Praktikumstermin Nr. 10, INF: Klassen, Templates

Abgabe im GIP-INF Praktikum der Woche 12.12.-16.12.2022.

2022-12-08: "Mocking" aus dem Titel entfernt, da es in diesem Praktikum nicht behandelt wird

(Pflicht-) Aufgabe INF-10.01: Klasse `MyCanvas`

Ein Canvas ist eine Zeichenfläche / Bildschirmfläche mit x- und y-Koordinaten. Der Canvas in dieser Aufgabe ist ein „Text-Canvas", d.h. es wird ein `char` Zeichen an jeder `x,y`-Position gespeichert und der Canvas-Inhalt kann auf `cout (stdout)` ausgegeben werden.

Der Canvas `MyCanvas` in dieser Aufgabe speichert seine `char` Werte in einem eindimensionalen (!) Array, d.h. die Koordinate `[x][y]` wird auf die Arrayposition `[ y * size_x + x ]` abgebildet (und umgekehrt). Eine Abweichung von diesem Prinzip (also insbesondere die Nutzung eines zweidimensionalen Arrays) ist nicht gestattet! Das Array zum Canvas soll auf dem Heap per „Array-new" `new[]` angelegt werden und das `MyCanvas` Objekt speichert einen Pointer auf die Startadresse dieses Arrays auf dem Heap.

Legen Sie in Visual Studio Code ein neues leeres Projekt an und innerhalb dieses Projekts dann leere Dateien `MyCanvas.h`, `MyCanvas.cpp`, `main.cpp`.

Programmieren Sie in den entsprechenden Dateien die Klasse `MyCanvas` gemäß den folgenden Anforderungen (dabei alle (!) Methoden in der `.cpp` Datei programmieren, in die Klassendefinition in der Headerdatei nur die Prototypen):

Jedes Objekt der Klasse `MyCanvas` besitze zwei `unsigned int` Attribute `size_x` und `size_y` für die x- und y-Größe der Zeichenfläche (canvas) (Positionen `0` bis `size_x - 1` und `0` bis `size_y - 1`) sowie ein `char*` Attribut `canvas_array_ptr`. Alle diese Attribute sollen gegen Zugriff von außen geschützt werden, aber in möglichen späteren abgeleiteten Klassen zugreifbar sein (auch wenn es in diesem Praktikum noch keine abgeleiteten Klassen geben wird). Programmieren Sie die Getter und Setter für jedes dieser Attribute.

Der `MyCanvas` Konstruktor soll zwei `unsigned int` Parameter nehmen und die beiden `size…` Attribute entsprechend setzen. Der Konstruktor soll in

seiner Initialisierungsliste (oder zur Not in seinem Rumpf) das eindimensionale Array auf dem Heap allozieren mit Arraygröße passend zu den beiden Parameterwerten. Ferner soll im Rumpf des Konstruktors die Methode …
`void MyCanvas::init()`
… aufgerufen werden, welche alle Einträge des Arrays mit dem Buchstaben „Punkt" `'.'` initialisiert.

Da die `MyCanvas` Objekte mit dem Array auf dem Heap eine interne Ressource verwalten, müssen gemäß der „Rule of 3" auch der Destruktor, der Copy Konstruktor und der Assignment Operator für die Klasse `MyCanvas` von ihnen programmiert werden.

Die von außen aufrufbare Methode …
`void MyCanvas::set(unsigned int x, unsigned int y, char c)`
… schreibe den Buchstaben `c` an die Stelle des internen Arrays, welche der logischen Position `x,y` entspricht.

Die von außen aufrufbare konstante Methode …
`char MyCanvas::get(unsigned int x, unsigned int y) const`
… liefere den Buchstaben `zurück`, welcher sich an der logischen Position `x,y` befindet.

Die von außen aufrufbare konstante Methode …
`std::string MyCanvas::to_string() const`
… liefere den String zurück, welcher der textuellen Darstellung des Canvas entspricht, sprich die Buchstaben aller Zeilen aneinandergereiht mit jeweils einem Zeilenumbruch-Zeichen `'\n'` am Ende der Zeichen jeder Zeile (also im Resultatstring enthalten).

Die von außen aufrufbare konstante Methode …
`void MyCanvas::print() const`
… soll den `to_string()` Wert auf den Bildschirm ausgeben, mit einem weiteren Zeilenumbruch danach.

Legen Sie Dateien `test_MyCanvas.cpp`, `main.cpp` und `gip_mini_catch.h` an und kopieren Sie den Inhalt der in Ilias gegebenen gleichnamigen Dateien dort hinein. Das Hauptprogramm in `main.cpp` startet nur die Unit Tests.

Testlauf *(keine Benutzereingaben)*:

```
Alle Tests erfolgreich (70 REQUIREs in 5 Test Cases)
Drücken Sie eine beliebige Taste . . .
```

(Pflicht-) Aufgabe INF-10.02:
Klasse `MyRectangle`

Legen Sie in Visual Studio Code ein neues leeres Projekt an und innerhalb dieses Projekts dann leere Dateien `MyRectangle.h`, `MyRectangle.cpp`, `main.cpp`. Kopieren Sie die Dateien `MyCanvas.h` und `MyCanvas.cpp` aus der vorigen Aufgabe in dieses neue Projekt (werden hier erweitert, also kopieren!).

Programmieren Sie in den entsprechenden Dateien die Klasse `MyRectangle` gemäß den folgenden Anforderungen:

Jedes Objekt der Klasse `MyRectangle` besitze zwei `unsigned int` Attribute `x1` und `y1` für die linke obere Ecke des Rechtecks und zwei weitere `unsigned int` Attribute `x2`, `y2` für die rechte untere Ecke. Ferner speichere jedes `MyRectangle` Objekt einen Pointer `MyCanvas* canvas_ptr` auf ein `MyCanvas` Objekt.
Alle diese Attribute sollen gegen Zugriff von außen geschützt werden, aber Zugriff aus abgeleiteten Klassen soll erlaubt sein.
Programmieren Sie Getter und Setter für jedes dieser Attribute.

Der `MyRectangle` Konstruktor soll ein `MyCanvas` Objekt per Referenz sowie vier `unsigned int` Parameter übernehmen und die Attribute entsprechend setzen. Das `MyCanvas` Objekt muss per Referenz übernommen werden, um den „Original-Canvas", also die „Original-Zeichenfläche" als Parameter zu übernehmen und nicht eine Kopie.

Die von außen aufrufbare Methode …
`void MyRectangle::draw()`
… rufe über den `canvas_ptr` die (jetzt neue) Methode `MyCanvas` Methode …
`void MyCanvas::draw_rectangle()`
… auf, welche das Rechteck (gefüllt) mittels lauter `'#'` Zeichen in den Canvas einzeichne.

Legen Sie Dateien `test_MyRectangle.cpp`, `main.cpp` und `gip_mini_catch.h` an und kopieren Sie den Inhalt der in Ilias gegebenen gleichnamigen Dateien dort hinein.

Das Hauptprogramm startet die Unit Tests, generiert dann zufällige Koordinaten und zeichnet das entsprechende Rechteck.

Testlauf *(keine Benutzereingaben, Koordinaten sind zufällig)*:

```
Alle Tests erfolgreich (71 REQUIREs in 6 Test Cases)
(10, 2) bis (15, 11)
....................
....................
..........######....
..........######....
..........######....
..........######....
..........######....
..........######....
..........######....
..........######....
..........######....
..........######....
....................
....................
....................

Drücken Sie eine beliebige Taste . . .
```

(Pflicht-) Aufgabe INF-10.03: Templates

Programmieren Sie einen `struct` Datentyp Tupel, der zwei Komponentenwerte `komponente1` und `komponente2` von beliebigen, ggfs. unterschiedlichen Datentypen speichert. Definieren Sie Tupel als Template-Datentyp.

Programmieren Sie außerdem eine Template-Funktion …

```
int vergleiche( … p1 … , … p2 … )
```

… die zwei Werte des Tupel-Typs miteinander vergleicht und …
–1 zurückgibt, wenn beide Komponentenwerte des ersten Tupels kleiner sind als die jeweiligen Parameterwerte des zweiten Tupels, …
+1 zurückgibt falls beide Komponentenwerte des ersten Tupels größer sind als die jeweiligen Parameterwerte des zweiten Tupels …
und ansonsten den Wert **0** zurückgibt.

Die beiden Parameterwerte der Funktion seien von einem beliebigen, aber identischen Tupel-Typ (also Typen der Komponenten der beiden zu vergleichenden

`struct` Werte jeweils identisch, so dass die `struct` Werte auch vergleichbar sind, siehe Hauptprogramm zum Testlauf).

Für die Definition des Tupel-Typs und der Template-Funktion sollen zwei Dateien `tupel.h` und `tupel.cpp` genutzt werden.

In eine Datei `tupel_main.cpp` soll das im Folgenden angegebene Hauptprogramm eingefügt werden. Die Templates sollen die Methodik der expliziten Instanziierung nutzen (damit sollte klar sein, was in die Datei `tupel.h` gehört und was in die Datei `tupel.cpp`).

```
// Datei: tupel_main.cpp

#include <string>
#include <iostream>

#include "tupel.h"

int main()
{
    Tupel<std::string, int> hansi = { "Hansi", 8 };
    Tupel<std::string, int> willi = { "Willi", 77 };

    std::cout << vergleiche<std::string, int>(hansi, willi) << std::endl;

    Tupel<int, int> t1 = { 3 , 4 };
    Tupel<int, int> t2 = { 1 , 2 };

    std::cout << vergleiche<int, int>(t1, t2) << std::endl;

    Tupel<int, int> t3 = { 9 , 1 };
    Tupel<int, int> t4 = { 3 , 5 };

    std::cout << vergleiche<int, int>(t3, t4) << std::endl;

    system("PAUSE");
    return 0;
```

Testlauf *(keine Benutzereingaben)*:

```
-1
1
0
Drücken Sie eine beliebige Taste . . .
```