Universität Regensburg

PD Dr. J. Reischer 11.02.2014

Klausur "C#" WS 2013/2014

Nachname, Vorname	
Abschluss (BA, MA, FKN etc.)	
Matrikelnummer, Semester	
Versuch (1/2/3)	

Bitte füllen Sie zuerst den Kopf des Angabenblattes aus!

Die Klausur dauert 90 Minuten.

Es sind maximal 30 Punkte zu erreichen.

Es sind keine Hilfsmittel zugelassen.

Bitte beantworten Sie alle Fragen direkt auf das Angabenblatt.
Nutzen Sie ggf. die Rückseite und kennzeichnen Sie dies entsprechend.
Eigene Schmierblätter sind nicht erlaubt.

Bei mehreren oder mehrdeutigen Lösungen wird die schlechtere Lösung gewertet. Streichen Sie daher ungültige Lösungen eindeutig durch.

Verwenden Sie nur C# für Programmieraufgaben.

Viel Erfolg!

Teil I: Konzeptionelles

Aufgabe 1a: (2 Punkte)

Beschreiben Sie in je einem Satz den Unterschied zwischen *Definition* und *Deklaration*.

Aufgabe 1b: (2 Punkte)

Beschreiben Sie in je einem Satz, wozu *Konstruktoren* und *Destruktoren* eingesetzt werden.

Aufgabe 1c: (4 Punkte)

Kreuzen Sie korrekte Aussagen über Arrays an. Falsch angekreuzte Antworten geben Punktabzug innerhalb dieser Aufgabe. (Achtung: Es gibt nicht unbedingt genau 1 Punkt pro korrekt gelöster Aussage!)

- ☐ Arrays können im Prinzip beliebig viele Dimensionen haben.
- ☐ Ein 2×2-`int`-Array kann per `new int[,] ((0,0),(0,0))` initialisiert werden.
- ☐ Ein 1D-Array kann mit einer `foreach`-Schleife durchlaufen werden.
- ☐ Aus `bool`-Werten kann kein Array gebildet werden.
- ☐ Bei der Zuweisung eines Arrays an ein anderes wird der Inhalt komplett kopiert.
- ☐ Ein String `S` ist ein Array, dessen Elemente über `S[…]` abgegriffen werden können.
- ☐ Ein rechteckiges 2D-Array `A` kann per `A[Zeile,Spalte]` indexiert werden.
- ☐ Zwei Arrays `A` und `B` können mit dem `*`-Operator zu `A*B` verknüpft werden.

Teil II: Fehleranalyse

Aufgabe 2: (4 Punkte)

Finden Sie vier Fehler in folgender Klassendefinition. Die Fehler können an jeder Stelle auftreten. Achtung: Die Programmlogik selbst beinhaltet keine Fehler!

```
// Zähler, dessen Werte zwischen Min und Max liegen müssen
class Counter
{
  int Count, Min, Max;

  public void Counter(MyMin, MyMax)        // Konstruktor
  {
    if (MyMin <= MyMax)
    {
      Min = MyMin; Max = MyMax;
    }
    else                           // Min und Max werden umgekehrt,
    {                              // wenn Min > Max
      Min = MyMax; Max = MyMin;
    }
    Count = Min;                   // Startwert für Count innerhalb
  }                                // Min <= Count <= Max

  !Counter()                       // Destruktor (hier arbeitslos)
  { }

  public void CountUp()            // um 1 raufzählen, wenn noch
  {                                // nicht Max (sonst nichts tun)
    if (Count < Max) Count++;
  }

  public void CountDown()          // um 1 runterzählen, wenn noch
  {                                // nicht Min (sonst nichts tun)
    if (Count > Min) Count--;
  }

  public string GetData()          // alle Daten zurückgeben
  {
    return ("" + Min + " <= " + Counter + " <= " + Max);
  }
}
```

Teil III: Entwicklung

Aufgabe 3a: (4 Punkte)

Umschreiben Sie den Selektionsoperator `(Bedingung) ? (Wert_wenn_true) : (Wert_wenn_false)` durch eine eigene Funktion `int Select(bool Bedingung, int Wert_wenn_true, int Wert_wenn_false)`. Beispiel:

```
int X = (true && false) ? 1 : 2;        soll ersetzbar sein durch
int X = Select(true && false, 1, 2);    (Ergebnis jeweils X = 2)
```

In der Funktion `Select` darf der Selektionsoperator selbst natürlich nicht vorkommen!

Aufgabe 3b: (2 Punkte)

Obige `Select`-Funktion gilt nur für die Auswahl zwischen zwei `int`-Werten. Beschreiben Sie kurz, was alles zu ändern wäre, wenn die Funktion aus zwei `string`- oder `double`-Werten auswählen soll.

Aufgabe 4a: (4 Punkte)

Schreiben Sie eine Funktion `int GetDoubleChars(string S)`, die im String S die Indexposition des ersten von zwei beliebig gleichen Characters zurückgibt, die direkt hintereinander auftreten. Tritt kein Zeichen zwei Mal hintereinander auf, soll −1 zurückgegeben werden. Bei Buchstaben-Zeichen ist Groß- vs. Kleinschreibung relevant, da es sich um verschiedene Zeichen handelt. Beispiele:

Index = −1: "abc", "Ananas", "Aachen", "123", "!?#$"

Index = 0: "aabc" Index = 1: "Anna" Index = 2: "12334"
Index = 3: "!?#$$" Index = 4: "Schneeengel"

Aufgabe 4b: (2 Punkte)

Schreiben Sie eine Funktion `bool TestDoubleChars(string S)`, die im String S prüft, ob zwei oder mehr Zeichen direkt hintereinander auftreten und entsprechend `true` oder `false` zurückgibt (statt der Indexposition). Beispiele:

Ergebnis `false`: "abc", "Ananas", "Aachen", "123", "!?#$"
Ergebnis `true`: "aabc", "Anna", "Schneeengel", "12334", "!?#$$"

Aufgabe 5: (6 Punkte)

Schreiben Sie eine Funktion `string CreatePalindrom(int N)`, die je nach N einen spiegelbildlich symmetrischen String aus den Anfangsbuchstaben des Alphabets erzeugt:

N = 1: "A"
N = 2: "ABA"
N = 3: "ABCBA"
…
N = 26: "ABCDEFGHIJKLMNOPQRSTUVWXYZYXWVUTSRQPONMLKJIHGFEDCBA"

Es ist nicht erlaubt, für jedes N einfach den entsprechenden Wert zurückzugeben; der String muss tatsächlich *erzeugt* werden. Achten Sie auf die Grenzfälle (auch auf nicht angegebene!): Der String soll je nach N nur aus den Großbuchstaben des engl. Alphabets gebildet werden (keine dt. Umlaute etc.). Überlegen Sie selbst, was als Ergebnis für Grenzfälle sinnvoll ist. Achtung: Der mittlere Buchstabe darf nur *ein* Mal auftreten!

Hinweise: Der Buchstabe 'A' besitzt den nummerischen Kode 65, 'Z' liegt bei 90 (Unicode-Werte). Die Umwandlung eines Zahlenwertes wie 65 in einen entsprechenden Character wie 'A' kann durch eine explizite Typkonversion erreicht werden.