9 Klassen

9.1 Übungsaufgabe

Schreiben Sie ein C++ Programm in eine Datei. Teilen Sie das Programm nicht wie sonst in Headerdatei, Implementationsdatei und Hauptprogramm auf.

Designen Sie im Folgenden die Klasse `Cis_wstring`:
Die Klasse `Cis_wstring` soll in der Lage sein, einen wstring intern in einem Array von wchar_t (Buchstaben-Memory) zu speichern. Die Klasse soll öffentliche Methoden haben, die die Zeichenkette in das interne Buchstaben-Memory eintragen und den Inhalt des Buchstaben-Memories ausgeben können. Außerdem soll der Benutzer in der Lage sein, abzufragen wieviele Buchstaben im Buchstaben-Memory sind.

Die Klasse `Cis_wstring` benötigt mindestens folgende public Komponenten:

- `int get_anzahl_chars()`: liefert die Anzahl der Buchstaben der `Cis_wstring` Daten
- `void get_cis_wstring_data(wstring &text)`: liefert als wstring in den Inhalt wchar_t Daten aus `data[132]`
- `void set_cis_wstring_data(wstring text)`: trägt den String text in die `Cis_wstring` Daten ein.

Die Klasse `Cis_wstring` benötigt mindestens folgende private Komponenten:

- `wchar_t data[132]`: alle Buchstaben der `Cis_wstring` Daten
- `int anzahl`: Anzahl der Buchstaben in `data`
- `int get_anzahl()`: liefert die Anzahl der Buchstaben der `Cis_wstring` Daten

Schreiben Sie public-member Funktionen für die Klasse `Cis_wstring`, die folgende Aktionen auf einem Objekt der Klasse `Cis_wstring` ausführen.

- Eine Konstruktorfunktion, die alle Daten des `Cis_wstring` auf ein SPACE setzen.
- Eine Konstruktorfunktion, die das Argument in den Buchstabenspeicher des `Cis_wstring` übertragen.

- Eine Setterfunktion `set_cis_wstring_data(wstring text)`, die das wstring Argument in den Buchstabenspeicher des `Cis_wstring` einträgt.
- Eine `void cis_wstring_to_upper()` Funktion, die die Daten des `Cis_wstring` in Großbuchstaben konvertiert.
- Eine `bool first_is_upper()` Funktion, die wahr/falsch liefert, je nachdem der erste Buchstaben des `Cis_wstring` ein Großbuchstabe ist (Verwenden Sie Libraryfunktionen zum Testen)
- Eine Getterfunktion `get_cis_wstring_data(wstring &text)`, die den Inhalt der `Cis_wstring` Daten in einem wstring `&text` speichern.

Nun soll folgende main() Funktion möglich sein:

```
int main()
{
   // Setzen des locale auf utf-8  (Verwenden Sie locale etc...)
   ...

   wstring eingabe;
   wstring ausgabe;
   Cis_wstring text;
   Cis_wstring moeller(L"Wie geht es Herrn Möller?");

   wcout << "Bitte geben sie einen String ein :";
   wcin >> eingabe;
   text.set_cis_wstring_data(eingabe);
   // eingabe in den Cis_wstring text eintragen
   wcout << "Die Anzahl der Buchstaben ist : "<<
   text.get_anzahl_chars() << endl;

   // test ob der erste Buchstabe groß ist:
   if (text.first_is_upper()){
      wcout << "Der erste Buchstabe des Strings ist ein
      großer Buchstabe" << endl;

      // Konvertieren auf upper:
      text.cis_wstring_to_upper();
      text.get_cis_wstring_data(ausgabe);
      // holt aus dem Cis_wstring text den wstring ...
      wcout << "TOUPPER: Der interne String von Cis_wstring ist : "<<
      ausgabe << endl;
   }
}
```

9.2 Wiederholung und zusätzliche Informationen

9.2.1 Klassen

Klassen fassen Daten und Funktionen zusammen und sind ein wichtiger Teil der Objektorientierten Programmierung. Klassen bestehen üblicherweise aus zwei Dateien, der Deklarations- und Implementationsdatei.

Bei der Deklaration einer Klasse werden die Daten und Funktionen der Klasse festgelegt. Diese können entweder `public` oder `private` sein. Auf `public` Daten und Funktionen kann von außerhalb der Klasse zugegriffen werden. Auf `private` Daten und Funktionen kann nur von innerhalb der Klasse zugegriffen werden.

Eine Klasse enthält Variablen, Funktionen und Konstruktoren. Konstruktoren werden beim Erzeugen eines Objekts der Klasse aufgerufen und initialisieren meist bestimmte Variablen. Wird kein eigener Konstruktor definiert, so nutzt C++ den Default-Konstruktor. Bei diesem wird ein Objekt der Klasse ohne Argumente erzeugt. Falls die übergebenen Argumente im Konstruktor den gleichen Namen tragen wie die Variablen der Klasse, so kann über `this->variablenname` die Variable der Klasse angesprochen werden.

Bei der Implementation der Klasse ist es wichtig jeder Funktion mitzuteilen, zu welcher Klasse sie gehört. Die geschieht durch das vorgestellt „Klassenname::“ vor den Funktionsnamen.

Klassendeklarationen beginnen mit `#ifndef KLASSENNAME_HPP` und `#define KLASSENNAME_HPP` und enden mit `#endif`.

Im Folgenden wurde das Prinzip einer Klasse am Beispiel „Mensch“ verdeutlicht.
Datei Mensch.hpp:

```
//Anfang der Klasse Mensch
#ifndef MENSCH_HPP
#define MENSCH_HPP
#include <iostream>
using namespace std;

//Klassendeklaration
class Mensch {
   //öffentliche Daten und Funktionen (auf diese kann von
   //außerhalb der Klasse zugegriffen werden
   public:
      wstring nachname;
      wstring vorname;
      wstring wohnort;
      void zieheUm(wstring neuerWohnort);
      bool altGenug(int benoetigtesAlter);
      Mensch(wstring nachname, wstring vorname, wstring wohnort, int alter);
```

```
    //private Daten und Funktionen (auf diese kann nur innerhalb der Klasse
    //zugegriffen werden
    private:
        int alter;
        int wieAlt();
};

//Konstruktor: so kann ein Objekt der Klasse erstellt werden
Mensch::Mensch(wstring nachname, wstring vorname, wstring wohnort, int alter) {
    this->nachname = nachname;
    this->vorname = vorname;
    this->wohnort = wohnort;
    this->alter = alter;
};

//Implementation der Methode zieheUm
void Mensch::zieheUm(wstring neuerWohnort) {
    wohnort = neuerWohnort;
};

//Implementation der Methode altGenug
bool Mensch::altGenug(int benoetigtesAlter) {
    if (wieAlt() >= benoetigtesAlter) {
        return true;
    }
    else {
        return false;
    }
};

//Implementation der Methode wieAlt
int Mensch::wieAlt() {
    return alter;
};

//Ende der Klasse Mensch
#endif
```

Datei MainMensch.cpp:

```
#include<iostream>
#include "Mensch.hpp"
using namespace std;

int main() {
    //zwei Objekte der Klasse Mensch werden erstellt
    Mensch susi(L"Maier", L"Susi", L"Berlin", 25);
    Mensch franz(L"Maier", L"Franz", L"Berlin",11);

    //Wohnorte der Objekte werden ausgegeben --> auf public kann direkt
    //zugegriffen werden
```

```
    wcout<<L"Susi wohnt in " << susi.wohnort<<endl;
    wcout<<L"Franz wohnt in " << franz.wohnort<<endl;

    //auf private Daten kann nicht direkt zugegriffen werden: FEHLER
     wcout<<L"Susi ist so alt: " << susi.alter << endl;

    //Abfrage: wenn das Objekt alt genug ist wird der Wohnort geändert
    if(susi.altGenug(18)){
       susi.zieheUm(L"Muenchen");
       wcout<<L"Susi ist alt genug und wohnt jetzt in " <<
       susi.wohnort<<endl;
    }
    else{
       wcout<<L"Susi ist nicht alt genug um umzuziehen"<<endl;
    }

    if(franz.altGenug(18)){
       franz.zieheUm(L"Muenchen");
       wcout<<L"Franz ist alt genug und wohnt jetzt in " <<
       franz.wohnort<<endl;
    }
    else{
       wcout<<L"Franz ist nicht alt genug um umzuziehen"<<endl;
    }
}
```

9.3 Lösungsvorschlag

```
/*Autor: Nicola Greth
 * Uebung 8
 * Klasse Cis_wstring
 */
#ifndef CISWSTRING_HPP
#define CISWSTRING_HPP
#include <iostream>
using namespace std;

//Klassendeklaration
class Cis_wstring {
   //public Komponenten
   public:
      int get_anzahl_chars();
      void get_cis_wstring_data(wstring &text);
      void set_cis_wstring_data(wstring text);
      Cis_wstring();
      Cis_wstring(wstring eingabe);
      void cis_wstring_to_upper();
      bool first_is_upper();
```

```
    //private Komponenten
    private:
        wchar_t data[132];
        int anzahl;
        int get_anzahl();
};

//Konstruktor, der alle Daten auf ein Space setzt
Cis_wstring::Cis_wstring() {
    anzahl = 0;

    for (int i = 0; i < 132; i++) {
        data[i] = L' ';
    }
};

//Konstruktor, der ein Argument in den Buchstabenspeicher überträgt
Cis_wstring::Cis_wstring(wstring eingabe) {
    anzahl = eingabe.size();

    for (int i = 0; i < anzahl; i++) {
        data[i] = eingabe.at(i);
    }
};

//liefert die Anzahl der Buchstaben der Cis_wstring Daten
int Cis_wstring::get_anzahl_chars() {
    return get_anzahl();
};

//liefert als wstring den Inhalt wchar_t Daten aus data[132]
void Cis_wstring::get_cis_wstring_data(wstring &text) {
    text.clear();

    for (int i = 0; i < anzahl; i++) {
        text = text + data[i];
    }
};

//trägt den String text in die Cis_wstring Daten ein = Konstruktor mit Argument
void Cis_wstring::set_cis_wstring_data(wstring text) {
    anzahl = text.size();

    for (int i = 0; i < anzahl; i++) {
        data[i] = text.at(i);
    }
};

//liefert die Anzahl der Buchstaben der Cis_wstring Daten (private)
int Cis_wstring::get_anzahl() {
    return anzahl;
```

```
};

//konvertiert die Daten des Cis_wstring in Großbuchstaben
void Cis_wstring::cis_wstring_to_upper() {
   for (int i = 0; i < anzahl; i++) {
      data[i] = towupper(data[i]);
   }
};

bool Cis_wstring::first_is_upper() {
   if (iswupper(data[0])) {
      return true;
   }
   return false;
};

//Ende der Klassendeklaration
#endif

//Anfang des Hauptprogramms
int main() {

   // Setzen des locale auf utf-8
   setlocale(LC_ALL, "de_DE.utf-8");

   wstring eingabe;
   wstring ausgabe;
   Cis_wstring text;
   Cis_wstring moeller(L"Wie geht es Herrn Möller?");
   wcout << L"Bitte geben sie einen String ein :";
   wcin >> eingabe;

   text.set_cis_wstring_data(eingabe); // eingabe in den Cis_wstring text
   //eintragen
   wcout << L"Die Anzahl der Buchstaben ist : " <<
   text.get_anzahl_chars() << endl;

   // test ob der erste Buchstabe groß ist:
   if (text.first_is_upper()) {
      wcout << L"Der erste Buchstabe des Strings ist ein großer Buchstabe"
      << endl;

      // Konvertieren auf upper:
      text.cis_wstring_to_upper();
      text.get_cis_wstring_data(ausgabe); // holt aus dem Cis_wstring text
      //den wstring ...
      wcout << L"TOUPPER: Der interne String von Cis_wstring ist : " <<
      ausgabe << endl;
   }
}
```