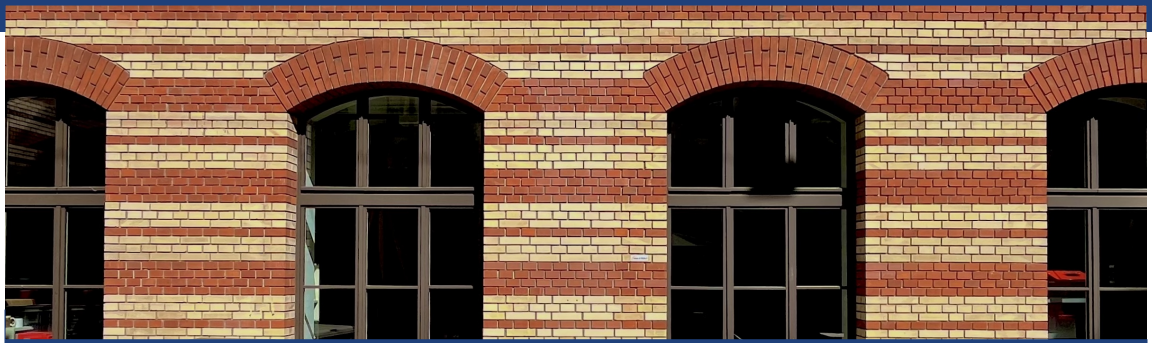

Exercise Session — Computer Science — 06

References, `std::vector`, Multidimensional vectors

Overview

Today's Plan

References

`std::vector<T>`

Multidimensional Vectors

Repetition: Normalized Floating Point Numbers

1. Feedback regarding **code** expert

General things regarding **code** expert

Any questions regarding **code** expert on your part?

2. References

Example of Program Tracing I

```
int a = 3;
int& b = a;

b = 7;

std::cout << a;
```

Output:

Example of Program Tracing I

```
int a = 3;
int& b = a;

b = 7;

std::cout << a;
```

Output: 7

Example of Program Tracing II

```
void foo(int i){
    i = 5;
}

int main(){
    int i = 4;
    foo(i);
    std::cout << i << std::endl;
}
```

Output:

Example of Program Tracing II

```
void foo(int i){
    i = 5;
}

int main(){
    int i = 4;
    foo(i);
    std::cout << i << std::endl;
}
```

Output: 4 ... but why?

Example of Program Tracing II

```
void foo(int i){
    i = 5;
}

int main(){
    int i = 4;
    foo(i);
    std::cout << i << std::endl;
}
```

Output: 4 … but why?

- References (`type&`) are used as type of function parameters (inputs) or return types (returns)
- If the parameters are **not** *referenced*, we say *passed to the function by value*. (This is how we did it for all previous functions); this always makes a copy of the input to the function

Example of Program Tracing III

```
void foo(int& i){
    i = 5;
}

int main(){
    int i = 4;
    foo(i);
    std::cout << i << std::endl;
}
```

Output:

Example of Program Tracing III

```
void foo(int& i){
    i = 5;
}

int main(){
    int i = 4;
    foo(i);
    std::cout << i << std::endl;
}
```

Output: 5

Example of Program Tracing III

```
void foo(int& i){
    i = 5;
}

int main(){
    int i = 4;
    foo(i);
    std::cout << i << std::endl;
}
```

Output: 5

- When a function parameter is a reference type (`type&`), we say "*passed (the argument) by reference*"

References

Why all this?

References

Why all this?

- you can influence several results/variables and don't have to rely on the `return`

References

Why all this?

- you can influence several results/variables and don't have to rely on the `return`
- you can save the (sometimes expensive) copying of parameters and thus improve the performance of the program.

References

Why all this?

- you can influence several results/variables and don't have to rely on the `return`
- you can save the (sometimes expensive) copying of parameters and thus improve the performance of the program.
- sometimes there is no other way (`std::cout` for example, we will have a look in a few weeks)

References as Return Types

We have now seen function parameters that have a reference type, but references can also be used for return types

References as Return Types

We have now seen function parameters that have a reference type, but references can also be used for return types

```
int& increment(int& m){
    return ++m;
}

int main(){
    int n = 3;

    increment(increment(n));

    std::cout << n << std::endl;
}
```

Output:

References as Return Types

We have now seen function parameters that have a reference type, but references can also be used for return types

```
int& increment(int& m){
    return ++m;
}

int main(){
    int n = 3;

    increment(increment(n));

    std::cout << n << std::endl;
}
```

Output: 5, but why?

References as Return Types

We have now seen function parameters that have a reference type, but references can also be used for return types

```
int& increment(int& m){
    return ++m;
}

int main(){
    int n = 3;

    increment(increment(n));

    std::cout << n << std::endl;
}
```

Output: 5, but why? Because of the reference in the return type!

Questions?

Reference or Copy? I

```
int foo(int& a, int b){
    a += b;
    return a;
}

int main(){
    int a = 0;
    int b = 1;
    for(int i = 0; i < 5; ++i){
        b = foo(a, b);
        std::cout << b << " ";
    }
    return 0;
}
```

Output:

Reference or Copy? I

```
int foo(int& a, int b){
    a += b;
    return a;
}

int main(){
    int a = 0;
    int b = 1;
    for(int i = 0; i < 5; ++i){
        b = foo(a, b);
        std::cout << b << " ";
    }
    return 0;
}
```

Output: `1 2 4 8 16`

Reference or Copy? II

```
int foo(int a, int b){
    a += b;
    return a;
}

int main(){
    int a = 0;
    int b = 1;
    for(int i = 0; i < 5; ++i){
        b = foo(a, b);
        std::cout << b << " ";
    }
    return 0;
}
```

Output:

Reference or Copy? II

```
int foo(int a, int b){
    a += b;
    return a;
}

int main(){
    int a = 0;
    int b = 1;
    for(int i = 0; i < 5; ++i){
        b = foo(a, b);
        std::cout << b << " ";
    }
    return 0;
}
```

Output: `1 1 1 1 1`

Reference or Copy? III

```
int foo(int a, int& b){
    a += b;
    return a;
}

int main(){
    int a = 0;
    int b = 1;
    for(int i = 0; i < 5; ++i){
        b = foo(a, b);
        std::cout << b << " ";
    }
    return 0;
}
```

Output:

Reference or Copy? III

```
int foo(int a, int& b){
    a += b;
    return a;
}

int main(){
    int a = 0;
    int b = 1;
    for(int i = 0; i < 5; ++i){
        b = foo(a, b);
        std::cout << b << " ";
    }
    return 0;
}
```

Output: 1 1 1 1 1

Questions?

3. std::vector<T>

How to `std::vector`

- `#include <vector>`

How to `std::vector`

- `#include <vector>`
- Vectors can be thought of as a series of boxes, each storing a value of the given type

How to `std::vector`

- `#include <vector>`
- Vectors can be thought of as a series of boxes, each storing a value of the given type
- You can treat vectors something like a new type

How to `std::vector`

- `#include <vector>`
- Vectors can be thought of as a series of boxes, each storing a value of the given type
- You can treat vectors something like a new type
- There are many ways to initialize/define a vector. Look in the lecture material or search online

How to `std::vector`

- `#include <vector>`
- Vectors can be thought of as a series of boxes, each storing a value of the given type
- You can treat vectors something like a new type
- There are many ways to initialize/define a vector. Look in the lecture material or search online
- `myvector[n-1]`
  to get the nth value of the vector

How to `std::vector`

- `#include <vector>`
- Vectors can be thought of as a series of boxes, each storing a value of the given type
- You can treat vectors something like a new type
- There are many ways to initialize/define a vector. Look in the lecture material or search online
- `myvector[n-1]`
  to get the nth value of the vector
- `myvector.push_back(x)`
  to append the value `x`

Questions?

Exercise: "Reversing Vectors"

Let’s code together!
Code Example "Reversing Vectors" on **code** expert

Exercise: "Reversing Vectors" – Example Solution

```cpp
// POST: Prints a vector in reverse without side-effects
void efficient_reverse_print(std::vector<int>& sequence) {

    for (int i = sequence.size() - 1; i >= 0; i--) {
        std::cout << sequence[i] << " ";
    }

    std::cout << std::endl;

}
```

4. Multidimensional Vectors

What are Multidimensional Vectors?

Multidimensional vectors are matrices[1]

```
matrix.at(row_index)                // Accessing vector<T> (entire row)
matrix.at(row_index).at(col_index)  // Accessing T (single element)
```

[1] they're actually vectors of vectors!

Exercise "Matrix Transpose"

- Open "Matrix Transpose" on **code** expert

Exercise "Matrix Transpose"

- Open "Matrix Transpose" on **code** expert

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{bmatrix}^\top = \begin{bmatrix} 1 & 3 & 5 \\ 2 & 4 & 6 \end{bmatrix}$$

- Think about how you would approach the problem with pen and paper

Exercise "Matrix Transpose"

- Open "Matrix Transpose" on **code** expert

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{bmatrix}^\top = \begin{bmatrix} 1 & 3 & 5 \\ 2 & 4 & 6 \end{bmatrix}$$

- Think about how you would approach the problem with pen and paper
- Simplification of the syntax:

```
using irow = std::vector<int>;
using imatrix = std::vector<irow>;
```

Exercise "Matrix Transpose"

- Open "Matrix Transpose" on **code** expert

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{bmatrix}^\top = \begin{bmatrix} 1 & 3 & 5 \\ 2 & 4 & 6 \end{bmatrix}$$

- Think about how you would approach the problem with pen and paper
- Simplification of the syntax:

```
using irow = std::vector<int>;
using imatrix = std::vector<irow>;
```

- Implement a solution (optionally in groups)

Solution to "Matrix Transpose"

Solution to "Matrix Transpose"

```
imatrix transpose_matrix(const imatrix& matrix) {
  int rows = get_rows(matrix);
  int cols = get_cols(matrix);

  // construct a matrix with zero rows
  imatrix transposed_matrix = imatrix(0);
  for (int col_index = 0; col_index < cols; col_index++) {
    // construct a row with zero entries
    irow row = irow(0);
    for (int row_index = 0; row_index < rows; row_index++) {
      row.push_back(matrix[row_index][col_index]);
    }
    transposed_matrix.push_back(row);
  }
  return transposed_matrix;
}
```

Questions?

5. Repetition: Normalized Floating Point Numbers

Normalized Floating Point Number Systems

Task

- Try to solve following tasks
- Ask if anything remains unclear

Informatik
Exercise Session

Consider the normalized floating point number system $F^*(\beta, p, e_{\min}, e_{\max})$ with $\beta = 2$, $p = 3$, $e_{\min} = -4$, $e_{\max} = 4$.
Compute the following expressions as the parentheses suggest, representing each intermediate result (and the final result) in the normalized floating point system according to the rules of computing with floating point numbers.

$(10 + 0.5) + 0.5$

	decimal		binary
	10		?????
+	0.5		?????
=			?????
+	0.5		?????
=	??	←	?????

$(0.5 + 0.5) + 10$

	decimal		binary
	0.5		?????
+	0.5		?????
=			?????
+	10		?????
=	??	←	?????

$(10 + 0.5) + 0.5$

	decimal		binary
	10		$1.01 \cdot 2^3$
$+$	0.5		$0.0001 \cdot 2^3$
$=$			?????
$+$	0.5		?????
$=$	??	$\leftarrow$	?????

$(0.5 + 0.5) + 10$

	decimal		binary
	0.5		?????
$+$	0.5		?????
$=$			?????
$+$	10		?????
$=$	??	$\leftarrow$	?????

$(10 + 0.5) + 0.5$

	decimal		binary
	10		$1.01 \cdot 2^3$
$+$	0.5		$0.0001 \cdot 2^3$
$=$			$1.0101 \cdot 2^3$
$+$	0.5		?????
$=$	??	$\leftarrow$	?????

$(0.5 + 0.5) + 10$

	decimal		binary
	0.5		?????
$+$	0.5		?????
$=$			?????
$+$	10		?????
$=$	??	$\leftarrow$	?????

$(10 + 0.5) + 0.5$

	decimal		binary
	10		$1.01 \cdot 2^3$
$+$	0.5		$0.0001 \cdot 2^3$
$=$			$1.01 \cdot 2^3$
$+$	0.5		$0.0001 \cdot 2^3$
$=$	??	$\leftarrow$	?????

$(0.5 + 0.5) + 10$

	decimal		binary
	0.5		?????
$+$	0.5		?????
$=$			?????
$+$	10		?????
$=$	??	$\leftarrow$	?????

$(10 + 0.5) + 0.5$

	decimal		binary
	10		$1.01 \cdot 2^3$
$+$	0.5		$0.0001 \cdot 2^3$
$=$			$1.01 \cdot 2^3$
$+$	0.5		$0.0001 \cdot 2^3$
$=$	10	$\leftarrow$	$1.01 \cdot 2^3$

$(0.5 + 0.5) + 10$

	decimal		binary
	0.5		?????
$+$	0.5		?????
$=$			?????
$+$	10		?????
$=$	??	$\leftarrow$	?????

$(10 + 0.5) + 0.5$

	decimal		binary
	10		$1.01 \cdot 2^3$
$+$	0.5		$0.0001 \cdot 2^3$
$=$			$1.01 \cdot 2^3$
$+$	0.5		$0.0001 \cdot 2^3$
$=$	10	$\leftarrow$	$1.01 \cdot 2^3$

$(0.5 + 0.5) + 10$

	decimal		binary
	0.5		$1.00 \cdot 2^{-1}$
$+$	0.5		$1.00 \cdot 2^{-1}$
$=$			?????
$+$	10		?????
$=$	??	$\leftarrow$	?????

$(10 + 0.5) + 0.5$

	decimal		binary
	10		$1.01 \cdot 2^3$
$+$	0.5		$0.0001 \cdot 2^3$
$=$			$1.01 \cdot 2^3$
$+$	0.5		$0.0001 \cdot 2^3$
$=$	10	$\leftarrow$	$1.01 \cdot 2^3$

$(0.5 + 0.5) + 10$

	decimal		binary
	0.5		$1.00 \cdot 2^{-1}$
$+$	0.5		$1.00 \cdot 2^{-1}$
$=$			$1.00 \cdot 2^0$
$+$	10		$1010.00 \cdot 2^0$
$=$	??	$\leftarrow$	?????

$(10 + 0.5) + 0.5$

	decimal		binary
	10		$1.01 \cdot 2^3$
$+$	0.5		$0.0001 \cdot 2^3$
$=$			$1.01 \cdot 2^3$
$+$	0.5		$0.0001 \cdot 2^3$
$=$	10	$\leftarrow$	$1.01 \cdot 2^3$

$(0.5 + 0.5) + 10$

	decimal		binary
	0.5		$1.00 \cdot 2^{-1}$
$+$	0.5		$1.00 \cdot 2^{-1}$
$=$			$1.00 \cdot 2^0$
$+$	10		$1010.00 \cdot 2^0$
$=$	??	$\leftarrow$	$1011.00 \cdot 2^0$

$(10 + 0.5) + 0.5$

	decimal		binary
	10		$1.01 \cdot 2^3$
$+$	0.5		$0.0001 \cdot 2^3$
$=$			$1.01 \cdot 2^3$
$+$	0.5		$0.0001 \cdot 2^3$
$=$	10	$\leftarrow$	$1.01 \cdot 2^3$

$(0.5 + 0.5) + 10$

	decimal		binary
	0.5		$1.00 \cdot 2^{-1}$
$+$	0.5		$1.00 \cdot 2^{-1}$
$=$			$1.00 \cdot 2^0$
$+$	10		$1010.00 \cdot 2^0$
$=$	??	$\leftarrow$	$1.011 \cdot 2^3$

$(10 + 0.5) + 0.5$

	decimal		binary
	10		$1.01 \cdot 2^3$
$+$	0.5		$0.0001 \cdot 2^3$
$=$			$1.01 \cdot 2^3$
$+$	0.5		$0.0001 \cdot 2^3$
$=$	10	$\leftarrow$	$1.01 \cdot 2^3$

$(0.5 + 0.5) + 10$

	decimal		binary
	0.5		$1.00 \cdot 2^{-1}$
$+$	0.5		$1.00 \cdot 2^{-1}$
$=$			$1.00 \cdot 2^0$
$+$	10		$1010.00 \cdot 2^0$
$=$	12	$\leftarrow$	$1.10 \cdot 2^3$

Questions?

6. Outro

General Questions?

See you next time

Have a nice week!