Teil 9
Templates

- Template-Funktionen
- Template-Klassen
- Template-Spezialisierung
- Generische Lambdas

Template-Funktionen

```
void print(int number)
{
  cout << number << endl;
}

void print(string text)
{
  cout << number << endl;
}

int main()
{
  print(123);
  print("eins zwei drei");
  return 0;
}
```

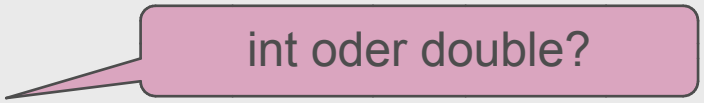
Template-Funktionen

```
template<typename T> void print(T value)
{
  cout << value << endl;
}

int main()
{
  print<int>(123);
  print<string>("eins zwei drei");
  return 0;
}
```

- Templates sind Schablonen, Anleitungen für den Compiler, eigenständig Funktionen zu erstellen
- werden für jeden verwendeten Typ instanziert
- Templates sind oft vollständig im Header-File definiert, weil jede Compilation-Unit diese „Anleitung“ braucht

Template-Funktionen

```
template<typename T> void print(T value)
{
  cout << value << endl;
}

int main()
{
  print(123);
  print("eins zwei drei");
  return 0;
}
```



- Template-Parameter können weggelassen werden, wenn sie sich aus dem Kontext ergeben
- lassen sich auch mit Overloading mischen

Template-Klassen

```
template<typename T, int N> class Stack
{
public:
  Stack() : values(), index(0) {}
  void push(T value)
  {
    values[index] = value;
    ++index;
  }
  T pop()
  {
    return values[index--];
  }
private:
  T values[N];
  int index;
};
```

Template-Klassen

```
int main()
{
  Stack<double, 100> stack;
  stack.push(0.5);
  stack.push(1.5);
  stack.push(2.5);
  cout << stack.pop() << endl;
  cout << stack.pop() << endl;
  cout << stack.pop() << endl;
  return 0;
}
```

Template-Klassen

```
template<typename T, int N> class Stack
{
public:
  Stack() : values(), index(0) {}
  void push(T value);
  T pop();
};

template<typename T, int N> void Stack<T, N>::push(T value)
{
  values[index] = value;
  ++index;
}

template<typename T, int N> T Stack<T, N>::pop()
{
  return values[index--];
}
```

Template-Spezialisierung

```
template<typename T> class Stack
{
public:
  Stack() : values() {}
  void push(T value);
  T pop();
private:
  std::vector<T>* values;
};

template<> class Stack<bool>
{
public:
  Stack() : values() {}
  void push(bool value);
  bool pop();
private:
  std::vector<unsigned int> values;
};
```

Generische Lambdas

C++ 14

- mit C++14 auch generische Lambdas möglich

```
auto add = [](auto a, auto b) { return a + b };

int sum = add(44, 55);
string text = add("c", "++");
```

- Lambda-Funktionen existieren seit C++11 und sind namenlose Funktionen, die direkt beim Aufruf oder einer Zuweisung definiert werden