Hochschule Furtwangen
Fakultät Informatik
Prof. Dr. Lothar Piepmeyer

Praktikum ‚Objektorientierte Programmierung'

Aufgabenblatt 4

Aufgabe 1:
In dieser Ausgabe sollen Sie das folgende Fragment einer Klasse in einigen Teilaufgaben um Funktionalität ergänzen:

```
class Friends{
private:
    const std::string* names;
    int size;
public:
    ...
}
```

a. Ergänzen Sie die Klasse `Friends` um einen sinnvollen Konstruktor, dem man ein Array mit Objekten vom Typ `string` sowie die Länge dieses Arrays als Parameter übergeben kann. Beide Attribute werden mit Hilfe der Parameter des Konstruktors geeignet initialisiert. Der Rumpf des Konstruktors darf keine Anweisungen enthalten. Insbesondere soll das folgende Code-Fragment übersetzt und ohne Laufzeitfehler ausgeführt werden:

```
std::string names[2]={"Donald", "Daisy"};
Friends friends(names, 2);
```

Formulieren Sie weitere Tests.

b. Objekte vom Typ Friends sollen sich auch erzeugen lassen, ohne dass eine Konstruktor mit Parametern aufgerufen wird. Ergreifen Sie Maßnahmen, damit der folgende Code übersetzt wird:

```
Friends friends
```

Das Array `names` soll den Wert `nullptr` und das Attribut `size` den Wert 0 enthalten. In C++ ist `nullptr` vordefiniert und wird für unbekannte Zeiger genutzt. In modernem C++ ist die Verwendung von `NULL` unüblich. Formulieren Sie Tests.

c. Ergänzen Sie eine Methode

```
const std::string& name(int v)
```

die den Namen des Freundes ermittelt, der in der Liste an Platz `v` steht. Das folgende Code-Fragment soll also fehlerfrei ausgeführt werden:

```
std::string names[2]={"Donald", "Daisy"};
Friends friends(names, 2);
assert(friends.name(0)=="Donald");
```

Formulieren Sie Tests.

d. Es gibt Parameter, für die der Konstruktor und die `names`-Methode ihren Vertrag nicht erfüllen können. Ergreifen Sie geeignete Maßnahmen. Formulieren Sie Tests. Die Entwicklung umfassender Tests ist vermutlich der Hauptanteil dieses Aufgabenteils.

e. Machen Sie sich klar, dass Anwender von `Friends` erwarten, dass der folgende Code fehlerfrei läuft:

```
std::string names[2]={"Donald", "Daisy"};
Friends friends(names, 2);
assert(friends.name (0)=="Donald");
names[0]="Mickey";
assert(friends.name(0)=="Donald");
```

Leistet Ihre Implementierung von `Friends` das auch? Möglicherweise müssen Sie Ihren Konstruktor anpassen. Formulieren Sie Tests.

Aufgabe 2:
Ersetzen Sie in Ihren Klassen `Date` und `Person` vom vorhergehenden Aufgabenblatt die init-Methoden durch geeignete Konstruktoren. Die Konstruktoren sollen auch hier mit Initialisiererlisten arbeiten. Die Rümpfe der Konstruktoren bleiben leer. Das hört sich einfacher an als es ist: Exceptions müssen nach wie vor geworfen werden, wenn der Konstruktor seinen Vertrag nicht erfüllen kann. Passen Sie die Tests vom letzten Aufgabenblatt an.