```cpp
// OnlineShop.cpp: struct, enum, Datenfeld, benutzergesteuerte Schleife
//
// Der übliche Vorspann:
#include "stdafx.h"
#include <iostream>
#include <string>
using namespace std;

// Globale Vorgaben:
// Kundenstatus als Aufzählungsliste
enum Kundenstatus {Neu,Premium,Gold,Platin};

// Vielleicht auch ganz passend:
string strKundenstatus[4] = {"Neuer Kunde", "Premium-Kunde", "Gold-Kunde", "Platin-Kunde"};

// Der Kunde als Struktur:
struct Kunde {
    string Vorname;
    string Nachname;
    int Kundennummer;
    float Kaufsumme;
    Kundenstatus Status;
};

// Funktion, um den Kundenstatus im Klartext anzuzeigen:
string printStatus(Kundenstatus KdSt) {
    switch (KdSt) {
    case Neu: return "Neuer Kunde";
    case Premium: return "Premium-Kunde";
    case Gold: return "Gold-Kunde";
    case Platin: return "Platin-Kunde";
    }
}

int main()
{
    // 10000 Kunden vorsehen, 3 Daten vorgeben:
    Kunde Kundenstamm[10000] = {
        {"Max","Mustermann",123,0.00,Neu},
        {"Peter","Pan  ",234,43.25,Gold},
        {"Darth", "Vader", 345,123.45,Platin}
    };
    // Das wollen wir uns angewöhnen: Variable Daten auch als Variable anlegen
    int AnzahlKunden = 3;

    system ("cls");
    cout << "Kundenliste:\n";
    for (int KdNr=0; KdNr<AnzahlKunden; KdNr++) {
        cout << Kundenstamm[KdNr].Kundennummer << "  ";
        cout << Kundenstamm[KdNr].Vorname << " ";
        cout << Kundenstamm[KdNr].Nachname << "\t ";
        cout << printStatus(Kundenstamm[KdNr].Status) << "\t";
        cout << Kundenstamm[KdNr].Kaufsumme << endl;

    }
```

```
56      system ("pause");
57      return 0;
58  }
59
```

```cpp
// Kundenliste.cpp Aufgabe 2 (Aufgabe 1 siehe OnlineShop.cpp - ausgeblendet)
//
// Der übliche Vorspann:
#include "stdafx.h"
#include <iostream>
#include <string>
using namespace std;

// Globale Vorgaben:
// Kundenstatus als Aufzählungsliste
enum Kundenstatus {Neu,Premium,Gold,Platin};

// Vielleicht auch ganz passend:
string strKundenstatus[4] = {"Neuer Kunde", "Premium-Kunde", "Gold-Kunde", "Platin-Kunde"};

// Der Kunde als Struktur:
struct Kunde {
    string Vorname;
    string Nachname;
    int Kundennummer;
    float Kaufsumme;
    Kundenstatus Status;
};

// Variablen global vorhalten, damit man sie im ganzen Programm benutzen kann:
    // 10000 Kunden vorsehen, 3 Daten vorgeben:
    Kunde Kundenstamm[10000] = {
        {"Max","Mustermann", 123,   0.00f, Neu},
        {"Peter","Pan  ",    234,  43.25f, Gold},
        {"Darth", "Vader",   345, 123.45f, Platin}
    };
    // Das wollen wir uns angewöhnen: Variable Daten auch als Variable anlegen
    int AnzahlKunden = 3;

// Jetzt alle Funktionen, zunächst als Prototyp:
// Funktion, um den Kundenstatus im Klartext anzuzeigen:
string printStatus(Kundenstatus KdSt);

// Funktion, um die Kundenliste anzuzeigen:
void printKundenliste(int AnzahlKunden);

// Funktion, um die Kundendaten zu aktualisieren:
void aktualisiereKunde(int Kundennummer, float Kaufsumme);

// Beginn Hauptprogramm:
int main()
{
    int inKundennummer;
    float inKaufsumme;

    do { // Beginn der benutzergesteuerten Schleife
        system ("cls");
        printKundenliste(AnzahlKunden);
        cout << "Geben Sie die Kundennummer fuer eine neue Bestellung ein: ";
        cin >> inKundennummer;
```

```
56          if (inKundennummer!=0) {
57              cout << "Geben Sie die Kaufsumme fuer diese Bestellung ein: ";
58              cin >> inKaufsumme;
59              aktualisiereKunde(inKundennummer,inKaufsumme);
60          }
61      } while (inKundennummer!=0); // Ende der benutzergesteuerten Schleife
62
63      return 0;
64  }   // Ende Hauptprogramm
65
66  // Jetzt die Funktionsdefinitionen:
67  // Funktion, um den Kundenstatus im Klartext anzuzeigen:
68  string printStatus(Kundenstatus KdSt) {
69      int Status = 0;
70      switch (KdSt) {
71      case Neu:       Status = 0; break;
72      case Premium:   Status = 1; break;
73      case Gold:      Status = 2; break;
74      case Platin:    Status = 3;
75      }
76      return strKundenstatus[Status];
77  }
78
79  // Funktion, um die Kundenliste anzuzeigen:
80  void printKundenliste(int AnzahlKunden) {
81      cout << "Kundenliste:\n";
82      for (int KdNr=0; KdNr<AnzahlKunden; KdNr++) {
83          cout << Kundenstamm[KdNr].Kundennummer << "   ";
84          cout << Kundenstamm[KdNr].Vorname << " ";
85          cout << Kundenstamm[KdNr].Nachname << "\t ";
86          cout << printStatus(Kundenstamm[KdNr].Status) << "\t";
87          cout << Kundenstamm[KdNr].Kaufsumme << endl;
88      }
89      cout << endl; // eine Leerzeile
90  }
91
92  // Funktion, um die Kundendaten zu aktualisieren:
93  void aktualisiereKunde(int Kundennummer, float Kaufsumme) {
94      // Schritt 1: den Kunden aus der Liste suchen:
95      int KundeGefunden = 0;
96      for (int KdNr=0; KdNr<AnzahlKunden; KdNr++) {
97          if (Kundenstamm[KdNr].Kundennummer == Kundennummer)
98              KundeGefunden = KdNr;
99      }
100     // Schritt 2: Kaufsumme erhöhen, wenn Kunde gefunden wurde:
101     if (KundeGefunden>=0) {
102         Kundenstamm[KundeGefunden].Kaufsumme += Kaufsumme;
103         // Schritt 3: Kundenstatus aktualisieren:
104         if (Kaufsumme >= 5.0) {
105             // wir müssen explizite Typumwandlungen vornehmen,
106             // weil der Status ja ein enum ist und kein Integer...
107             int derKundenStatus = int(Kundenstamm[KundeGefunden].Status);
108             if (derKundenStatus < 4)
109                 derKundenStatus++;
110             Kundenstamm[KundeGefunden].Status = Kundenstatus(derKundenStatus);
111         }
```

```
112     } else {
113         cout << "Kunde " << Kundennummer << " ist nicht vorhanden!\n";
114         system("pause");
115     }
116 }
```

```cpp
// Kundenklasse.cpp : Aufgabe 3: jetzt als Klasse
// Aufgabe 2 siehe Kundenliste.cpp / Aufgabe 1 siehe OnlineShop.cpp - im Projekt OnlineShop)
//
// Der übliche Vorspann:
#include "stdafx.h"
#include <iostream>
#include <string>
using namespace std;

// Globale Vorgaben:
// Kundenstatus als Aufzählungsliste
enum Kundenstatus {Neu,Premium,Gold,Platin};

// Vielleicht auch ganz passend:
string strKundenstatus[4] = {"Neuer Kunde", "Premium-Kunde", "Gold-Kunde", "Platin-Kunde"};

// Der Kunde als Struktur:
class Kunde {
public:
    string Vorname;
    string Nachname;
    int Kundennummer;
    float Kaufsumme;
    Kundenstatus Status;
};

// Variablen global vorhalten, damit man sie im ganzen Programm benutzen kann:
    // 10000 Kunden vorsehen, 3 Daten vorgeben:
    Kunde Kundenstamm[10000] = {
        {"Max","Mustermann", 123,   0.00f, Neu},
        {"Peter","Pan  ",    234,  43.25f, Gold},
        {"Darth", "Vader",   345, 123.45f, Platin}
    };
    // Das wollen wir uns angewöhnen: Variable Daten auch als Variable anlegen
    int AnzahlKunden = 3;

// Jetzt alle Funktionen, zunächst als Prototyp:
// Funktion, um den Kundenstatus im Klartext anzuzeigen:
string printStatus(Kundenstatus KdSt);

// Funktion, um die Kundenliste anzuzeigen:
void printKundenliste(int AnzahlKunden);

// Funktion, um die Kundendaten zu aktualisieren:
void aktualisiereKunde(int Kundennummer, float Kaufsumme);

// Beginn Hauptprogramm:
int main()
{
    int inKundennummer;
    float inKaufsumme;

    do { // Beginn der benutzergesteuerten Schleife
        system ("cls");
```

```
55          printKundenliste(AnzahlKunden);
56          cout << "Geben Sie die Kundennummer fuer eine neue Bestellung ein: ";
57          cin >> inKundennummer;
58          if (inKundennummer!=0) {
59              cout << "Geben Sie die Kaufsumme fuer diese Bestellung ein: ";
60              cin >> inKaufsumme;
61              aktualisiereKunde(inKundennummer,inKaufsumme);
62          }
63      } while (inKundennummer!=0); // Ende der benutzergesteuerten Schleife
64
65      return 0;
66  }   // Ende Hauptprogramm
67
68  // Jetzt die Funktionsdefinitionen:
69  // Funktion, um den Kundenstatus im Klartext anzuzeigen:
70  string printStatus(Kundenstatus KdSt) {
71      int Status = 0;
72      switch (KdSt) {
73      case Neu:       Status = 0; break;
74      case Premium:   Status = 1; break;
75      case Gold:      Status = 2; break;
76      case Platin:    Status = 3;
77      }
78      return strKundenstatus[Status];
79  }
80
81  // Funktion, um die Kundenliste anzuzeigen:
82  void printKundenliste(int AnzahlKunden) {
83      cout << "Kundenliste:\n";
84      for (int KdNr=0; KdNr<AnzahlKunden; KdNr++) {
85          cout << Kundenstamm[KdNr].Kundennummer << "  ";
86          cout << Kundenstamm[KdNr].Vorname << " ";
87          cout << Kundenstamm[KdNr].Nachname << "\t ";
88          cout << printStatus(Kundenstamm[KdNr].Status) << "\t";
89          cout << Kundenstamm[KdNr].Kaufsumme << endl;
90      }
91      cout << endl; // eine Leerzeile
92  }
93
94  // Funktion, um die Kundendaten zu aktualisieren:
95  void aktualisiereKunde(int Kundennummer, float Kaufsumme) {
96      // Schritt 1: den Kunden aus der Liste suchen:
97      int KundeGefunden = -1;
98      for (int KdNr=0; KdNr<AnzahlKunden; KdNr++) {
99          if (Kundenstamm[KdNr].Kundennummer == Kundennummer)
100             KundeGefunden = KdNr;
101     }
102     // Schritt 2: Kaufsumme erhöhen, wenn Kunde gefunden wurde:
103     if (KundeGefunden>=0) {
104         Kundenstamm[KundeGefunden].Kaufsumme += Kaufsumme;
105         // Schritt 3: Kundenstatus aktualisieren:
106         if (Kaufsumme >= 5.0) {
107             // wir müssen explizite Typumwandlungen vornehmen,
108             // weil der Status ja ein enum ist und kein Integer...
109             int derKundenStatus = int(Kundenstamm[KundeGefunden].Status);
110             if (derKundenStatus < 4)
```

```
111                 derKundenStatus++;
112             Kundenstamm[KundeGefunden].Status = Kundenstatus(derKundenStatus);
113         }
114     } else {
115         cout << "Kunde " << Kundennummer << " ist nicht vorhanden!\n";
116         system("pause");
117     }
118 }
```