWDTask(size_t sos, size_t soe);
    void operator() ();
};
```

Caution: avoid unnecessary copies of `std::string`

- Copying data blocks the processors

```
int smith_waterman_distance(std::string a, std::string b);

for (/* hot loop */) {
  smith_waterman_distance(/* ... */, /* ... */);
}
```

A. Have the sequences as global variables and just pass start and end positions

```
std::string n = /* ... */;
std::string n = /* ... */;

int smith_waterman_distance(int start, int end);

for (/* hot loop */) {
  smith_waterman_distance(/* ... */, /* ... */);
}
```

B. Or use C++ 17 `std::string_view`

- Runtimes may vary from several seconds up to one hour!

Post processing

- For each starting position in one sequence
  - Find the starting position in the other sequence with the highest score
  - Add this highest-score-triple to your post-processed final results

Start in SOX3	Start in SRY	Score
0	0	80
0	1	85
0	2	81
1	10	90
1	15	96
2	4	72

- These are fictional results

Output results

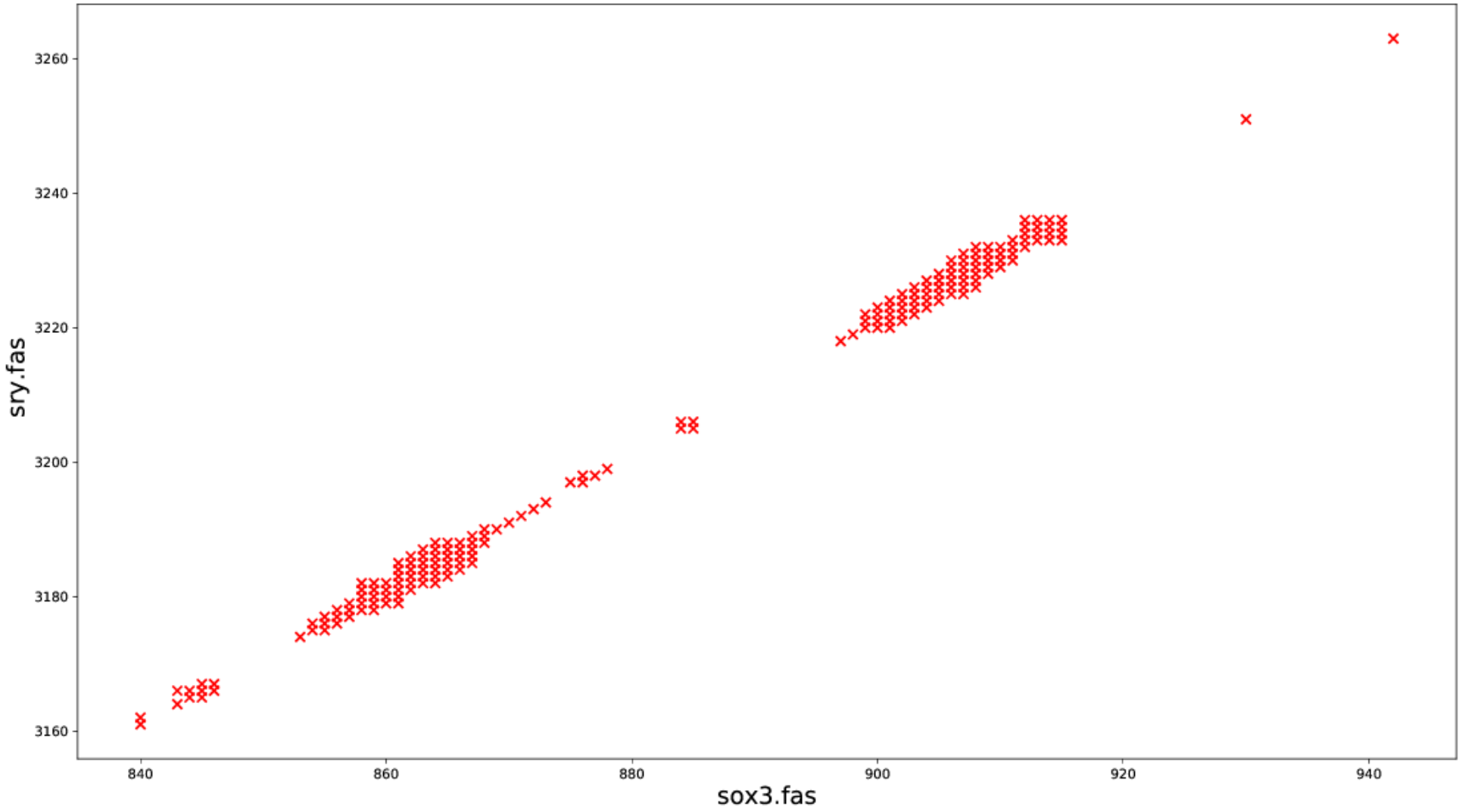
- Write the post-processed results back to a file
- Use a csv (comma separated values) file format

```
SOX3,SRY,Score
10,20,74
14,25,80
123,243,96
214,501,81
```

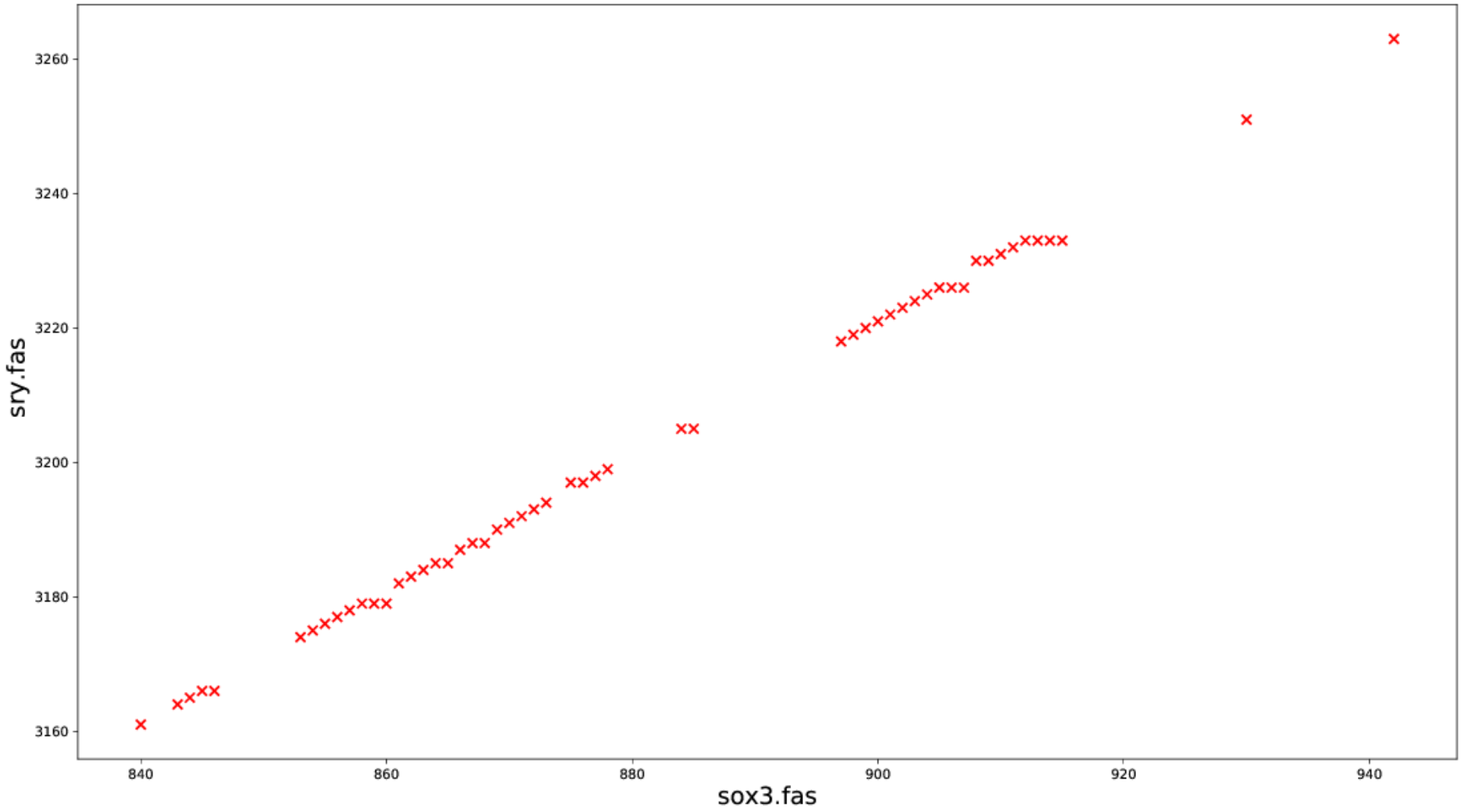
Again, those are fictional results

- Plot the results using the python script
- Or plot the results using a spreadsheet software like Libre Office, Google Sheets or MS Excel
- Hand-in your solution using the PANDA submission system
  - The entire source code, compile command(s) (e.g. Makefile), and plots
  - Include a README with your complete name (first name, middle name, last name), field of study and faculty

Results before preprocessing



Results after preprocessing



Questions about the project?

This is not a group project: plagiarism is prohibited will not be tolerated.

There is still more!

Optimize optimized things

- “Writing fast code”, Andrei Alexandrescu
  - Part I https://www.youtube.com/watch?v=vrfYLlR8X8k
  - Part II https://www.youtube.com/watch?v=9tvbz8CSI8M
- Example

```
size_t count_digits(size_t number){
  size_t digits = 0;
  do {
    ++digits;
    number /= 10;
  } while (number);
  return digits;
}
```

- An (micro-)optimized example

```
size_t count_digits(size_t number){
  size_t digits = 1;
  for (;;) {
    if (number < 10) return digits;
    if (number < 100) return digits + 1;
    if (number < 1000) return digits + 2;
    if (number < 10000) return digits + 3;
    number /= 10000;
    digits += 4;
  }
}
```

- Why is the second version faster?
  - Division is a more expensive operation
  - Comparison and addition is much cheaper

Miscellaneous

- Very incomplete list of names to know
  - Bjarne Stroustup
  - Andrei Alexandrescu
  - Chandler Carruth
  - Sean Parent
  - Herb Sutter
  - Scott Meyers
  - … many more
- C++ on youtube
  - CppCon
  - code::dive
  - … many more

Allocators for container types

- C++ concept – Allocator
  - http://en.cppreference.com/w/cpp/concept/Allocator

```cpp
#include <iostream>
#include <memory>
int main() {
  // usually
  int *i = new int(42);
  int *array = new int[10];
  delete i;
  delete[] array;
  // one level deeper
  std::allocator<int> a;
  int *other = a.allocate(10);
  for (int i = 0; i < 10; ++i)
    other[i] = 2;
  a.deallocate(other, 10);
  return 0;
}
```

Defined in header <vector>

```cpp
template<
    class T,
    class Allocator = std::allocator<T>
> class vector;
```

Defined in header <map>

```cpp
template<
    class Key,
    class T,
    class Compare = std::less<Key>,
    class Allocator = std::allocator<std::pair<const Key, T> >
> class map;
```

- Every STL/BOOST container can be parameterized by an allocator!
- Allocator defines an allocation strategy
  - When to allocate memory?
  - When to deallocate memory?

[Figure taken from http://www.quickmeme.com/img/e7/e7633bedf897bb24ce668ac9c5df6bf88a58ff7e114d27606a756f4c4888a3f1.jpg]

Allocators for container types

- Calls to `new` and `delete` are bottle-necks in HPC
- Calls go to the operating system, everything else has to wait
- Imagine some iterative algorithm

```
matrix a = // some matrix;
matrix b = // some matrix;

// some iterative algorithm
while ( some condition ) {
  matrix c = a * b;
  a = update(a, c);
  b = update(b, c);
}
// use matrix a, b, c
```

```
matrix update(const matrix& m,
              const matrix& n) {
  matrix result(...); // initalize
  for ...
    for ...
      result[][] = m[][]
  return result;
}
```

- Suppose `matrix` allocate its elements on the heap
- `new` and `delete` are called many times!
  - If `operator*` and `update()` are optimized, `new` and `delete` will become a bottle-neck
  - A custom allocator helps with that!

Allocators for container types

- Allocators allow to define your own allocation strategy
- For example (most game consoles do this)
  1. Call `new` only once at program start
     - Allocate everything you need up-front
  2. At runtime your allocator takes care
  3. Call `delete` only once at the end of your program
- BOOST provides some allocator implementations
- Caution
  - Objects allocated with different allocators cannot be used together!
- http://en.cppreference.com/w/cpp/concept/Allocator

- The minimal allocator

```
#include <cstddef>
template <class T>
struct SimpleAllocator {
  typedef T value_type;
  SimpleAllocator(/*ctor args*/);
  template <class U>
  SimpleAllocator(const SimpleAllocator<U> &other);
  T* allocate(std::size_t n);
  void deallocate(T* p, std::size_t n);
};

template <class T, class U>
bool operator==(const SimpleAllocator<T>&,
                const SimpleAllocator<U>&);
template <class T, class U>
bool operator!=(const SimpleAllocator<T>&,
                const SimpleAllocator<U>&);
```

Separate allocation from initialization: `new` and `delete` revisited

- Allocating a type dynamically is a two step process
  - Allocate memory on the heap
  - Initialize the memory using the constructor
- Can we re-use the allocated heap memory?
  - Yes!


```
struct S {
  int x;
  int y;
  S(int x, int y) : x(x), y(y) {}
}

int main() {
  S *s = new S(1, 2);
  s->x = 13;
  s->y = 13;
  delete s;
  return 0;
}
```

Separate allocation from initialization

- http://en.cppreference.com/w/cpp/memory/new/operator_new
- Use (default) **placement new**

```
#include <iostream>
#include <cstdlib>
#include <memory>
struct S {
  int x;
  int y;
  S(int x, int y) : x(x), y(y) {}
  void print() {
    std::cout << "x: " << x
              << ", y: " << y << '\n';
  }
};
```

- You can also define your own operator `new` and `delete`

```
int main() {
  // using the heap
  S *s = new S(1, 2);
  s->x = 13;
  s->print();
  // call dtor but do not free
  s->~S();
  // construct and place in 's'
  S *t = new(s) S(42, 1024);
  t->print();
  // call dtor and free
  delete t;
  // using the stack
  unsigned char buffer[100];
  // construct and place in 'buffer'
  S *u = new(buffer) S(11, 22);
  u->print();
  // is on stack, so call dtor
  u->~S();
  return 0;
}
```

Debug your code: gdb and lldb

- If the code is too complex to be executed in your head …

  let a debugger execute it for you!
- `gdb` GNU debugger
- `lldb` LLVM debugger
- Command-line debugging tools
- What is debugging:
  - Inspect your code and your variables, registers, … by executing it line by line
  - Set break points and halt your program at interesting points
  - Painful (but practical) to use in the command-line
  - Better use it within some IDE like VS Code

How to debug your code?

- Set break-points right before the code of interest
  - Multiple break-points can be set
- ‘Watch’ variables of interest
- Step through the code
- Detect where it goes wrong
- Fix the bug
- Check the fix

Six Stages of Debugging

1. That can’t happen.
2. That doesn’t happen on my machine.
3. That shouldn’t happen.
4. Why does that happen?
5. Oh, I see.
6. How did that ever work

[Figure taken from https://www.linkedin.com/pulse/debug-your-code-easy-way-sanette-tanaka-1]

How to debug your code?

- Compile your code with `-g`

```
-g  Produce debugging information in the operating system's native
        format (stabs, COFF, XCOFF, or DWARF 2).  GDB can work with this
        debugging information.
        […]
        GCC allows you to use -g with -O.  The shortcuts taken by optimized
        code may occasionally produce surprising results: some variables
        you declared may not exist at all; flow of control may briefly move
        where you did not expect it; some statements may not be executed
        because they compute constant results or their values are already
        at hand; some statements may execute in different places because
        they have been moved out of loops.
        Nevertheless it proves possible to debug optimized output.  This
        makes it reasonable to use the optimizer for programs that might
        have bugs.
                    […]
```

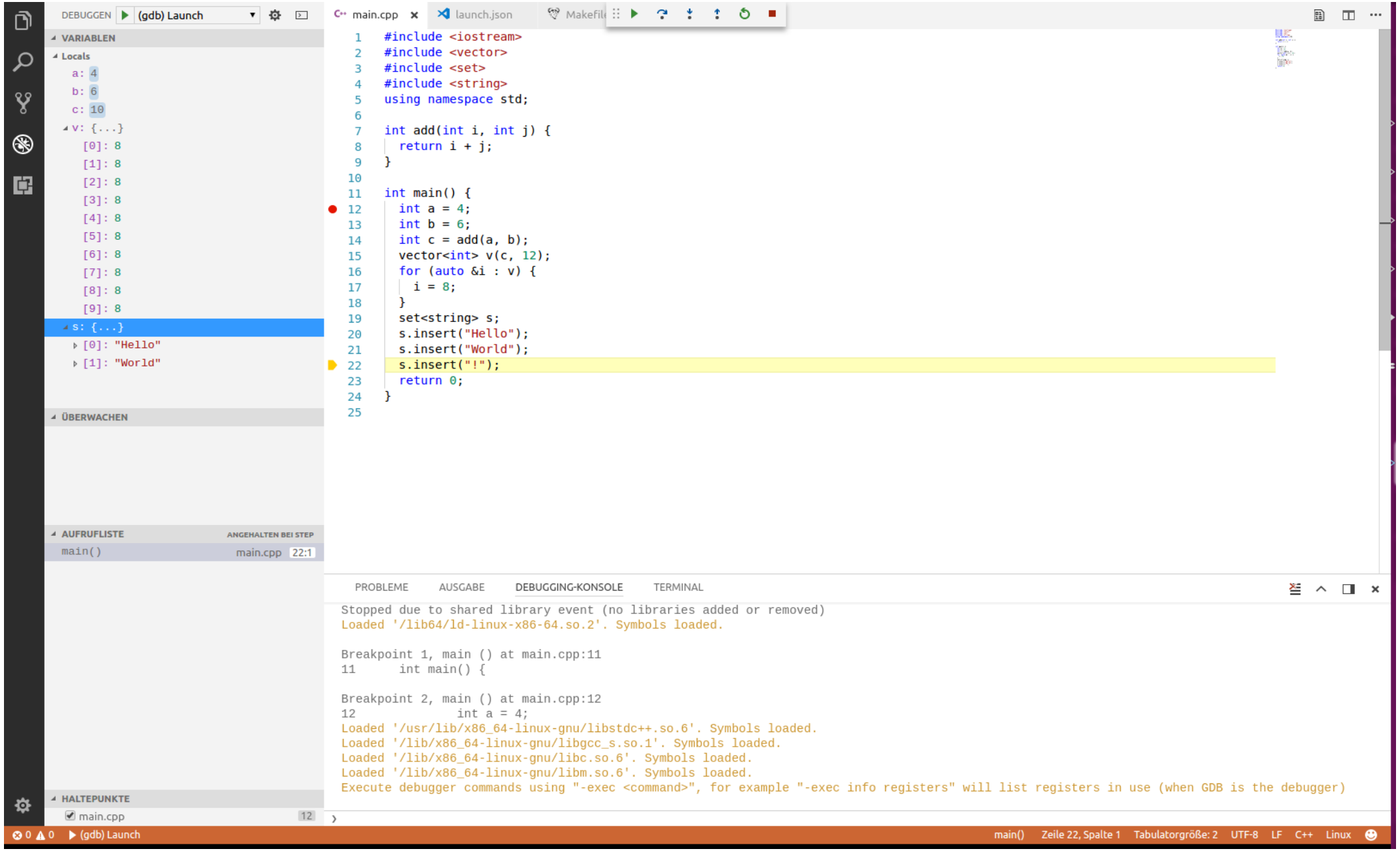
How to debug your code using VS Code?

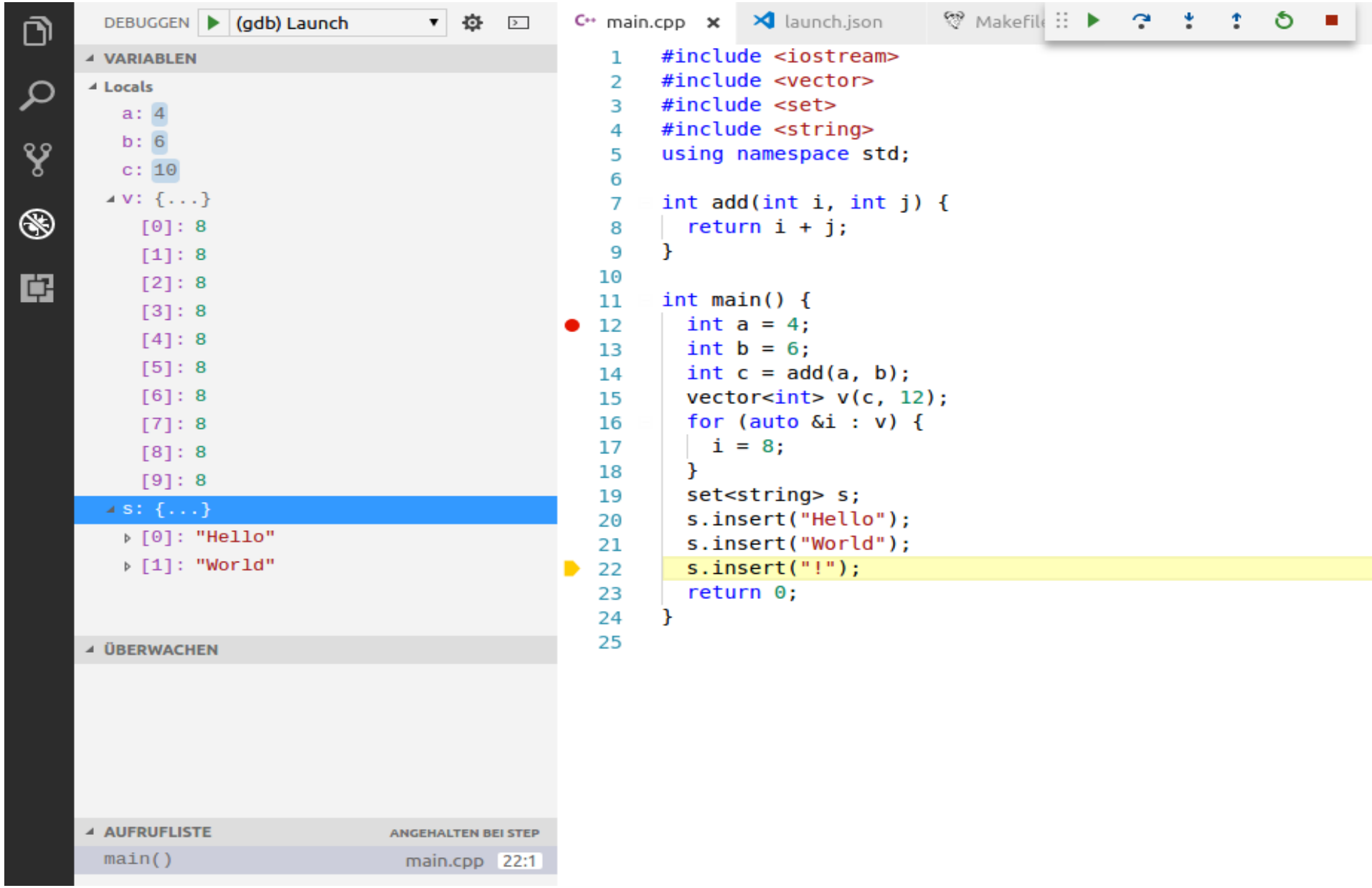
main.cpp | launch.json | Makefile

```
1  all:
2      clang++ -std=c++14 -Wall -Wextra -g -O0 main.cpp -o main
3
4  clean:
5      rm -f main
6
```

main.cpp | launch.json | Makefile

```
{
  // Verwendet IntelliSense zum Ermitteln möglicher Attribute.
  // Zeigen Sie auf vorhandene Attribute, um die zugehörigen Beschreibungen anzuzeigen.
  // Weitere Informationen finden Sie unter https://go.microsoft.com/fwlink/?linkid=830387
  "version": "0.2.0",
  "configurations": [
    {
      "name": "(gdb) Launch",
      "type": "cppdbg",
      "request": "launch",
      "program": "${workspaceFolder}/main",
      "args": [],
      "stopAtEntry": false,
      "cwd": "${workspaceFolder}",
      "environment": [],
      "externalConsole": true,
      "MIMode": "gdb",
      "setupCommands": [
        {
          "description": "Enable pretty-printing for gdb",
          "text": "-enable-pretty-printing",
          "ignoreFailures": true
        }
      ]
    }
  ]
}
```





What next?

- Use C++ in your projects
- Get more experience
- Be curious
- Make mistakes
- Take your time
- C++ is huge
- Reads books, blog articles, programming forums
- Learn the tools used in professional software development
  - Build tools e.g. `make`, `cmake`, …
  - Debuggers e.g. `gdb`, `lldb`
  - Tools from the compiler tool chain e.g. `nm`
  - Version control systems e.g. `git` (https://git.cs.upb.de)

Thank you very much!

- At the end, I hope that you find C++ somewhat useful!

Thank you for your attention
Questions?