



Übungsstunde W08

Informatik (RW & CBB & Statistik) – HS 23

Heutiges Programm

Follow-up

Ziele

Multidimensionale Vektoren

Rekursion

Outro



rwko.ch/lily

1. Follow-up

Follow-up

- Euer Feedback
- Falls ihr mir sonst etwas mitteilen möchtet, schreibt mir gerne eine Mail oder nutzt das Feedback Form auf Notion
- Bonusaufgabe 1: Ihr müsst falschen Input nicht beachten

2. Ziele

Ziele

- Programme schreiben können, die multidimensionale Vektoren beinhalten
- Programme mit Rekursion verstehen und schreiben können

3. Multidimensionale Vektoren

Was sind Multidimensionale Vektoren?

Multidimensionale Vektoren sind Matrizen ¹

```
matrix.at(row)           //Reihenzugriff -> Vektor  
matrix.at(row).at(col)  //Elementzugriff -> Element
```

¹eigentlich sind sie Vektoren von Vektoren!

Aufgabe "Matrix Transpose"

- Öffnet "Matrix Transpose" auf **code expert**

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

- Überlegt euch, wie ihr das Problem mit Stift und Papier angehen würdet

Aufgabe "Matrix Transpose"

- Öffnet "Matrix Transpose" auf **code expert**

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

- Überlegt euch, wie ihr das Problem mit Stift und Papier angehen würdet
- Vereinfachung der Syntax:

```
using irow = std::vector<int>;  
using imatrix = std::vector<irow>;
```
- Programmiert eine Lösung (optional in Gruppen)

Lösung zu "Matrix Transpose"

Lösung zu "Matrix Transpose"

```
imatrix transpose_matrix(const imatrix &matrix){
    unsigned int r, c;
    r = get_rows(matrix);           // number of rows
    c = get_cols(matrix);           // number of columns
    imatrix transposed_matrix;      // init' transp. matrix
    for(unsigned int col_i = 0; col_i < c; col_i++){
        irow row;                   // init' transp. row
        // entry-wise add transp. row to transp. matrix
        for(unsigned int row_i = 0; row_i < r; row_i++){
            row.push_back(matrix.at(row_i).at(col_i));
        }
        transposed_matrix.push_back(row);
    }
    return transposed_matrix;
}
```

4. Rekursion

Was ist Rekursion?

Rekursion

oft hilfreich für das Lösen von Problemen mit dem *divide and conquer*-approach

Wir wollen ein Problem für n lösen.

1. Finde einen Weg, das Problem in kleinere Teilprobleme zu unterteilen:
 $k_0, k_1, \dots, k_m \quad (\forall 0 \leq i \leq m : k_i < n)$
2. Löse jedes k_i separat (vielleicht durch weiteres Unterteilen)
3. Setze die Lösung des Problems aus den Lösungen der Teilproblemen k_i zusammen

Beispiel zu Rekursion

Wir wollen eine Funktion mit den folgenden PRE und POSTs schreiben

```
// PRE:   a positive integer n
//
// POST:  returns the n-th number of a series x_n, defined as
//        x_n = 2,           for n = 1
//        x_n = 1,           for n = 2
//        x_n = x_(n-1) + x_(n-2),   for n > 2
//
// Example:
// * n == 1 ~~> 2
// * n == 2 ~~> 1
// * n == 3 ~~> 3
```

Beispiel zu Rekursion

```
// PRE:   a positive integer n
//
// POST:  returns the n-th number of a serie x_n, defined as
//        x_n = 2,           for n = 1
//        x_n = 1,           for n = 2
//        x_n = x_(n-1) + x_(n-2),   for n > 2

unsigned int compute_element(unsigned int n) {
    if (n == 1) return 2;
    else if (n == 2) return 1;
    else return compute_element(n-1) + compute_element(n-2);
}
```


Videoempfehlungen

Versucht insbesondere das Konzept von *Recursive Leap of Faith* nachzuvollziehen. Das ist quasi die Induktionsannahme bei einem Induktionsbeweis aus der Mathematik

Videos zum Thema Rekursion

- ▀ [Towers of Hanoi: A Complete Recursive Visualization](#)
- ▀ [5 Simple Steps for Solving Any Recursive Problem](#)

Übung "Partial Sum"

Aufgabe

Schreibe eine Funktion, die

1. die Summe aller natürlichen Zahlen kleiner oder gleich n per Rekursion berechnet und diesen Wert zurückgibt
2. alle addierten Terme in aufsteigender Reihenfolge (von 0 bis n) in der gleichen Rekursiven Funktion ausdrückt

Übung "Partial Sum"

- Öffnet "Partial Sum" auf **code expert**
- Überlegt euch, wie ihr das Problem mit Stift und Papier angehen würdet
- Programmiert eine (rekursive) Lösung (optional in Gruppen)

Lösung zu "Partial Sum"

```
unsigned int partial_sum(const unsigned int n) {
    if (n == 0){
        return 0;
    } else {
        // print descending
        // std::cout << n << std::endl;

        unsigned int partial = partial_sum(n - 1);

        // print ascending
        std::cout << n << std::endl;

        return n + partial;
    }
}
```

Lösung zu "Partial Sum"

```
int main() {
    std::cout << "n = ";

    unsigned int n;
    std::cin >> n;

    std::cout << partial_sum(n) << std::endl;

    return 0;
}
```

Übung "Power Function"

Frage

Wie viele Rekursionsaufrufe braucht die folgende Funktion, um x^7 zu berechnen?

```
unsigned int power(const unsigned int x, const unsigned int n) {  
  
    if (n == 0){  
        return 1;  
    } else if (n == 1) {  
        return x;  
    }  
  
    return x * power(x, n - 1);  
}
```

Antwort:

Übung "Power Function"

Frage

Wie viele Rekursionsaufrufe braucht die folgende Funktion, um x^7 zu berechnen?

```
unsigned int power(const unsigned int x, const unsigned int n) {  
  
    if (n == 0){  
        return 1;  
    } else if (n == 1) {  
        return x;  
    }  
  
    return x * power(x, n - 1);  
}
```

Antwort: 7

Übung "Power Function"

Aufgabe

Schreibt eine Funktion, die signifikant weniger Rekursionsaufrufe benötigt für grössere n . Wie viele Rekursionsaufrufe braucht eure Implementation?

Übung "Power Function"

- Öffnet "Power Function" auf **code expert**
- Überlegt euch, wie ihr das Problem mit Stift und Papier angehen würdet
- Programmiert eine (rekursive) Lösung (optional in Gruppen)
- Tipp: Die Aufgabe ist eine Verallgemeinerung der Aufgabe "Multiply with 29" der ersten Woche

Lösung zu "Power Function"

```
// POST: result == x^n
unsigned int power (const unsigned int x, const unsigned int n) {
    if(n == 0) {
        return 1;
    } else if(n == 1) {
        return x;
    } else if(n % 2 == 0) {
        int temp = power(x, n/2);
        return temp * temp;
    } else {
        return x * power(x, n-1);
    }
}
```

5. Outro

E8:T1: "Vector and matrix operations"

- Die Aufgabe kann einschüchtern. Behaltet einen Überblick über die möglichen Fälle (vielleicht mit Skizzen) und versucht, separate Funktionen für die Operationen zu implementieren.
- Verwendet **using** um das Programm übersichtlicher zu machen
- Vergesst **// Kommentare** & Referenzen und **const** nicht!

Allgemeine Fragen?

Die RW/CSE-Wiki wird von RW/CSE-Student:innen mit dem Ziel gepflegt, Informationen zum RW/CSE-Studium auszutauschen.

Geschrieben und unterhalten von RW/CSE-Studierenden. Organisiert durch:

[*rw* :: ko]



<https://wiki.math.ethz.ch/RW/>

Bis zum nächsten Mal

Schöne Woche!