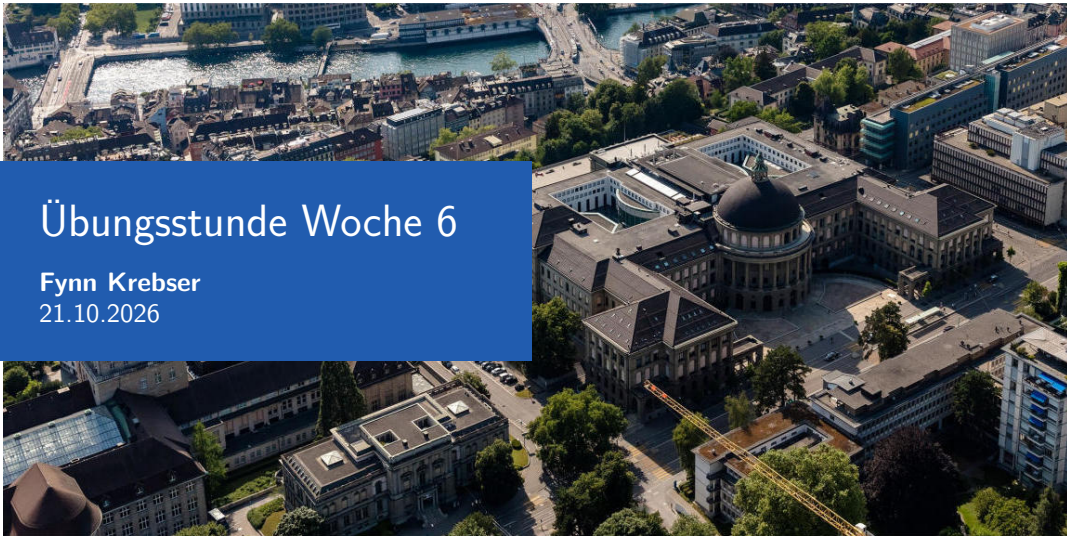


Übungsstunde Woche 6

Fynn Krebser

21.10.2026



50:50

Pub

Tel

Frage 1 ■ 50 €

Was passiert, wenn man versucht, eine C++ Referenz ohne Initialisierung zu deklarieren (z. B. `int& ref;`)?

A: Sie zeigt automatisch auf 0

B: Laufzeitfehler

C: Erzeugt eine Warnung

D: Kompilierfehler

15 1 000 000 €

14 500 000 €

13 125 000 €

12 64 000 €

11 32 000 €

10 16 000 €

9 8 000 €

8 4 000 €

7 2 000 €

6 1 000 €

5 500 €

4 300 €

3 200 €

2 100 €

1 50 €

50:50

Pub

Tel

Frage 1 ■ 50 €

Was passiert, wenn man versucht, eine C++ Referenz ohne Initialisierung zu deklarieren (z. B. `int& ref;`)?

A: Sie zeigt automatisch auf 0

B: Laufzeitfehler

C: Erzeugt eine Warnung

D: Kompilierfehler

15 1 000 000 €

14 500 000 €

13 125 000 €

12 64 000 €

11 32 000 €

10 16 000 €

9 8 000 €

8 4 000 €

7 2 000 €

6 1 000 €

5 500 €

4 300 €

3 200 €

2 100 €

1 50 €

50:50

Pub

Tel

Frage 2 ■ 100 €

Was passiert beim Aufruf von `vec[i]`, wenn der Index `i` außerhalb der Grenzen des Vectors `vec` liegt?

A: Das Programm bricht mit Exception ab

B: Vector vergrößert sich automatisch

C: Undefiniertes Verhalten (Undefined Behavior)

D: Wert 0 wird zurückgegeben

15	1 000 000 €
14	500 000 €
13	125 000 €
12	64 000 €
11	32 000 €
10	16 000 €
9	8 000 €
8	4 000 €
7	2 000 €
6	1 000 €
5	500 €
4	300 €
3	200 €
2	100 €
1	50 €

50:50

Pub

Tel

Frage 2 ■ 100 €

Was passiert beim Aufruf von `vec[i]`, wenn der Index `i` außerhalb der Grenzen des Vectors `vec` liegt?

A: Das Programm bricht mit Exception ab

B: Vector vergrößert sich automatisch

C: Undefiniertes Verhalten (Undefined Behavior)

D: Wert 0 wird zurückgegeben

15	1 000 000 €
14	500 000 €
13	125 000 €
12	64 000 €
11	32 000 €
10	16 000 €
9	8 000 €
8	4 000 €
7	2 000 €
6	1 000 €
5	500 €
4	300 €
3	200 €
2	100 €
1	50 €

50:50

Pub

Tel

Frage 3 ■ 200 €

Was beschreibt der Begriff "Call by Reference" bei Funktionsparametern am besten?

A: Es wird eine Kopie des Arguments erstellt

B: Es wird eine Referenz auf das Original übergeben

C: Es können nur Zahlen übergeben werden

D: Die Funktion darf den Wert nicht verändern

15 1 000 000 €

14 500 000 €

13 125 000 €

12 64 000 €

11 32 000 €

10 16 000 €

9 8 000 €

8 4 000 €

7 2 000 €

6 1 000 €

5 500 €

4 300 €

3 200 €

2 100 €

1 50 €

50:50

Pub

Tel

Frage 3 ■ 200 €

Was beschreibt der Begriff "Call by Reference" bei Funktionsparametern am besten?

A: Es wird eine Kopie des Arguments erstellt

B: Es wird eine Referenz auf das Original übergeben

C: Es können nur Zahlen übergeben werden

D: Die Funktion darf den Wert nicht verändern

15 1 000 000 €

14 500 000 €

13 125 000 €

12 64 000 €

11 32 000 €

10 16 000 €

9 8 000 €

8 4 000 €

7 2 000 €

6 1 000 €

5 500 €

4 300 €

3 200 €

2 100 €

1 50 €

Frage 4 ■ 300 €

Was macht der Aufruf `std::vector<int> v(5, 10);`?

A: Erstellt einen Vektor mit 10 Elementen, alle mit dem Wert 5

B: Erstellt einen Vektor mit 5 Elementen, alle mit dem Wert 10

C: Fügt die Zahlen 5 und 10 in einen leeren Vektor ein

D: Reserviert Speicher für 5 bis 10 Elemente

15	1 000 000 €
14	500 000 €
13	125 000 €
12	64 000 €
11	32 000 €
10	16 000 €
9	8 000 €
8	4 000 €
7	2 000 €
6	1 000 €
5	500 €
4	300 €
3	200 €
2	100 €
1	50 €

50:50

Pub

Tel

Frage 4 ■ 300 €

Was macht der Aufruf `std::vector<int> v(5, 10);`?

A: Erstellt einen Vektor mit 10 Elementen, alle mit dem Wert 5

B: Erstellt einen Vektor mit 5 Elementen, alle mit dem Wert 10

C: Fügt die Zahlen 5 und 10 in einen leeren Vektor ein

D: Reserviert Speicher für 5 bis 10 Elemente

15 1 000 000 €

14 500 000 €

13 125 000 €

12 64 000 €

11 32 000 €

10 16 000 €

9 8 000 €

8 4 000 €

7 2 000 €

6 1 000 €

5 500 €

4 300 €

3 200 €

2 100 €

1 50 €

Serie letzter Woche

- **Calendar:** Gut gelöst, hauptsächlich Sorgfältigkeitsaufgabe
- **Pre und Post:** Ca alle haben den Fall 0^0 vergessen.¹ Gibt verschiedene valide Lösungen
- **Fixing Functions:** Auch hier gibt es mehrere richtige Lösungen. Bei Bedarf schaut in die Lösungen.
- **Run-Length:** Wenige, aber gute Abgaben.

Fragen?

¹Ich vor einem Jahr ebenso

Ziel von Heute

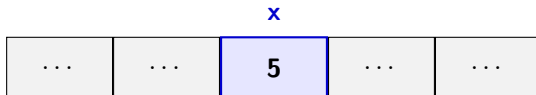
- Referenzen
- Vektoren und Matrizen

Referenzen

- Erlaubt es, einen anderen Namen für eine Variable zu verwenden
- Syntax: `int& y = x;`

```
1 int x = 5;  
2 int& ref = x; // ref ist eine Referenz auf x  
3 ref = 10; // x ist jetzt 10
```

- Hauptsächlich nützlich als Funktionsparameter.
- Wichtig: Referenzen müssen immer initialisiert werden!

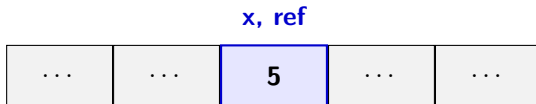


Referenzen

- Erlaubt es, einen anderen Namen für eine Variable zu verwenden
- Syntax: `int& y = x;`

```
1 int x = 5;  
2 int& ref = x; // ref ist eine Referenz auf x  
3 ref = 10; // x ist jetzt 10
```

- Hauptsächlich nützlich als Funktionsparameter.
- Wichtig: Referenzen müssen immer initialisiert werden!

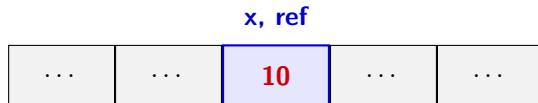


Referenzen

- Erlaubt es, einen anderen Namen für eine Variable zu verwenden
- Syntax: `int& y = x;`

```
1 int x = 5;  
2 int& ref = x; // ref ist eine Referenz auf x  
3 ref = 10; // x ist jetzt 10
```

- Hauptsächlich nützlich als Funktionsparameter.
- Wichtig: Referenzen müssen immer initialisiert werden!



Kompiliert der Code?

```
1  const double& var1;  
2  var1 = 123.1;
```

```
1  double& var2;  
2  var2 = 123.1;
```

Kompiliert der Code?

```
1  const double& var1;  
2  var1 = 123.1;
```

```
1  double& var2;  
2  var2 = 123.1;
```

Antwort

Nein, Referenzen müssen immer initialisiert werden.

Pass by Value und Pass by Reference

- Pass by Value: Kopie wird übergeben. Änderungen wirken sich NICHT auf Original aus.
- Pass by Reference: Referenz wird übergeben. Änderungen wirken sich auf Original aus.

Pass by Value:

```
void foo(int i) {  
    i = 5;  
}  
  
int main() {  
    int i = 4;  
    foo(i);  
    std::cout << i << std::endl;  
}
```

Ausgabe:

Pass by Value und Pass by Reference

- Pass by Value: Kopie wird übergeben. Änderungen wirken sich NICHT auf Original aus.
- Pass by Reference: Referenz wird übergeben. Änderungen wirken sich auf Original aus.

Pass by Value:

```
void foo(int i) {  
    i = 5;  
}  
  
int main() {  
    int i = 4;  
    foo(i);  
    std::cout << i << std::endl;  
}
```

Ausgabe: 4

Pass by Value und Pass by Reference

- Pass by Value: Kopie wird übergeben. Änderungen wirken sich NICHT auf Original aus.
- Pass by Reference: Referenz wird übergeben. Änderungen wirken sich auf Original aus.

Pass by Value:

```
void foo(int i) {  
    i = 5;  
}  
  
int main() {  
    int i = 4;  
    foo(i);  
    std::cout << i << std::endl;  
}
```

Ausgabe: 4

Pass by Reference:

```
void foo(int& i) {  
    i = 5;  
}  
  
int main() {  
    int i = 4;  
    foo(i);  
    std::cout << i << std::endl;  
}
```

Ausgabe:

Pass by Value und Pass by Reference

- Pass by Value: Kopie wird übergeben. Änderungen wirken sich NICHT auf Original aus.
- Pass by Reference: Referenz wird übergeben. Änderungen wirken sich auf Original aus.

Pass by Value:

```
void foo(int i) {  
    i = 5;  
}  
  
int main() {  
    int i = 4;  
    foo(i);  
    std::cout << i << std::endl;  
}
```

Ausgabe: 4

Pass by Reference:

```
void foo(int& i) {  
    i = 5;  
}  
  
int main() {  
    int i = 4;  
    foo(i);  
    std::cout << i << std::endl;  
}
```

Ausgabe: 5

50:50

Pub

Tel

Welche Werte haben m, n und result am Ende des Codes?

Frage 5 ■ 500 €

```
int fun(int& y, int& z) {  
    y = y + z;  
    z = y - z;  
    return y * z;  
}  
  
int main() {  
    int m = 7;  
    int n = 4;  
    int result = fun(m, n);  
}
```

A: m = 11, n = 7, result = 77

B: m = 7, n = 4, result = 77

C: m = 11, n = 3, result = 33

D: m = 4, n = 7, result = 28

15 1 000 000 €

14 500 000 €

13 125 000 €

12 64 000 €

11 32 000 €

10 16 000 €

9 8 000 €

8 4 000 €

7 2 000 €

6 1 000 €

5 500 €

4 300 €

3 200 €

2 100 €

1 50 €

50:50

Pub

Tel

Welche Werte haben m, n und result am Ende des Codes?

Frage 5 ■ 500 €

```
int fun(int& y, int& z) {  
    y = y + z;  
    z = y - z;  
    return y * z;  
}  
  
int main() {  
    int m = 7;  
    int n = 4;  
    int result = fun(m, n);  
}
```

A: m = 11, n = 7, result = 77

B: m = 7, n = 4, result = 77

C: m = 11, n = 3, result = 33

D: m = 4, n = 7, result = 28

15 1 000 000 €

14 500 000 €

13 125 000 €

12 64 000 €

11 32 000 €

10 16 000 €

9 8 000 €

8 4 000 €

7 2 000 €

6 1 000 €

5 500 €

4 300 €

3 200 €

2 100 €

1 50 €

Referenzen: Übung (a)

Was ist die Ausgabe des Programms für die folgende Variante von foo?

```
1 int main() {  
2     int a = 0;  
3     int b = 1;  
4     for (int i = 0; i < 5; ++i) {  
5         b = foo(a, b);  
6         std::cout << b << " ";  
7     }  
8     return 0;  
9 }
```

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

Referenzen: Übung (a)

Was ist die Ausgabe des Programms für die folgende Variante von foo?

```
1 int main() {  
2     int a = 0;  
3     int b = 1;  
4     for (int i = 0; i < 5; ++i) {  
5         b = foo(a, b);  
6         std::cout << b << " ";  
7     }  
8     return 0;  
9 }
```

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

Ausgabe

1 2 4 8 16

Referenzen: Übung (b)

Was ist die Ausgabe des Programms für die folgende Variante von foo?

```
1 int main() {  
2     int a = 0;  
3     int b = 1;  
4     for (int i = 0; i < 5; ++i) {  
5         b = foo(a, b);  
6         std::cout << b << " ";  
7     }  
8     return 0;  
9 }
```

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

Referenzen: Übung (b)

Was ist die Ausgabe des Programms für die folgende Variante von foo?

```
1 int main() {  
2     int a = 0;  
3     int b = 1;  
4     for (int i = 0; i < 5; ++i) {  
5         b = foo(a, b);  
6         std::cout << b << " ";  
7     }  
8     return 0;  
9 }
```

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

Ausgabe

1 1 1 1 1

Referenzen: Übung (c)

Was ist die Ausgabe des Programms für die folgende Variante von foo?

```
1 int main() {  
2     int a = 0;  
3     int b = 1;  
4     for (int i = 0; i < 5; ++i) {  
5         b = foo(a, b);  
6         std::cout << b << " ";  
7     }  
8     return 0;  
9 }
```

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

Referenzen: Übung (c)

Was ist die Ausgabe des Programms für die folgende Variante von foo?

```
1 int main() {  
2     int a = 0;  
3     int b = 1;  
4     for (int i = 0; i < 5; ++i) {  
5         b = foo(a, b);  
6         std::cout << b << " ";  
7     }  
8     return 0;  
9 }
```

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

Ausgabe

1 1 1 1 1

50:50

Pub

Tel

Frage 6 ■ 1 000 €

Welche Ausgabe erzeugt dieser Codeabschnitt?

```
int a = 10;  
int& r1 = a;  
int& r2 = r1;  
  
r2 = 20;  
std::cout << a << " " << r1;
```

A: 10 10

B: 20 10

C: 10 20

D: 20 20

15 1 000 000 €

14 500 000 €

13 125 000 €

12 64 000 €

11 32 000 €

10 16 000 €

9 8 000 €

8 4 000 €

7 2 000 €

6 1 000 €

5 500 €

4 300 €

3 200 €

2 100 €

1 50 €

50:50

Pub

Tel

Frage 6 ■ 1 000 €

Welche Ausgabe erzeugt dieser Codeabschnitt?

```
int a = 10;  
int& r1 = a;  
int& r2 = r1;  
  
r2 = 20;  
std::cout << a << " " << r1;
```

A: 10 10

B: 20 10

C: 10 20

D: 20 20

15 1 000 000 €

14 500 000 €

13 125 000 €

12 64 000 €

11 32 000 €

10 16 000 €

9 8 000 €

8 4 000 €

7 2 000 €

6 1 000 €

5 500 €

4 300 €

3 200 €

2 100 €

1 50 €

Referenzen: swap

Schreibt eine Funktion `swap`, die die Werte zweier `int`-Variablen vertauscht.

```
1  int a = 5;
2  int b = 6;
3  // hier kommt euer Funktionsaufruf
4  std::cout << a << "\n"; // Ausgabe: 6
5  std::cout << b << "\n"; // Ausgabe: 5
```

Referenzen: swap (Lösung)

```
1  // POST: the values of i and j are swapped
2  void swap (int& i, int& j) {
3      const int h = i;
4      i = j;
5      j = h;
6  }
```

```
int a = 5;
int b = 6;
swap(a, b);
```

Anwendungen Referenzen

Mögliche Anwendungen von Referenzen:

- Eine Funktion welche mehrere Werte zurückgibt
- Gewisse Objekte können nicht kopiert werden (z.B. output streams)²

```
1 // POST: x1 and x2 are the solutions of the quadratic equation
2 int solve_quadratic(double a, double b, double c,
3     double& x1, double& x2);
```

²Das ist lediglich ein Beispiel, ist für die Prüfung nicht relevant.

Frage 7 ■ Was ist das Hauptproblem in folgendem Codeabschnitt?

```
int& get_value() {  
    int x = 42;  
    return x;  
}  
  
int main() {  
    int& ref = get_value();  
    std::cout << ref;  
}
```

A: Gibt Referenz auf lokale Variable zurück

B: Funktion muss vom Typ void sein

C: std::cout kann keine Referenzen ausgeben

D: x muss als const int deklariert werden

15 1 000 000 €

14 500 000 €

13 125 000 €

12 64 000 €

11 32 000 €

10 16 000 €

9 8 000 €

8 4 000 €

7 2 000 €

6 1 000 €

5 500 €

4 300 €

3 200 €

2 100 €

1 50 €

Frage 7 ■ Was ist das Hauptproblem in folgendem Codeabschnitt?

```
int& get_value() {  
    int x = 42;  
    return x;  
}  
  
int main() {  
    int& ref = get_value();  
    std::cout << ref;  
}
```

A: Gibt Referenz auf lokale Variable zurück

B: Funktion muss vom Typ void sein

C: std::cout kann keine Referenzen ausgeben

D: x muss als const int deklariert werden

15 1 000 000 €

14 500 000 €

13 125 000 €

12 64 000 €

11 32 000 €

10 16 000 €

9 8 000 €

8 4 000 €

7 2 000 €

6 1 000 €

5 500 €

4 300 €

3 200 €

2 100 €

1 50 €

Referenzen als Rückgabewert

- Referenzen können auch als Rückgabewert verwendet werden.

```
1  int& increment (int& m){
2      return m + 1;
3  }
4
5  int main() {
6      int a = 5;
7      increment(increment(a));
8      std::cout << a << std::endl; // Ausgabe: 7
9  }
```

Achtung!

Die zurückgegebene Referenz darf nicht auf eine lokale Variable zeigen, da diese nach dem Funktionsaufruf zerstört wird.

⇒ Nur Referenzen auf Variablen zurückgeben, die ihr per Referenz bereits übergeben habt!

Vektoren

- LinAlg: Vektoren sind Elemente eines Vektorraums.

Vektoren

- LinAlg: Vektoren sind Elemente eines Vektorraums.
- C++: Vektoren sind eine Liste von Elementen *desselben* Typs (z.B. `intn`)
- ABER: Die Grösse kann dynamisch angepasst werden.

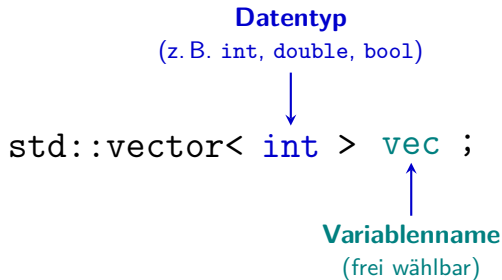
Datentyp
(z. B. `int`, `double`, `bool`)

↓

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

↑

Variablenname
(frei wählbar)



Wichtigste Vektoroperationen

Die wichtigsten Operationen auf einem Vektor `v` sind:

- `v.push_back(x)`: Fügt das Element `x` am Ende des Vektors `v` hinzu.
- `std::ssize(v)`: Gibt die Anzahl der Elemente im Vektor `v` zurück.³
- `v[i]`: Zugriff auf das `i`-te Element des Vektors `v` (0-basiert). `i` muss kleiner als `std::ssize(v)` sein. Ansonsten **undefined behavior**.

³Funktioniert erst seit C++20 Standard.

Wichtigste Vektoroperationen

Die wichtigsten Operationen auf einem Vektor `v` sind:

- `v.push_back(x)`: Fügt das Element `x` am Ende des Vektors `v` hinzu.
- `std::ssize(v)`: Gibt die Anzahl der Elemente im Vektor `v` zurück.³
- `v[i]`: Zugriff auf das `i`-te Element des Vektors `v` (0-basiert). `i` muss kleiner als `std::ssize(v)` sein. Ansonsten **undefined behavior**.

Um Vektoren zu erstellen gibt es folgende Möglichkeiten:

- `std::vector<int> v`; Erstellt einen leeren Vektor `v` vom Typ `int`.
- `std::vector<int> v(n, a)`; Erstellt einen Vektor `v` vom Typ `int` mit `n` Elementen, die alle den Wert `a` haben.

³Funktioniert erst seit C++20 Standard.

Frage 8 ■ 4 000 €

Was beschreibt die Ausgabe am besten?

```
std::vector<unsigned int> arr = {2, 3, 1, 4, 5};  
int sum = 0;  
  
for (unsigned int i = 0;  
     i < arr.size() && arr.at(i) < 10;  
     i++) {  
    sum += arr.at(i);  
}  
  
std::cout << sum;
```

A: 15

B: 10

C: Terminiert nie

D: Bricht wegen illegalem Indexzugriff ab

15	1 000 000 €
14	500 000 €
13	125 000 €
12	64 000 €
11	32 000 €
10	16 000 €
9	8 000 €
8	4 000 €
7	2 000 €
6	1 000 €
5	500 €
4	300 €
3	200 €
2	100 €
1	50 €

Frage 8 ■ 4 000 €

Was beschreibt die Ausgabe am besten?

```
std::vector<unsigned int> arr = {2, 3, 1, 4, 5};  
int sum = 0;  
  
for (unsigned int i = 0;  
     i < arr.size() && arr.at(i) < 10;  
     i++) {  
    sum += arr.at(i);  
}  
  
std::cout << sum;
```

A: 15

B: 10

C: Terminiert nie

D: Bricht wegen illegalem Indexzugriff ab

15	1 000 000 €
14	500 000 €
13	125 000 €
12	64 000 €
11	32 000 €
10	16 000 €
9	8 000 €
8	4 000 €
7	2 000 €
6	1 000 €
5	500 €
4	300 €
3	200 €
2	100 €
1	50 €

50:50

Pub

Tel

Frage 9 ■ 8 000 €

Welcher Wert wird hier auf der Konsole ausgegeben?

```
std::vector<int> v;  
v.push_back(3);  
v.push_back(7);  
v.push_back(1);  
  
v[1] += v[0];  
std::cout << v[1];
```

A: 3

B: 4

C: 8

D: 10

15 1 000 000 €

14 500 000 €

13 125 000 €

12 64 000 €

11 32 000 €

10 16 000 €

9 8 000 €

8 4 000 €

7 2 000 €

6 1 000 €

5 500 €

4 300 €

3 200 €

2 100 €

1 50 €

50:50

Pub

Tel

Frage 9 ■ 8 000 €

Welcher Wert wird hier auf der Konsole ausgegeben?

```
std::vector<int> v;  
v.push_back(3);  
v.push_back(7);  
v.push_back(1);  
  
v[1] += v[0];  
std::cout << v[1];
```

A: 3

B: 4

C: 8

D: 10

15 1 000 000 €

14 500 000 €

13 125 000 €

12 64 000 €

11 32 000 €

10 16 000 €

9 8 000 €

8 4 000 €

7 2 000 €

6 1 000 €

5 500 €

4 300 €

3 200 €

2 100 €

1 50 €

[code]expert: Reversing Vector

- [code]expert: Reversing Vector 

LinAlg Einschub: Matrizen

Eine Matrix ist ein Gitter von Zahlen,

$$\begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{pmatrix}.$$

- Die Dimension einer Matrix wird als $m \times n$ angegeben, wobei m die Anzahl der Zeilen und n die Anzahl der Spalten ist.

Matrizen können transponiert werden, indem man die Zeilen und Spalten vertauscht. Die Transponierte der obigen Matrix ist:

$$\begin{pmatrix} 1 & 4 \\ 2 & 5 \\ 3 & 6 \end{pmatrix}.$$

LinAlg Einschub: Matrizenmultiplikation

Die Multiplikation zweier Matrizen A und B ist definiert, wenn die Anzahl der Spalten von A gleich der Anzahl der Zeilen von B ist. Das Ergebnis ist eine neue Matrix C , wobei jedes Element c_{ij} durch das Skalarprodukt der i -ten Zeile von A und der j -ten Spalte von B berechnet wird:

$$c_{ij} = \sum_{k=1}^n a_{ik} b_{kj}.$$

Beispiel:

$$\begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{pmatrix} \times \begin{pmatrix} 7 & 8 \\ 9 & 10 \\ 11 & 12 \end{pmatrix} = \begin{pmatrix} 58 & 64 \\ 139 & 154 \end{pmatrix}.$$

Z.B. ist $c_{11} = 1 * 7 + 2 * 9 + 3 * 11 = 58$ und $c_{12} = 1 * 8 + 2 * 10 + 3 * 12 = 64$.

Matrizen

- Eine Matrix ist eine zweidimensionale Anordnung von Elementen desselben Typs.
- In C++ kann man Matrizen als Vektor von Vektoren darstellen.⁴

⁴Daher auch 2d-Vektor genannt.

Matrizen

- Eine Matrix ist eine zweidimensionale Anordnung von Elementen desselben Typs.
- In C++ kann man Matrizen als Vektor von Vektoren darstellen.⁴
- Um nicht immer `std::vector<std::vector<int>>` schreiben zu müssen, kann man einen Typ-Alias definieren:

```
using irow    = std::vector<int>;    // Typ-Alias für eine Zeile
using imatrix = std::vector<irow>;   // Typ-Alias für eine Matrix
```

⁴Daher auch 2d-Vektor genannt.

Matrix erstellen

Alternative 1: Zeile für Zeile

Dynamischer Aufbau mit push_back

```
imatrix read_matrix() {
    int r, c;
    std::cin >> r >> c;
    imatrix matrix = imatrix(0);
    for (int row_index = 0; row_index < r; row_index
        ++){
        irow row = irow(0);
        for (int col_index = 0; col_index < c;
            col_index++){
            int element;
            std::cin >> element;
            row.push_back(element);
        }
        matrix.push_back(row);
    }
    return matrix;
}
```

Alternative 2: Direkt allozieren

Erstellt $r \times c$ Matrix vorab; Zugriff per [i][j]

```
imatrix read_matrix() {
    int r, c;
    std::cin >> r >> c;
    // Construct r x c matrix filled with 0
    imatrix matrix(r, irow(c, 0));

    for (int row_index = 0; row_index < r; row_index
        ++){
        for (int col_index = 0; col_index < c;
            col_index++){
            int element;
            std::cin >> element;
            matrix[row_index][col_index] = element;
        }
    }
    return matrix;
}
```

50:50

Pub

Tel

Frage 10 ■ 16 000 €

Welcher Wert wird hier auf der Konsole ausgegeben?

```
std::vector<std::vector<int>> M = {  
    {1, 2, 3},  
    {4, 5, 6}  
};  
std::cout << M[1][0];
```

A: 1

B: 2

C: 4

D: 5

15 1 000 000 €

14 500 000 €

13 125 000 €

12 64 000 €

11 32 000 €

10 16 000 €

9 8 000 €

8 4 000 €

7 2 000 €

6 1 000 €

5 500 €

4 300 €

3 200 €

2 100 €

1 50 €

50:50

Pub

Tel

Frage 10 ■ 16 000 €

Welcher Wert wird hier auf der Konsole ausgegeben?

```
std::vector<std::vector<int>> M = {  
    {1, 2, 3},  
    {4, 5, 6}  
};  
std::cout << M[1][0];
```

A: 1

B: 2

C: 4

D: 5

15 1 000 000 €

14 500 000 €

13 125 000 €

12 64 000 €

11 32 000 €

10 16 000 €

9 8 000 €

8 4 000 €

7 2 000 €

6 1 000 €

5 500 €

4 300 €

3 200 €

2 100 €

1 50 €

[code]expert: Matrix Tanspose

- [code]expert: Matrix Transpose 

Tipps zu den Übungen

Sorting by Swapping

Tipps zu den Übungen

Sorting by Swapping

- Verwendet die Swap-Funktion der Übungsstunde

Tipps zu den Übungen

Sorting by Swapping

- Verwendet die Swap-Funktion der Übungsstunde
- Ein- und Ausgabe müsst ihr nicht selbst machen. Das wird bereits an die Funktion übergeben. Ihr müsst nur die Funktion implementieren. (Insbesondere kein `main()` schreiben!)

Tipps zu den Übungen

Sorting by Swapping

- Verwendet die Swap-Funktion der Übungsstunde
- Ein- und Ausgabe müsst ihr nicht selbst machen. Das wird bereits an die Funktion übergeben. Ihr müsst nur die Funktion implementieren. (Insbesondere kein `main()` schreiben!)

Vector and matrix operations

Tipps zu den Übungen

Sorting by Swapping

- Verwendet die Swap-Funktion der Übungsstunde
- Ein- und Ausgabe müsst ihr nicht selbst machen. Das wird bereits an die Funktion übergeben. Ihr müsst nur die Funktion implementieren. (Insbesondere kein `main()` schreiben!)

Vector and matrix operations

- Verwendet Funktionen die ihr bereits implementiert habt (z.B. `dot_produc` wenn ihr Matrixmultiplikation implementiert)