

**OOP1**

**KU**

**21-04-2021**

# HEUTE

- Organisatorisches
- Singleton Beispiel 15min
- Datenstruktur der Woche - Map (and Multimap) 5min
- File einlesen - map - rapidjson ausgeben 30min
- Menti Fragen end

# ORGANISATORISCHES A1

- Samstag 24.4.2021 - 11 Uhr
- A1 Lösung von Aleks

# ORGANISATORISCHES A2

- Milestone 1
  - ausgegeben
  - 28.4.2021 KU-Fragestunde
  - 8.5.2021 (soft) Deadline
  - 10.5. - 14.5.2021 Feedbackgespräche
- Milestone 2
  - 22.4.2021 Morgen (nach VO-Stream)
  - 19.5.2021 KU-Fragestunde
  - 29.5.2021 Deadline
  - 31.5. - 11.6.2021 Abgabegespräche

# SINGLETON BEISPIEL

# DATENSTRUKTUR DER WOCHE

## Map - std::map (std::multimap)

<https://www.cplusplus.com/reference/map/map/> <http://www.cplusplus.com/reference/map/multimap/>

- Key - Value Pairs
- Ein Schlüssel (key) verweist auf einen bestimmten Wert (value)

	KEYS	VALUES	
	Jan	327.2	
	Feb	368.2	
	Mar	197.6	
	Apr	178.4	
	May	100.0	
	Jun	69.9	
	Jul	32.3	
Aug →	Aug	37.3	→ 37.3
	Sep	19.0	
	Oct	37.0	
	Nov	73.2	
	Dec	110.9	
	Annual	1551.0	

# STD::MAP BRAUCHT STD::PAIR

<https://www.cplusplus.com/reference/utility/pair/pair/>

- `std::pair` ist wie ein struct aus 2 Variablen
- Die Datentypen können unterschiedlich sein
- Wir wollen festlegen, dass es im Jänner durchschnittlich 3.55 Grad hat

```
1 // erstellt ein pair
2 //           type of 1st, type of 2nd
3 std::pair<std::string, double> mypair("January", 3.55);
4
5 // Zugriff
6 std::cout << mypair.first << std::endl; // "January"
7 std::cout << mypair.second << std::endl; // 3.55
```

# STD::MAP BRAUCHT STD::PAIR

<https://www.cplusplus.com/reference/utility/pair/pair/>

- `std::pair` ist wie ein struct aus 2 Variablen
- Die Datentypen können unterschiedlich sein
- Wir wollen festlegen, dass es im Jänner durchschnittlich 3.55 Grad hat

```
1 // erstellt ein pair
2 //           type of 1st, type of 2nd
3 std::pair<std::string, double> mypair("January", 3.55);
4
5 // Zugriff
6 std::cout << mypair.first << std::endl; // "January"
7 std::cout << mypair.second << std::endl; // 3.55
```



# MAP - STD::MAP

```
1 //      KEY      VALUE
2 //      month      Ø temperature
3 std::map<std::string, double> mymap; // erstellt eine map
4
5 // Element in die Map einfügen
6 mymap.insert(std::pair<std::string, double>("January", 3.55));
7
8 // Finde ein Element mithilfe des Keys - aber sehr lang :(
9 std::map<std::string, double>::iterator it=mymap.find("January");
10
11 auto it = mymap.find("January"); // verwende auto alternativ
12
13 // zugriff auf den key und value
14 std::cout << it->first << std::endl; // "January"
15 std::cout << it->second << std::endl; // 3.55
16
17 mymap.erase(it); // löscht das Element aus map mittels iterator
18 mymap.erase("January"); // löscht das Element mittels key
19
20 mymap.empty(); // gibt true oder false zurück
21 mymap.clear(); // leert die map
```

# MAP - STD::MAP

```
1 //          KEY          VALUE
2 //          month        Ø temperature
3 std::map<std::string, double> mymap; // erstellt eine map
4
5 // Element in die Map einfügen
6 mymap.insert(std::pair<std::string, double>("January", 3.55));
7
8 // Finde ein Element mithilfe des Keys - aber sehr lang :(
9 std::map<std::string, double>::iterator it=mymap.find("January");
10
11 auto it = mymap.find("January"); // verwende auto alternativ
12
13 // zugriff auf den key und value
14 std::cout << it->first << std::endl; // "January"
15 std::cout << it->second << std::endl; // 3.55
16
17 mymap.erase(it); // löscht das Element aus map mittels iterator
18 mymap.erase("January"); // löscht das Element mittels key
19
20 mymap.empty(); // gibt true oder false zurück
21 mymap.clear(); // leert die map
```

# MAP - STD::MAP

```
1 //          KEY          VALUE
2 //          month        Ø temperature
3 std::map<std::string, double> mymap; // erstellt eine map
4
5 // Element in die Map einfügen
6 mymap.insert(std::pair<std::string, double>("January", 3.55));
7
8 // Finde ein Element mithilfe des Keys - aber sehr lang :(
9 std::map<std::string, double>::iterator it=mymap.find("January");
10
11 auto it = mymap.find("January"); // verwende auto alternativ
12
13 // zugriff auf den key und value
14 std::cout << it->first << std::endl; // "January"
15 std::cout << it->second << std::endl; // 3.55
16
17 mymap.erase(it); // löscht das Element aus map mittels iterator
18 mymap.erase("January"); // löscht das Element mittels key
19
20 mymap.empty(); // gibt true oder false zurück
21 mymap.clear(); // leert die map
```

# MAP - STD::MAP

```
1 //          KEY          VALUE
2 //          month        Ø temperature
3 std::map<std::string, double> mymap; // erstellt eine map
4
5 // Element in die Map einfügen
6 mymap.insert(std::pair<std::string, double>("January", 3.55));
7
8 // Finde ein Element mithilfe des Keys - aber sehr lang :(
9 std::map<std::string, double>::iterator it=mymap.find("January");
10
11 auto it = mymap.find("January"); // verwende auto alternativ
12
13 // zugriff auf den key und value
14 std::cout << it->first << std::endl; // "January"
15 std::cout << it->second << std::endl; // 3.55
16
17 mymap.erase(it); // löscht das Element aus map mittels iterator
18 mymap.erase("January"); // löscht das Element mittels key
19
20 mymap.empty(); // gibt true oder false zurück
21 mymap.clear(); // leert die map
```

# MAP - STD::MAP

```
1 //          KEY          VALUE
2 //          month        Ø temperature
3 std::map<std::string, double> mymap; // erstellt eine map
4
5 // Element in die Map einfügen
6 mymap.insert(std::pair<std::string, double>("January", 3.55));
7
8 // Finde ein Element mithilfe des Keys - aber sehr lang :(
9 std::map<std::string, double>::iterator it=mymap.find("January");
10
11 auto it = mymap.find("January"); // verwende auto alternativ
12
13 // zugriff auf den key und value
14 std::cout << it->first << std::endl; // "January"
15 std::cout << it->second << std::endl; // 3.55
16
17 mymap.erase(it); // löscht das Element aus map mittels iterator
18 mymap.erase("January"); // löscht das Element mittels key
19
20 mymap.empty(); // gibt true oder false zurück
21 mymap.clear(); // leert die map
```

# MAP - STD::MAP

```
1 //      KEY      VALUE
2 //      month      Ø temperature
3 std::map<std::string, double> mymap; // erstellt eine map
4
5 // Element in die Map einfügen
6 mymap.insert(std::pair<std::string, double>("January", 3.55));
7
8 // Finde ein Element mithilfe des Keys - aber sehr lang :(
9 std::map<std::string, double>::iterator it=mymap.find("January");
10
11 auto it = mymap.find("January"); // verwende auto alternativ
12
13 // zugriff auf den key und value
14 std::cout << it->first << std::endl; // "January"
15 std::cout << it->second << std::endl; // 3.55
16
17 mymap.erase(it); // löscht das Element aus map mittels iterator
18 mymap.erase("January"); // löscht das Element mittels key
19
20 mymap.empty(); // gibt true oder false zurück
21 mymap.clear(); // leert die map
```

# MAP - STD::MAP

```
1 //      KEY      VALUE
2 //      month      Ø temperature
3 std::map<std::string, double> mymap; // erstellt eine map
4
5 // Element in die Map einfügen
6 mymap.insert(std::pair<std::string, double>("January", 3.55));
7
8 // Finde ein Element mithilfe des Keys - aber sehr lang :(
9 std::map<std::string, double>::iterator it=mymap.find("January");
10
11 auto it = mymap.find("January"); // verwende auto alternativ
12
13 // zugriff auf den key und value
14 std::cout << it->first << std::endl; // "January"
15 std::cout << it->second << std::endl; // 3.55
16
17 mymap.erase(it); // löscht das Element aus map mittels iterator
18 mymap.erase("January"); // löscht das Element mittels key
19
20 mymap.empty(); // gibt true oder false zurück
21 mymap.clear(); // leert die map
```

**UND WAS IST JETZT DIE  
MULTIMAP?**



# MAP VS. MULTIMAP

```
1  std::map<std::string, double> mymap;  
2  
3  mymap.insert(std::pair<std::string, double>("January", 3.55));  
4  mymap.insert(std::pair<std::string, double>("January", 4.55));  
5  mymap.insert(std::pair<std::string, double>("February", 8.02));  
6  
7  for(auto iterator : mymap)  
8  {  
9      std::cout << iterator.first << " " << iterator.second << std::endl;  
10 }
```

```
1  February 8.02  
2  January 3.55
```

Map kann nur ein Element mit selbem Key speichern

# MAP VS. MULTIMAP

```
1  std::multimap<std::string, double> mymap;  
2  
3  mymap.insert(std::pair<std::string, double>("January", 3.55));  
4  mymap.insert(std::pair<std::string, double>("January", 4.55));  
5  mymap.insert(std::pair<std::string, double>("February", 8.02));  
6  
7  for(auto iterator : mymap)  
8  {  
9      std::cout << iterator.first << " " << iterator.second << std::endl;  
10 }
```

```
1  February 8.02  
2  January 3.55  
3  January 4.55
```

Multimap kann mehrere Elemente mit selbem Key speichern

# FILE EINLESEN - MAP - RAPIDJSON AUSGEBEN

# MENTI FRAGEN