ETH zürich

Übungsstunde W07

Informatik (RW & CBBI & Statistik) – HS 23

Übersicht

Heutiges Programm

Feedback zu **code** expert
Ziele
Referenzen
`std::vector<T>`
(ASCII) Symbole
Feedback
Repetition: Floating Point Numbers
Outro

`rwko.ch/lily`

1. Feedback zu **code** expert

Allgemeines zu **code** expert

- Ihr löst die Aufgaben wirklich sehr gut! Weiter so :)
- Zu E5T1: 0.11 in $F^*(2, 4, -3, 3)$

Fragen/Unklarheiten?

2. Ziele

Ziele

- ☐ Programme, die Referenzen beinhalten, tracen können
- ☐ Programme schreiben können, die Vektoren erstellen, modifizieren und darüber iterieren
- ☐ Programme, die ASCII-Zeichen modifizieren, tracen und schreiben können

3. Referenzen

Beispiel zu Program Tracing I

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

b = 2;

std::cout << a;
```

Output:

Beispiel zu Program Tracing I

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

b = 2;

std::cout << a;
```

Output: 2

Beispiel zu Program Tracing II

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

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

Output:

Beispiel zu Program Tracing II

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

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

Output: 4 …aber wieso?

Beispiel zu Program Tracing II

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

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

Output: 4 …aber wieso? Referenzen (`type&`) werden als Typ von Funktionsparametern (Inputs) oder Rückgabetypen (Returns) verwendet. Wenn die Parameter **nicht** *referenced* sind, so sagt man *passed to the function by value* (So haben wir das bei allen bisherigen Funktionen gemacht). Dabei wird immer eine Kopie des Inputs für die Funktion angefertigt.

Beispiel zu Program Tracing III

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

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

Output:

Beispiel zu Program Tracing III

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

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

Output: 5

Beispiel zu Program Tracing III

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

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

Output: 5 Wenn ein Funktionsparameter ein Referenztyp (`type&`) ist, sagt man “*passed (the argument) by reference*”

Übungen

Wieso das Ganze?

Übungen

Wieso das Ganze?

- da man so mehrere Resultate/Variablen beeinflussen kann und nicht nur auf das `return` angewiesen ist

Übungen

Wieso das Ganze?

- da man so mehrere Resultate/Variablen beeinflussen kann und nicht nur auf das `return` angewiesen ist
- da man sich so das (teils teure) Kopieren der Parameter spart und somit die Performance des Programms verbessern kann

Übungen

Wieso das Ganze?

- da man so mehrere Resultate/Variablen beeinflussen kann und nicht nur auf das `return` angewiesen ist
- da man sich so das (teils teure) Kopieren der Parameter spart und somit die Performance des Programms verbessern kann
- da es manchmal einfach nicht anders geht (std::cout zum Beispiel werden wir uns in paar Wochen anschauen)

Fragen/Unklarheiten?

Referenzen als Return Types

Wir haben jetzt Funktionsparameter gesehen, die einen Referenztypen haben, aber auch für return types lassen sich Referenzen verwenden

Referenzen als Return Types

Wir haben jetzt Funktionsparameter gesehen, die einen Referenztypen haben, aber auch für return types lassen sich Referenzen verwenden

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

int main(){
    int n = 3;

    increment(increment(n));

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

Output:

Referenzen als Return Types

Wir haben jetzt Funktionsparameter gesehen, die einen Referenztypen haben, aber auch für return types lassen sich Referenzen verwenden

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

int main(){
    int n = 3;

    increment(increment(n));

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

Output: 5, aber wieso?

Referenzen als Return Types

Wir haben jetzt Funktionsparameter gesehen, die einen Referenztypen haben, aber auch für return types lassen sich Referenzen verwenden

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

int main(){
    int n = 3;

    increment(increment(n));

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

Output: 5, aber wieso? Wegen der Referenzen!

Fragen/Unklarheiten?

Referenz oder Kopie?

Aufgabe

- 3 Gruppen bilden
- Jede Gruppe erhält ein Code Snippet; versucht herauszufinden, was der Code macht
- Ruft micht, sobald ihr eine Lösung habt
- Erklärt den anderen euer Resultat

Referenz oder Kopie? 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:

Referenz oder Kopie? 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`

Referenz oder Kopie? 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:

Referenz oder Kopie? 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`

Referenz oder Kopie? 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:

Referenz oder Kopie? 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`

Fragen/Unklarheiten?

4. std::vector<T>

`std::vector`, aber wie?

- `#include <vector>`

`std::vector`, aber wie?

- `#include <vector>`
- Vektoren kann man sich als eine Reihe von Kästchen vorstellen, die je einen Wert des gegebenen Typs speichern

`std::vector`, aber wie?

- `#include <vector>`
- Vektoren kann man sich als eine Reihe von Kästchen vorstellen, die je einen Wert des gegebenen Typs speichern
- Man kann Vektoren etwa so behandeln wie einen neuen Typen

`std::vector`, aber wie?

- `#include <vector>`
- Vektoren kann man sich als eine Reihe von Kästchen vorstellen, die je einen Wert des gegebenen Typs speichern
- Man kann Vektoren etwa so behandeln wie einen neuen Typen
- `std::vector<int> myvector{1,2,3};`
  um einen Vektoren zu initialisieren

`std::vector`, aber wie?

- `#include <vector>`
- Vektoren kann man sich als eine Reihe von Kästchen vorstellen, die je einen Wert des gegebenen Typs speichern
- Man kann Vektoren etwa so behandeln wie einen neuen Typen
- `std::vector<int> myvector{1,2,3};`
  um einen Vektoren zu initialisieren
- Es gibt viele Möglichkeiten, einen Vector zu initialisieren/definieren. Schaut in den Summaries nach oder sucht online

`std::vector`, aber wie?

- `#include <vector>`
- Vektoren kann man sich als eine Reihe von Kästchen vorstellen, die je einen Wert des gegebenen Typs speichern
- Man kann Vektoren etwa so behandeln wie einen neuen Typen
- `std::vector<int> myvector{1,2,3};`
  um einen Vektoren zu initialisieren
- Es gibt viele Möglichkeiten, einen Vector zu initialisieren/definieren. Schaut in den Summaries nach oder sucht online
- `myvector.at(n-1)`
  um den n-ten Wert im Vektor `myvector` zu benutzen

`std::vector`, aber wie?

- `#include <vector>`
- Vektoren kann man sich als eine Reihe von Kästchen vorstellen, die je einen Wert des gegebenen Typs speichern
- Man kann Vektoren etwa so behandeln wie einen neuen Typen
- `std::vector<int> myvector{1,2,3};`
  um einen Vektoren zu initialisieren
- Es gibt viele Möglichkeiten, einen Vector zu initialisieren/definieren. Schaut in den Summaries nach oder sucht online
- `myvector.at(n-1)`
  um den n-ten Wert im Vektor `myvector` zu benutzen
- `myvector.push_back(x)`
  um den Wert `x` anzuhängen

Fragen/Unklarheiten?

5. (ASCII) Symbole

Übung "Converting Input to UPPER CASE"

Aufgabe

Schreibt ein Programm, das eine durch ein Zeilenumbruchszeichen begrenzte Zeichenfolge als Vektor von `char` einliest. Anschließend soll das Programm die Folge ausgeben, wobei alle Kleinbuchstaben in GROSSBUCHSTABEN umgewandelt werden.
Um die Sequenz zu lesen:

- lies ein Zeichen aus dem Standardinput
- speichere das Zeichen in einem Vektor aus Zeichen
- wiederholen, bis ein neues Zeilenumbruchszeichen (`/n`) gelesen wird..

Fügt den Code, der die gesamte Sequenz in Grossbuchstaben und ein einzelnes Zeichen in Grossbuchstaben umwandelt, in separate Funktionen ein (Ihr solltet mindestens drei Funktionen haben).
Hinweis: Variablen vom Typ `char` können wie Zahlen behandelt werden.

Übung "Converting Input to UPPER CASE"

Aufgabe

1. Überlegt, wie ihr die Aufgabe "Converting Input to UPPER CASE" auf **code** expert am besten angeht

Übung "Converting Input to UPPER CASE"

Aufgabe

1. Überlegt, wie ihr die Aufgabe "Converting Input to UPPER CASE" auf **code** expert am besten angeht
2. Programmiert (optional gruppenweise) eine Lösung

(Lösung) "Converting Input to UPPER CASE"

```
#include <iostream>
#include <vector>
#include <ios>
```

(Lösung) "Converting Input to UPPER CASE"

```
// POST: Converts the letter to upper case.
void char_to_upper(char& letter){
    if('a' <= letter && letter <= 'z'){
        letter -= 'a' - 'A';  // 'a' > 'A'
    }
}

// POST: Converts all letters to upper-case.
void to_upper(std::vector<char>& letters){
    for(unsigned int i = 0; i < letters.size(); ++i){
        char_to_upper(letters.at(i));
    }
}
```

Lösung "Converting Input to UPPER CASE"

```
std::cin >> std::noskipws;
std::vector<char> letters;
char ch;

// Step 1: Read input.
do{
    std::cin >> ch;
    letters.push_back(ch);
}while(ch != '\n');

// Step 2: Convert to upper-case.
to_upper(letters);

// Step 3: Output.
for(unsigned int i = 0; i < letters.size(); ++i){
    std::cout << letters.at(i);
}
```

Fragen/Unklarheiten?

6. Feedback

Euer Feedback an mich

Feedback-formular

(Nehmt euch Zeit und seid ehrlich)

7. Repetition: Floating Point Numbers

Normalized Floating Point Number Systems

Aufgabe

- Versucht (in der Gruppe), folgende Aufgaben zu lösen
- Fragt, wenn etwas unklar ist

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	$\leftarrow$	$1.10 \cdot 2^3$

Fragen/Unklarheiten?

8. Outro

Allgemeine Fragen?

Bis zum nächsten Mal

Schöne Woche!