Programmieren in C++ WiSe 22/23

Prog2ETWS2021

Hinweise:

- Die Klausur umfasst 6 Blätter mit 5 Aufgaben: Vergewissern Sie sich bitte zuerst, dass Sie alle Blätter erhalten haben.
- Geben Sie auf allen Lösungsblättern Ihren Namen, das Semester und die Matrikelnummer an.
- Falls Sie zusätzliches Papier benötigen, melden Sie sich bitte.
- Schreiben Sie bitte deutlich, sauber, leserlich und mit einem dokumentenechten Schreibgerät, also nicht mit Bleistift.
- Benutzen Sie keine rote Farbe.
- Streichen Sie Ungültiges deutlich und eindeutig durch. Verwenden Sie hierzu kein Tipp-Ex, Korrekturband u. ä.
- Bieten Sie keine mehrfachen Lösungen an, sondern entscheiden Sie sich für eine und streichen Sie die anderen durch.
- Die zu verwendende Programmiersprache ist C bzw. C++. Präprozessor-Anweisungen, wie z. B. **`#include`**, sind in der Lösung nicht erforderlich. Namensräume können als automatisch verfügbar angesehen werden. Wenn nicht anders angegeben, können Programmausgaben mit **`cout`** oder mit **`printf`** durchgeführt werden. In Bezeichnern sowie Ein- und Ausgaben können Umlaute und ß ohne besondere Maßnahmen verwendet werden. **`main()`** kann ohne Rückgabetyp und **`return`** notiert werden.
- Halten Sie sich streng an vorgegebene Funktions- und Methodensignaturen. Definieren Sie Klassen jeweils mit dem Code für getrennte Kopf- und Implementierungsdateien.
- Viel Erfolg!

---

1. Aufgabe (20 Punkte): Dynamische Speicherallokation und Templates:

a. Gegeben sei der Kopf einer Funktion `arrayConcat`:

```
int* arrayConcat(int* a1, int length1, int* a2, int length2)
```

Die Funktion kopiert die beiden Arrays in einen neuen Array der Größe length1+length2 und gibt diesen wieder zurück. a1 ist damit an erster Position und a2 an 2. Position. Schreiben Sie die Funktion: `arrayConcat`.

Hinweise:

a1 = {1,2,3} und a2 = {4,5,6,7} würde damit den neuen Array: {1,2,3,4,5,6,7} ergeben.

Ob Sie C oder C++ verwenden ist Ihnen überlassen.

b. Änderen Sie die Funktion `arrayConcat` in ein Funktionstemplate so, dass beliebige Typen übergeben werden können.

2. Aufgabe (20 Punkte): C++ Strukturen und Listen

Gegeben sei die folgende C++ Struktur für eine verkettete Liste von Studenten. Diese beinhaltet einen Namen als String, eine Matrikelnummer als Integer-Zahl, sowie einen Pointer auf den nächsten Eintrag:

```
struct Student
{
    string Name;
    int Matrikelnummer;
    Student* next;
};
```

Weiterhin gegeben ist eine Liste von Studenten. Diese besteht aus dem Kopf der Studentenliste als Pointer auf einen Student, sowie aus der Funktion `insert` und `printList`:

```
struct StudentList
{
    Student * list = 0;
    void insert(string Name, int Matrikelnummer);
    void printList(void);
};
```

Die Funktion `insert` soll einen neuen Studenten mit Name und Matrikelnummer anlegen und diesen an den Kopf der Liste schreiben. Die Funktion `printList` soll alle Elemente der Liste ausgeben. Am Ende der Liste oder bei einer leeren Liste soll sie EOL (End of List) ausgeben.

Beispiel: Der Programmcode in der main-Methode ist wie folgt gestaltet:

```
    StudentList List;
    List.insert("Nagel",100001);
    List.insert("Mueller",100002);
    List.insert("Maier",100003);
    List.printList();
```

Dieser Programmcode soll die folgende Ausgabe hervorrufen:

```
Name: Maier Matrikelnummer: 100003
Name: Mueller Matrikelnummer: 100002
Name: Nagel Matrikelnummer: 100001
EOL
```

a. Implementieren Sie die Methode `insert`.

b. Implementieren Sie die Methode `printList`.

3. Aufgabe (20 Punkte): C++ Klassen

Eine Bank speichert die Pin und den Namen Ihrer Kunden und zahlt bei verfügbaren Guthaben und korrekter Authentifizierung Bargeld aus. Dieses Verhalten soll programmiert werden.

a. Erstellen Sie eine Klasse `Kunde`. In dieser Klasse sollen folgende Informationen öffentlich zugänglich gespeichert werden: der Name als String, die Pin als Integer und der Kontostand als Integer. Diese Klasse verfügt über keine spezielle Methode, dem Konstruktor sollen allerdings alle drei Parameter übergeben werden, damit diese initial angelegt werden.

b. Erstellen Sie eine weitere Klasse `Bank`. Der Einfachheit halber hat diese Bank nur einen einzigen Kunden. Dieser wird als nicht öffentlich zugänglichen Member angelegt. Die Bank hat zwei öffentlich zugängliche Methoden:

   ```
   Bank(string Kunde, int Pin, int Kontostand);
   ```
   Dies ist der Konstruktor, der intern den Kunden anlegt.

   ```
   bool auszahlung(string Kunde, int Pin, int Betrag);
   ```
   Die Methode `auszahlung` überprüft den korrekten Namen und Pin. Ist der gewünschte Betrag dann nicht höher als der Kontostand, wird das Geld in Form einer Textausgabe ausgezahlt und der Kontostand entsprechend angepasst. Die Methode liefert in diesem Fall ein `true` zurück. Bei jeglicher fehlerhafter Eingabe gibt die Methode ein `false` und eine Meldung auf der Textausgabe zurück. Fehlerhafte Eingaben sind zum Beispiel: ein Name, der nicht existiert, eine Pin, die nicht zum Namen passt, sowie ein Auszahlungsbetrag der größer ist als der Kontostand. Als Fehlermeldung reicht hier lediglich: "Falsche Eingabe"

Das folgende Hauptprogramm soll die unten dargestellte Ausgabe ausgeben:

```
Bank bank("Nagel", 1234, 1000);
bank.auszahlung("Nagel",1111,200);
bank.auszahlung("Nagel",1234,100);
bank.auszahlung("Nagel",1234,200);
```

Ausgabe:

```
Falsche Eingabe
Ihr neuer Kontostand betraegt: 900 Euro
Ihr neuer Kontostand betraegt: 700 Euro
```

4. Aufgabe (20 Punkte): Vererbung und Polymorphie

Wir implementieren eine Verkehrszählung. Die Verkehrszählung unterscheidet dabei ob ein Motorrad oder ein Automobil gezählt wurden.

a. Implementieren Sie ein Oberklasse `Fahrzeug`. Diese Klasse ist eine rein abstrakte Schnittstellenvorgabe. Der Name der Schnittstelle ist `identify`. Der Rückgabewert ist ein `char`. Diese Methode benötigt keinen Parameter.

b. Implementieren Sie eine Klasse `Automobil`, welche sich aus dem `Fahrzeug` vererbt. Das `Automobil` soll sich über die `identify`-Methode identifizieren, indem es ein ‚`a`' zurückgibt.

c. Implementieren Sie eine Klasse `Motorrad`, welche sich aus dem Fahrzeug vererbt. Das `Motorrad` soll sich über die `identify`-Methode identifizieren, indem es ein ‚`m`' zurückgibt.

d. Implementieren Sie eine Klasse `Verkehrszaehler`. Diese besitzt, neben einem Konstruktor zur Initialisierung der Member, zwei nicht öffentliche Member zur Zählung von `Motorrädern` und `Automobilen`. Weiterhin besitzt sie eine Methode:
```
bool Zaehle(Fahrzeug * fahrzeug);
```
Diese inkrementiert den jeweiligen Zähler und gibt ein `true` zurück. Sollte weder ein `Motorrad`, noch ein `Automobil` erkannt worden sein, gibt die Methode ein `false` zurück.

e. Implementieren Sie ein Hauptprogramm, welches jeweils ein Objekt vom Typ `Motorrad`, `Automobil` und `Verkehrszaehler` erzeugt. Danach soll zuerst das `Automobil` und dann das `Motorrad` gezählt werden.

5. Aufgabe (20 Punkte): Grafische Bedienoberfläche

Gegeben sei die in Abbildung 1 und Abbildung 2 dargestellte Oberfläche eines QT-Programmes. In Abbildung 1 ist der Programmstart angezeigt: Der Button ist nicht aktiv und die beiden Checkboxes sind nicht ausgewählt. Erst wenn beide Checkboxes gewählt sind, aktiviert sich der Button (Abbildung 2). Auf Druck des Buttons soll dann wieder der Anfangszustand (Abbildung 1) hergestellt werden.

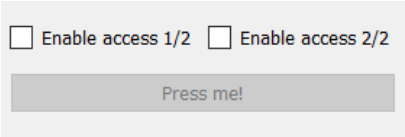

Abbildung 1: Programmstart und Status nach Druck auf „Press me!".

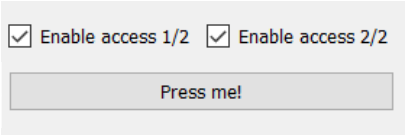

Abbildung 2: Aktivierung des Buttons durch Auswahl beider Checkboxes.

Gegeben sei auch der Inhalt der Headerdatei mainwindow.h:
class **MainWindow** : public QMainWindow

```
{
    Q_OBJECT
public:
    MainWindow(QWidget *parent = nullptr);
    ~MainWindow();
private:
    QWidget centralWidget;
    QHBoxLayout hlayout;
    QVBoxLayout vlayout;
    QPushButton b;
    QCheckBox check1,check2;
public slots:
    void checkHandler(void);
    void buttonHandler(void);
};
```

Sowie die unvollständige Implementierungsdatei:

```
MainWindow::MainWindow(QWidget *parent)
    : QMainWindow(parent)
{
    // Zu ergänzender Code
}
void MainWindow::checkHandler()
{
    // Zu ergänzender Code
}
void MainWindow::buttonHandler()
{
    // Zu ergänzender Code
}
```

Schreiben Sie den Programmcode auf, um den die Implementierungsdatei zu ergänzen ist, damit das Programm in der oben beschriebenen Form funktioniert. Setzen Sie die Bedienoberfläche programmatisch um. Nutzen Sie für die Platzierung der Widgets den Layout-Manager.

Hinweis: Die folgenden Methoden für die Klasse QCheckBox aus der QT-Dokumentation, können Ihnen hier helfen:

checked : bool

This property holds whether the button is checked

Only checkable buttons can be checked. By default, the button is unchecked.

Access functions:

bool	**isChecked**() const
void	**setChecked**(*bool*)

Zuletzt geändert: Dienstag, 28. Juni 2022, 13:11

◄ ProgC++SS2022

Direkt zu:

LsgKlausurWS2021 ►