

# Lecture 6: Referential Treatment

Bart Iver van Blokland

Forelesningsplan (fremdeles tentativ)				
Uke	Tirsdag 8:15 – 10, R1	Torsdag 14:15 – 16, R1	Fredag 10:15 – 12, R1	Lever
2	Teknisk støtte	Forelesning 1	Teknisk støtte	Øving 0
3	Øvingsforelesning 1	Forelesning 2	Forelesning 3	Øving 1
4	Øvingsforelesning 2	Forelesning 4	Forelesning 5	Øving 2
5	Forelesning 6	Øvingsforelesning 3	Øvingsforelesning 4	Øving 3
6		Forelesning 7	Øvingsforelesning 5	Øving 4
7		Forelesning 8	Øvingsforelesning 6	Øving 5
8		Forelesning 9	Øvingsforelesning 7	Øving 6
9		Forelesning 10	Øvingsforelesning 8	Øving 7
10		Forelesning 11	Øvingsforelesning 9	Øving 8
11		Forelesning 12	Prosjekt	Øving 9
12		Forelesning 13		
13		Forelesning 14		
14	Påskeferie			
15			Eksamenssett (?)	Prosjekt
16	Insperaøving (9-13)	Oppsummering 1/2	Insperaøving (9-13)	Demonstrer prosjekt
17		Oppsummering 2/2 Prosjektene	Eksamenssett	

# Practical stuff

- This is the last week where we use all lecture slots
- From next week and onwards:
  - Thursdays: Ordinær forelesning
  - Fridays: Øvingsforelesning
- Need help?
  - TAs are available monday-friday 8-18 in A3-100+138
  - Need an und.ass or vit.ass? They're available in the same place between 10 and 14
  - Piazza has answers for many common problems
  - println() in inginius: we're looking into it!
  - **Is stuff not working? Please come talk to us!**

# Do you remember?

- What does the `#include` statement do?
- What is the purpose of a header file?
- What is the difference between a declaration and a definition?
- What can cause a multiple definition error to occur?
- What are the main steps for compiling a C++ program?
- What does the `#pragma once` directive do?

# Today

- References
- Data structures
- String encoding
- Naming variables

# Today

- **References**
- Data structures
- String encoding
- Naming variables



Upper D. (1974). The unsuccessful self-treatment of a case of "writer's block". Journal of applied behavior analysis, 7(3), 497. <https://doi.org/10.1901/jaba.1974.7-497a>

# Clarification: assignment

- Assigning the value of one variable to another causes the contained value to be copied

```
std::vector<int> list1 = {1, 2, 3, 4, 5};
```

```
// list1 is assigned here to list2
```

```
// Result: list2 becomes a copy of list1
```

```
std::vector<int> list2 = list1;
```

```
// modifications to list1 are independent of list2
```

```
list1.at(2) = 99;
```

```
std::cout << list2.at(2) << std::endl; // prints: 3
```

# Clarification: assignment

- Function parameters are also implicitly assigned when that function is called.

```
double computeAverage(std::vector<int> numbers) {  
    // numbers now contains a copy of list1  
    return 0;  
}
```

```
int main() {  
    std::vector<int> list1 = {34, 83, 29, 75, 22};  
    computeAverage(list1);  
    return 0;  
}
```



# References

- Reference variable: a variable that does not contain a value itself, but instead modifies the value of another
- Declared by appending an & to the data type

```
int value = 5;  
int& reference = value;  
reference = 10;  
cout << value << endl; // 10
```

↙ The value to which the reference will refer

# Why references?

- Primary use case: function parameters

- Create functions with multiple return values

```
void readSensors(double &temperature, double &humidity);
```

- Create functions which apply modifications on a variable

```
void fillPetrolTank(Car &car);
```

- Avoids creating unnecessary copies

(in my experience: most common reason)

```
void computeAverage(std::vector<int>& bigVectorGoesHere);
```

# Demonstration: references

# Downsides

- It is not possible to create a `std::vector` or `std::array` containing references

```
std::vector<int> references;
```

## the error message

# Course TDT4102 – Lecture 6

[illegible]

# Downsides

- It is not possible to create a `std::vector` or `std::array` containing references
- After a reference has been created, it is no longer possible to change which variable it references

```
int ultimateAnswer = 42;  
int& ref = ultimateAnswer;  
// ref will always reference ultimateAnswer
```

# Const reference

- Create a read-only version of a variable

```
int value = 5;  
const int& reference = value;  
value = 10; // allowed: value is not const  
reference = 10; // not allowed: reference is const
```

- Useful in function parameters: signals that the function does not want to make a copy, but also promises not to modify the value you pass in

```
int computeSum(const std::vector<int>& values);
```

# Const reference

- Any reference to a constant must be declared as const

```
const int value = 5;  
int& reference = value; // Error: binding drops const qualifier  
const int& fixedReference = value; // Ok!
```

# Pass-by-value vs pass-by-(const-)reference

- Common terminology for function parameters
- Pass-by-value
  - Creates a copy of the value passed into the parameter

```
int computeSum(std::vector<int> values);
```

- Pass-by-reference
  - Creates a reference to the variable passed in
  - Referenced value may be modified by the function

```
int computeSum(std::vector<int>& values);
```

- Pass-by-const-reference
  - Creates a const reference to the variable passed in
  - Referenced value may not be modified by the function

```
int computeSum(const std::vector<int>& values);
```



# Oppgave: referanser

- Teori (eksamen mai 2025): Beskriv en situasjon hvor det er ønskelig å bruke "pass-by-const-reference" fremfor "pass-by-reference".
- Teori (eksamen mai 2022): Hvilke fordeler er det med konstante referanser (*const reference*)?
- Koding: Skriv en funksjon som bytter to verdier
  - Bruk:
    - Create New Project > Lectures > Lecture 6 > Task – References
  - Dukker mappen ikke opp?
    - Force refresh content + gjenstart VS Code

# Today

- References
- **Data structures**
- String encoding
- Naming variables

# Demonstration

# Engineering Flowchart

DOES IT MOVE?

NO

YES

SHOULD IT?

SHOULD IT?

NO

YES

NO

YES

NO  
PROBLEM!



NO  
PROBLEM!

# Struct

- Effectively a group of variables

```
struct Point {  
    double x = 0;  
    double y = 0;  
};
```

Can be named anything, but by convention first name is capitalised

Contents are a list of variables called «members».  
Remember to initialise to default values!

Definition ends with a semicolon

# Struct

- Structs become their own data type, which can be used like any other

```
Point point;
```

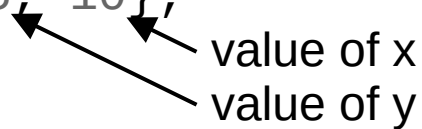
- Read and write variables inside of it using a .

```
point.x = 5;  
std::cout << point.y << std::endl;
```

# Struct

- Values can be initialised using { }
  - The order is the same as the one in which the member variables are defined in the struct

```
Point anotherPoint {5, 10};
```



value of x  
value of y

```
struct Point {  
    double x = 0;  
    double y = 0;  
};
```

- We used this for drawing with AnimationWindow:

```
AnimationWindow window;  
window.draw_line({100, 100}, {200, 200});  
window.wait_for_close();
```



Two instances of type Point!

# Struct

- Best practice: define each struct in its own header file
- Not possible to print a struct using cout directly

```
std::cout << point << std::endl; // error!
```

- Assignment copies all values within the struct

```
Point point {1, 2};  
Point anotherPoint {3, 6};  
point = anotherPoint;  
// point is now (3, 6)
```



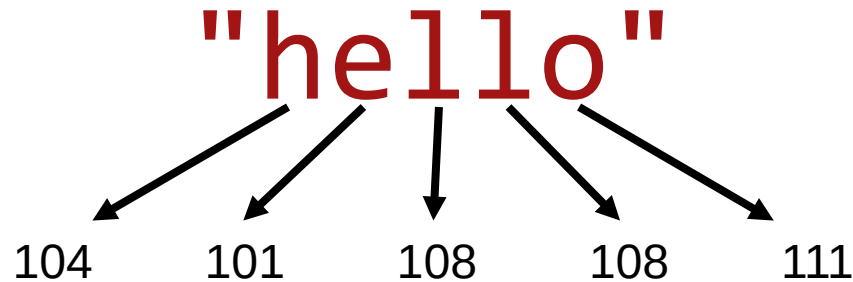
# Struct

- Unlike functions, it is no problem for structs to be defined in multiple .cpp files
- However, **a struct can only be defined once per .cpp file**
  - You should create one header file per struct definition
  - As for all header files, all you need to do is writing `#pragma once` on the first line
- Defining a struct lists its member (containing) variables

# Today

- References
- Data structures
- **String encoding**
- Naming variables

# Computers store text as a sequence of numbers



Each character has a unique number

How numbers are interpreted as characters determines how they are shown on screen.

By default, C++ uses an encoding called «ASCII»

# Characters

- Single characters are stored in the char data type
- Need the number that represents a specific character?  
Just write it in single quotes!

```
char letterA = 'a';  
std::cout << "a is the number " << int(letterA) << std::endl;
```

- Letters are encoded sequentially, so you can calculate letters of the alphabet using by:

```
char letterA = 'a';  
char letterD = 'a' + 3;  
char capitalH = 'A' + 7;
```

# Strings

- A string is more or less a `std::vector<char>`  
(at least for managing its contents)
- You can read individual characters using the `.at()` function

```
std::string message = "G'day mate!";  
char character = message.at(4);  
std::cout << character << std::endl; // Prints y
```

- Or overwrite them:

```
message.at(4) = 'g';  
std::cout << message << std::endl; // Better!  
// Prints: G'dag mate!
```

# Today

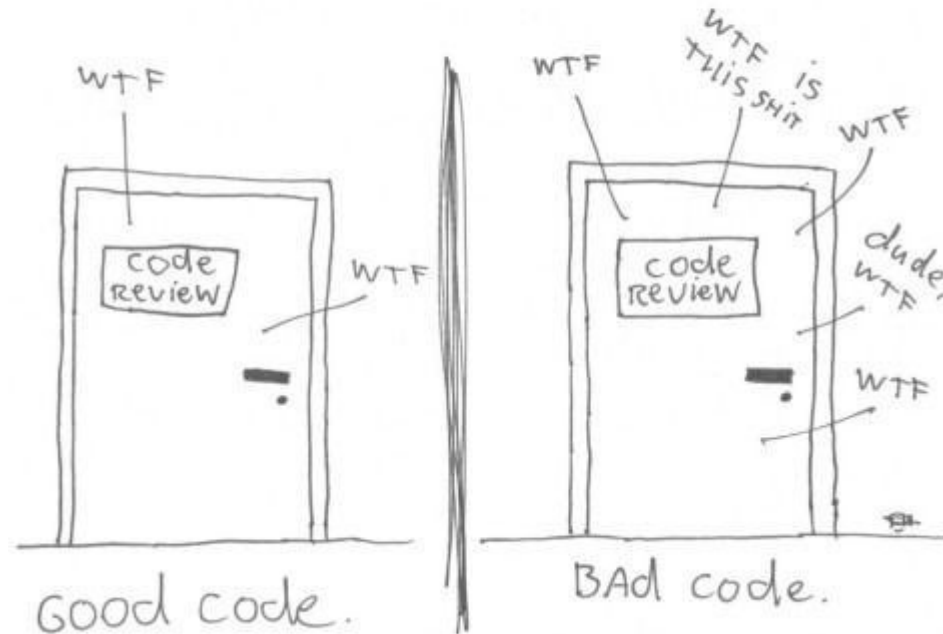
- References
- Data structures
- String encoding
- **Naming variables**

# Our code might work, but how well is it built?



# Code is not always easy to understand

The ONLY valid measurement  
of code quality: WTFs/minute



(c) 2008 Focus Shift/OSNews/Thom Holwerda - <http://www.osnews.com/comics>



What does this function do?

```
int r(int x, int y) {  
    int b = 0;  
    for(int q = x; q < y; q++) {  
        if(q % 2 == 1) {  
            b++;  
        }  
    }  
    return b;  
}
```

What does this function do?

```
int countOddNumbersInRange(int start, int end) {  
    int count = 0;  
    for(int i = start; i < end; i++) {  
        bool numberIsOdd = i % 2 == 1;  
        if(numberIsOdd) {  
            count++;  
        }  
    }  
    return count;  
}
```

Names help figure out what code does!

## Mars Probe Lost Due to Simple Math Error

BY ROBERT LEE HOTZ

OCT. 1, 1999 12 AM PT



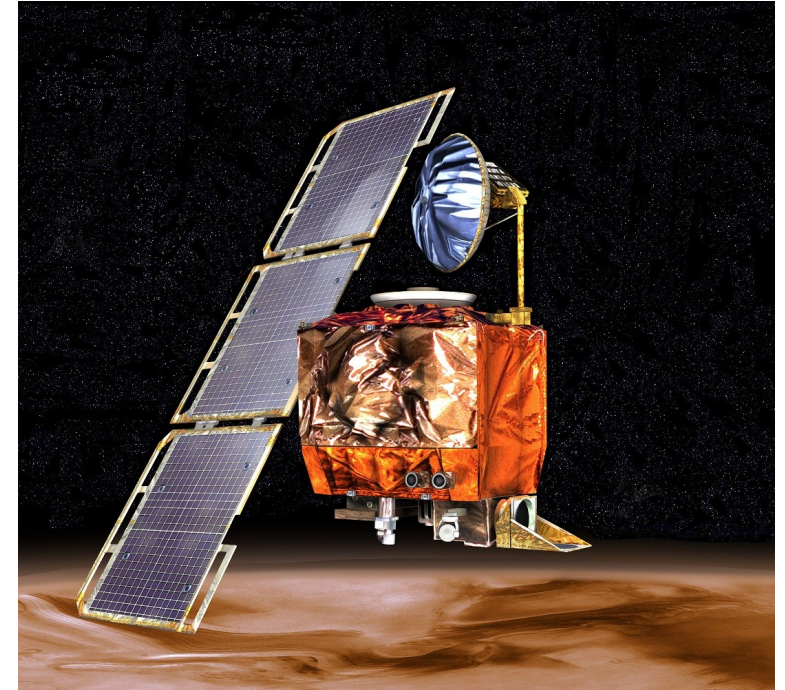
TIMES SCIENCE WRITER

NASA lost its \$125-million Mars Climate Orbiter because spacecraft engineers failed to convert from English to metric measurements when exchanging vital data before the craft was launched, space agency officials said Thursday.

A navigation team at the Jet Propulsion Laboratory used the metric system of millimeters and meters in its calculations, while Lockheed Martin Astronautics in Denver, which designed and built the spacecraft, provided crucial acceleration data in the English system of inches, feet and pounds.

As a result, JPL engineers mistook acceleration readings measured in English units of pound-seconds for a metric measure of force called newton-seconds.

In a sense, the spacecraft was lost in translation.



```
double distance = getDistance();
```

# Names: Things you can do

- Communicate intent

```
int timeInDays;  
int visitedPageCount;  
bool buttonWasClicked;
```

- Names give meaning to values

```
if(distance > 42.195) {  
  
}
```

```
const double marathonDistanceKm = 42.195;  
bool hasCompletedMarathon = distance > marathonDistanceKm;  
if(hasCompletedMarathon) {  
  
}
```

# Names: Things you can do

- Don't hide information

*// hard to see the difference*

```
double dailyTemperatureMeasureIncrementInDegreesCelcius;  
double dailyTemperatureMeasureDecrementInDegreesCelcius;
```

*// possible solution: shorten name*

```
double temperatureIncrement_Celcius;  
double temperatureDecrement_Celcius;
```

*// Or create a type!*

```
struct Temperature {  
    double value = 0;  
    string unit = "Celcius";  
};
```

```
Temperature increment {20, "Celcius"};
```

# Names: Things you can do

- Don't hide information

```
// what can you expect to find in one of these?  
string productInfo;  
string measuredData;  
int value;
```

# Today

- Data structures
- References

# Next week

- Object-oriented programming
- Enumerations
- Namespaces