G. Einsendeaufgabe

Objektorientierte Programmierung mit C++ und Qt

Code:
CPBS18E-XX1-N01

Name:	Vorname:
Rein	Wjatscheslaw
Postleitzahl und Ort:	**Straße:**
14193 Berlin	Cunostr. 54A
Studien- bzw. Vertrags-Nr.:	**Lehrgangs-Nr.:**
800463563	246

Fernlehrer/in:
Datum:
Note:
Unterschrift Fernlehrer/in:

Bitte reichen Sie Ihre Lösungen über die Online-Lernplattform ein oder schicken Sie uns diese per Post. Geben Sie bitte immer den Code zum Studienheft an (siehe oben rechts).

Inhaltsverzeichnis

1 Aufgabe:

Aufgabe:

Bitte sehen Sie sich das folgende Funktions-Template an. Es soll zwei Werte vom selben Typ addieren und das Ergebnis zurückliefern.

```
template <typename T> int addieren(T wert1, T wert2)
{
 return (wert1 + wert2);
}
```

Wo könnten Probleme mit dem Template auftreten? Wie können Sie diese Probleme vermeiden?

10 Pkt.

Lösung:

Beim Template ist ein Rückgabewert als „int“ definiert, das heißt es können nur die Integer-Werte an die aufrufende Funktion zurück geliefert werden.

Um diese Probleme zu vermeiden, sollte expliziert der Datentyp von dem Rückgabewert auch als Template-Typ (T) eingegeben werden (siehe Listing 1.1).

```
1       template <typename T> T addieren(T wert1, T wert2)
2       {
3         T temp;
4         temp = (wert1 + wert2);
5         return temp;
6       }
7
```

Listing 1.1: Lösung der Aufgabe 1

2 Aufgabe:

Aufgabe:

Lässt sich das Funktions-Template aus der Aufgabe 1 mit der folgenden Anweisung aufrufen?

```
addieren<int, int>(20, 10);
```

Begründen Sie bitte Ihre Antwort.

10 Pkt.

Lösung:

Beim Aufruf ist es möglich explizit den Template-Typ einzugeben, deren Anzahl aber hängt nicht von der Anzahl der Argumente sondern von der definierten Typen im Template ab. Deswegen bringt der Compiler ein Fehler, da es kein zweiter Template-Typ zu finden ist (Vgl. Heft CPBS18E Seite 9).

3 Aufgabe:

Aufgabe:

Erstellen Sie ein Funktions-Template als Konsolenprogramm, das zwei Werte addiert und das Ergebnis zurückliefert.

Programmieren Sie das Template so, dass es grundsätzlich auch mit `string`-Typen aufgerufen werden kann. Dann soll allerdings zusätzlich eine Meldung erscheinen, dass beim Addieren von Zeichenketten die Werte aneinandergehängt werden. Diese Meldung soll im Template erzeugt werden.

20 Pkt.

Lösung:

```
1   template <typename T> void ausgabe(T wert)
2   {
3     cout << "Die addierten Werte: " << wert << "\n\n";
4   }
5   template <typename T> T addition(T wert1, T wert2)
6   {
7     T temp;
8     temp = wert1 + wert2;
9
10    //Gibt den Namen des Typen aus. Ausgegeben wird nur die Kurzform:
11    cout << typeid(T).name() <<": \n";
12    //Detaillierte Ausgabe
13    if (*(typeid(T).name()) == 'i')
14    {
15      cout << "Typ ist integer" << endl;
16    }
17    if (*(typeid(T).name()) == 'd')
18    {
19      cout << "Typ ist double" << endl;
20    }
21    if (!(strcmp(typeid(T).name(),
22          "NSt7__cxx1112basic_stringIcSt11char_traitsIcESaIcEEE")))
23    {
24      cout << "Typ ist string" << endl;
25      //Bitte in einer Zeile eingeben
26      cout << "Beim Addieren von Zeichenketten werden die Werte aneinandergehaengt"
27           << endl;
28    }
29    return temp;
30  }
31
32
```

Listing 3.1: Templates für Addieren und Ausgabe

```
1   int main()
2   {
3     int retI, zahl1, zahl2;
4     double retD, doubleZahl1, doubleZahl2;
5     string retS;
6     string zeichenkette1 = "aufgabe 3";
7     string zeichenkette2 = "Einsende";
8     zahl1 = 100;
9     zahl2 = 15;
10    doubleZahl1 = 101.981;
11    doubleZahl2 = 102.109;
12
13    //Additionen von unterschiedlichen Typen
14    retI = addition(zahl1, zahl2);
15    ausgabe(retI);
16    retD = addition(doubleZahl1, doubleZahl2);
17    ausgabe(retD);
18    retS = addition(zeichenkette2, zeichenkette1);
19    ausgabe(retS);
20    return 0;
21  }
22
```

Listing 3.2: Hauptprogram für die Aufgabe 3

Die Abbildung (3.1) zeigt die Ergebnisse der dritten Aufgabe.

```
i:
Typ ist integer
Die addierten Werte: 115

d:
Typ ist double
Die addierten Werte: 204.09

NSt7__cxx1112basic_stringIcSt11char_traitsIcESaIcEEE:
Typ ist string
Hinweis: Beim Addieren von Zeichenketten werden die Werte aneinandergehaengt
Die addierten Werte: Einsendeaufgabe 3

13:29:51: C:\Qt\Projekte\CPBS18E\build-aufgabe3-Desktop_Qt_5_15_2_MinGW_64_bit-Debug\debug\aufgabe3.exe
```

Abbildung 3.1: Ausgabefenster der Aufgabe 3

4 Aufgabe:

Aufgabe:

Erstellen Sie ein Programm mit grafischer Oberfläche, das eine beliebig lange Liste mit ganzen Zahlen in ein Listenfeld einliest, die Liste sortiert und alle Duplikate entfernt. Die geänderte Liste soll dann in einem zweiten Listenfeld anzeigt werden. Das Einlesen der Zahlen muss dabei nicht unbedingt per Hand durch den Anwender erfolgen. Sie können die Liste auch automatisch mit zufälligen Zahlen füllen lassen. Es gibt mehrere Ansätze für die Lösung dieser Aufgabe. Welchen Ansatz Sie wählen, ist Ihnen freigestellt. Sie sollten aber in jedem Fall Container-Klassen nutzen. Bitte beschreiben Sie Ihren Ansatz kurz.

Ein Tipp zur Lösung:

> Es gibt eine Container-Klasse von Qt, die keine doppelten Einträge zulässt. Sie können eine Liste vom Typ `QList` in diese Container-Klasse umwandeln. Entsprechende Methoden finden Sie in der Qt-Dokumentation.
>
> Sie können aber auch über einen Iterator selbst nach doppelten Einträgen suchen und diese Einträge mit der Methode `remove()` entfernen.

Beschreiben Sie für diese Aufgabe zusätzlich, welche grundsätzlichen Schritte für die Lösung erforderlich sind. Schicken Sie auch das vollständige Projekt mit allen Unterordnern und Dateien ein. Um Übertragungszeit und -kosten zu sparen, können Sie das Projekt mit einem geeigneten Programm packen – zum Beispiel mit WinZip oder direkt über das Betriebssystem.

60 Pkt.

Lösung:

Wie in der Aufgabenstellung erwähnt ist, existieren mehrere Lösungsansetze um angeforderte Aufgabe zu implementieren. Um es expliziert auf die gelernte Container-Klassen Akzent zu setzen und der Vorteil von denen Einsatz zu zeigen, wird eine existierende Benutzeroberfläche von dem Projekt „QtQList“ genommen und entsprechend angepasst. Außerdem wird ein Tipp zur Lösung aus dem Studienheft verwendet, indem eine QSet Klasse, die keine Duplikate zulässt, zum Listenumwandlung eingesetzt wird. Für diesen Zweck wird ein Template definiert (siehe Listing 4.2, Vgl. Qt Dokumentation).

Als erstes werden zwei Methoden bzw. Slots in der Header-Datei deklariert (Listing 4.1) und die bereits vorhandenen Funktionen im Quellcode angepasst (Listings 4.3 und 4.4). Danach in dem Slot „on buttonRemove clicked()“ anhand eines Templates wird eine Liste vom Typ QList in eine andere Liste vom Typ QSet umgewandelt, dabei werden die Duplikate nicht mit kopiert, da diese Klasse keine zulässt. Dadurch wird eine „Remove“-Funktion stillschweigend realisiert (Listing 4.5).

```
class MainWindow : public QMainWindow
{
  ...
  //Aufgabe 4
  //Deklaration von den Slots
  void on_buttonRemove_clicked();
  void on_buttonInput_clicked();
};

```

Listing 4.1: Deklaration von den Slots für die Aufgabe 4

```
/*****************************************************************
* template <typename T> QSet<T> QListToQSet()
*
* Aufgabe 4
* Funktion: Ein Template fuer die Listenumwandlung.
*****************************************************************/
template <typename T> QSet<T> QListToQSet(const QList<T>& qlist)
{
  return QSet<T> (qlist.constBegin(), qlist.constEnd());
}

```

Listing 4.2: Ein Template für die Listenumwandlung

```
void MainWindow::on_buttonSortieren_clicked()
{
  ...
  //Aufgabe 4: die zweite Liste sortieren
  std::sort(liste1New.begin(), liste1New.end());
  listeAktualisieren();
}

```

Listing 4.3: Änderungen in der Methode für Liste Sortieren

```
void MainWindow::listeAktualisieren()
{
  ...
  //Aufgabe 4: Die zweite Liste leeren und mit den Elementen befuehlen.
  ui->listWidget_2->clear();
  foreach(int element, liste1New)
    ui->listWidget_2->addItem(QString::number(element));
  ...
}

```

Listing 4.4: Änderungen in der Methode für Liste Aktualisieren

```
/*****************************************************************
* void MainWindow::on_buttonRemove_clicked()
*
* Aufgabe 4
* Funktion: Anhand eines Templates wird eine Liste vom Typ QList in
* eine andere Liste vom Typ QSet umgewandelt, dabei werden die Duplikate
* nicht mit kopiert, da diese Classe keine zulaesst. Dadurch wird eine
* "Remove"-Funktion stillschweigend realisiert.
*****************************************************************/

void MainWindow::on_buttonRemove_clicked()
{
  QSet<int> aSet = QListToQSet(liste1);
  liste1New = aSet.values();
  ui->listWidget_2->clear();
  foreach(int element, liste1New)
    ui->listWidget_2->addItem(QString::number(element));
}

```

Listing 4.5: Eine Remove-Funktion für die Duplikate

Um die gewünschte Anzahl von den Duplikaten in der ersten Liste zu erreichen, wird eine zusätzliche Funktion, die einen Eingabe Dialog anbietet, implementiert und bei dem Aufruf eine Standard Input-Dialog Klasse verwendet (Listing 4.6). Die Abbildung (4.1) zeigt noch mal die Ergebnisse der vierten Aufgabe.

```
/*****************************************************************
* void MainWindow::on_buttonInput_clicked()
*
* Aufgabe 4
* Funktion: Hilfsfunktion.
* Ein Input-Dialog fuer die gezielte Eingabe von Duplikaten.
*****************************************************************/
void MainWindow::on_buttonInput_clicked()
{
  int valueIn;
  //den Wert ueber einen QInputDialog lesen
  valueIn = QInputDialog::getInt(this, "Eingeben",
            "Geben Sie gezielt ein Duplikat ein: ", -1);
  if (valueIn != -1)
  {
    liste1.push_back(valueIn);
    listeAktualisieren();
  }
}

```

Listing 4.6: Eine Hilfsfunktion für die gezielte Eingabe von Duplikaten

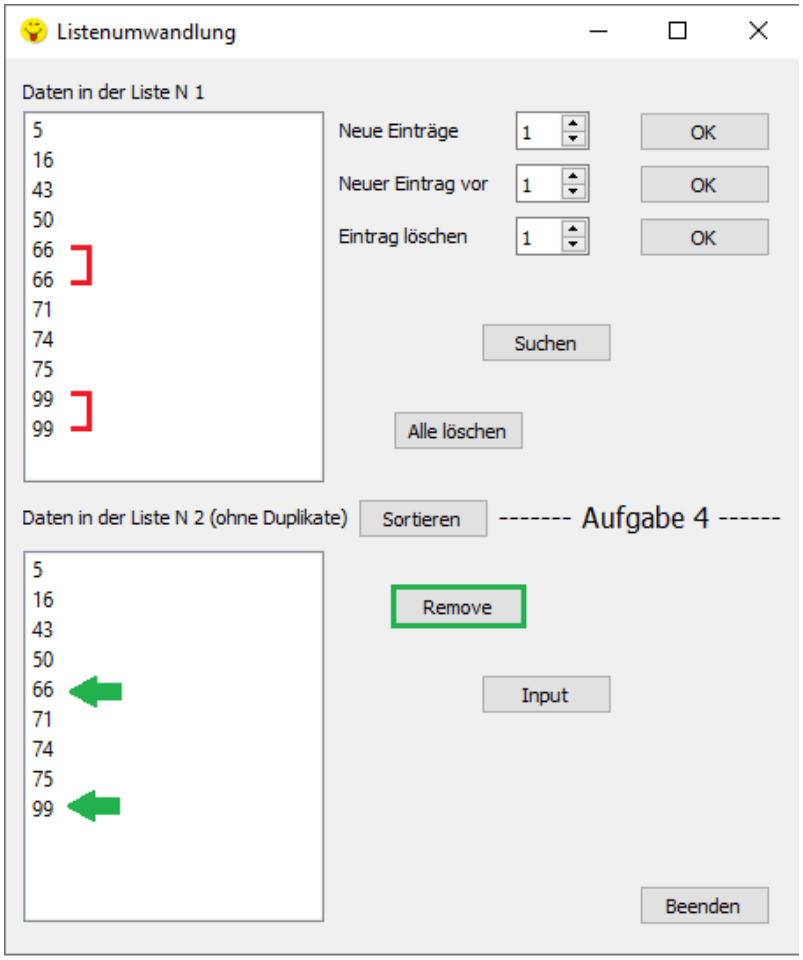

Abbildung 4.1: Ergebnis der Aufgabe 4

Abbildungsverzeichnis

Listings