Testaufgabe

Für ein Navigationsgerät ist ein Container zu entwerfen, in dem vom Benutzer eingegebene Wegepunkte abgespeichert werden können.

Der Container wird innerhalb einer statischen Bibliothek realisiert und ist im Namespace **HHN** zu entwerfen!

1) Die Klasse *WayPoint*

Zur Repräsentation eines Wegepunktes erstellen wir eine eigene Klasse ***WayPoint***.

```
class WayPoint
	{
	private:
		std::string name{ "" };
		std::pair<double, double> coords{ 0.0, 0.0 };
	public:
		WayPoint() {};
		WayPoint(const WayPoint& orig);
		WayPoint(const std::string& name, double xCoord, double yCoord);

		WayPoint& operator=(const WayPoint& rhs);

		std::string Name() const;
		double first() const;
		double second() const;
	};
```

Wie aus der Deklaration ersichtlich, sind für die Klasse ***WayPoint*** u.a. ein Copy Konstruktor und ein Zuweisungsoperator zu erstellen. Der Standard Konstruktor wird im Rahmen des deklarativen Teils der Klasse mit leerem Funktionskörper definiert, damit er zur Laufzeit zur Verfügung steht.
Die Klasse verwendet zudem aus der Standardbibliothek ***std::pair***. Schnittstelle und Verwendung können leicht im Internet (z.B. www.cplusplus.com) nachgeschlagen werden.

2) Die Klasse *WayPointContainer*

Der eigentliche Container wird in der Klasse ***WayPointContainer*** realisert.

```
class WayPointContainer
	{
	private:
		std::vector<HHN::WayPoint>* pContainer{ nullptr };

	public:
		WayPointContainer();
		WayPointContainer(const WayPointContainer& orig);
		virtual ~WayPointContainer();

		WayPointContainer& operator=(const WayPointContainer& rhs);
		WayPoint& operator[](int idx) const;

		void Add(const WayPoint& arg);

		int Size() const;
		void Print() const;
	};
```

Um die einzelnen Wegepunkte im Attributbereich zu halten, wird aus der Standardbibliothek der vorgefertigte Container ***std::vector*** verwendet.
Dieser Vektor wird als Zeiger deklariert und mit ***nullptr*** vorbelegt (Standardinitialisierung). Er ist im Default Konstruktor mit einem Heap Bereich zu verbinden (***new*** Operator). Nach dem bekannten Prinzip ***RAII*** (Resource Acquisition Is Initialization) muss dieser Speicher im Destruktor der Klasse wieder an das Betriebssystem zurückgegeben werden. Dabei übernimmt der Destruktor des Containers ***std::vector*** auch das Aufräumen der Elemente, die sich im Container befinden (soweit notwendig).
Im Copy Konstruktor verwenden wir den Default Kontruktor, um dieses Heap Stück zu generieren, indem wir das C++11 Feature der Delegation von Konstruktoren verwenden:

Beispiel:

```
class A {
    int x, y, z;

public:
    A()
    { ... }

    // Constructor delegation
    A(int z) : A()
    { ... }
```

3) Prüfungen am Code

Zum Überprüfen Ihres Codes schreiben Sie sich ein Programm, in dem Sie u.a. die folgenden Dinge überprüfen sollten:

- Füllen Sie einen Container mit einer Anzahl von Wegepunkten
- Kopieren Sie den Container mit Hilfe des Copy Konstruktors und zerstören Sie danach das Original
- Benutzen Sie den Zugriffsoperator, um eine zweite Kopie zu erzeugen

Zwischen den einzelnen Schritten geben Sie den Inhalt der Container auf der Konsole aus, um zu überprüfen, ob Ihre instanziierten Objekte noch korrekt verfügbar sind.