Anmerkungen

- Sichern Sie ihre Berechnungen, so weit wie möglich gegen fehlerhafte Eingaben ab.
- Verwenden Sie sprechende Variablennamen und formatieren Sie Ihren Programmtext ordentlich.
- Erstellen Sie Inputprompts und Ausgaben, sodass das Programm für BenutzerInnen angenehm zu bedienen ist.
- Schreiben Sie Ihre Programm so, dass mehrere Berechnungen ausgeführt werden können, ohne das Programm immer wieder neu starten zu müssen.
- Testen Sie Ihre Programme und unterziehen Sie die Ergebnisse einer Prüfung (Schätzung des erwarteten Ergebnisses, bzw. im Notfall mit dem Taschenrechner prüfen).
- Mit * markierte Beispiele sind etwas anspruchsvoller
- Für Beispiele mit Strings (Zeichenketten) verwenden Sie bitte die Klasse string. Hilfreiche Methoden für einen String x: x.at(pos) liefert das (veränderbare) Zeichen an der Stelle pos. Das erste Zeichen hat die Position 0. x.size() bzw. x.length() liefern die Länge (Anzahl der Zeichen) des Strings. Die beiden Methoden sind Synonyme. Mit dem Operator + können Zeichenketten verkettet werden. Weitere Methoden der Klasse string werden für die Aufgaben nicht benötigt und sollten auch nicht verwendet werden.

Aufgabe 1

Lösen Sie die Zusatzaufgaben zum Musterbeispiel aus der Übungseinheit.

Aufgabe 2 *

Lösen Sie die Zusatzaufgaben zu den Türmen von Hanoi

Aufgabe 3

Schreiben Sie ein Programm, das zunächst eine Zeichenkette einliest und dann immer wieder ein einzelnes Zeichen einliest und die Zeichenkette ausgibt, die entsteht wenn man das zuletzt eingegebene Zeichen aus der ursprünglichen Zeichenkette entfernt. (Das Programm soll durch Eingabe eines ! beendet werden.)

```
z.B. Eingabe: KrankeSchwestern
Eingabe: K
Ausgabe: rankeSchwestern
Eingabe: e
Ausgabe: KrankSchwstrn
Eingabe: !
Programmende
```

Aufgabe 4 *

Welche der folgenden Funktionen sind nicht tail-recursive? Können diese Funktionen in äquivalente tail-recursive Funktionen umgeschrieben werden?

```
int fact(int n) {
  if (n<=0) return 1;
  return n*fact(n-1);
}

int sum(int n, int res) {
  if (n<=0) return res;
  return sum(n-1, res+n);
}

int fibo(int n) {
  if (n<=0) return 0;
  if (n<=2) return 1;
  return fibo(n-1)+fibo(n-2);
}
```

Aufgabe 5

Lesen Sie eine Zeichenkette ein und geben Sie, je nach Benutzerwunsch eine kodierte, bzw. dekodierte Version der Eingabe aus. Verwenden Sie dazu eine einfache Cäsar-Kodierung.

```
z.B. Eingabe (kodieren): Test
Ausgabe: Uftu
```

Aufgabe 6

Lesen Sie eine Zeichenkette ein und geben Sie aus, welche Zeichen wie oft in der Eingabe vorkommen. Berücksichtigen Sie nur Zeichen mit ASCII-Codierungen zwischen ! (33) und ~(126). Zeichen, die gar nicht vorkommen, sollen ignoriert werden.

```
z.B. Eingabe: achtung,test
Ausgabe: ,(1) a(1) c(1) e(1) g(1) h(1) n(1) s(1) t(3) u(1)
```

Aufgabe 7

Schreiben Sie ein Programm, das eine Zeichenkette (nur Kleinbuchstaben) einliest und eine zweite Zeichenkette erstellt, in der das erste Zeichen der ursprünglichen Zeichenkette alphabetisch korrekt einsortiert ist. Das heißt im Ergebnis sollen zunächst alle Zeichen, die kleiner als das ursprünglich erste Zeichen sind, dann dieses erste Zeichen und schließlich die Zeichen, die größer als (oder gleich wie) das ursprünglich erste Zeichen sind, auftreten.

```
z.B. Eingabe: keinblasserschimmer
mögliche Ausgabe: eibaechieknlssrsmmr
```

Aufgabe 8 *

Wie Aufgabe 7, allerdings soll keine zweite Zeichenkette verwendet werden, sondern die Buchstaben in der eingegebenen Zeichenkette einfach entsprechend vertauscht werden.

Aufgabe 9

Schreiben Sie zwei Funktionen, die jeweils eine Zeichenkette als Parameter erhalten und die umgekehrte Zeichenkette (also die ursprüngliche Zeichenkette von rechts nach links gelesen) retournieren. Verwenden Sie dazu in der ersten Funktion eine Schleife und in der zweiten Funktion keine Schleife. Es dürfen nur die in den Anmerkungen am Beginn dieses Dokuments angeführten Methoden der Klasse string verwendet werden.

Aufgabe 10

Die Folge Fn ist definiert durch die Rekursion

$F_n = 2 * F_{n-1} + F_{n-2} * F_{n-3}$

mit den Startwerten $F_0=1$, $F_1=1$ und $F_2=1$

Schreiben Sie eine Funktion, die den Wert F_n an einer beliebigen Position n berechnet.

Aufgabe 11 *

Wie Aufgabe 10, verwenden Sie aber tail-recursion.

Aufgabe 12

Schreiben Sie eine Funktion `int ggt(int a, int b)`, die den größten gemeinsamen Teiler der beiden natürlichen Zahlen a und b berechnet. Verwenden Sie dazu keine Schleifen!

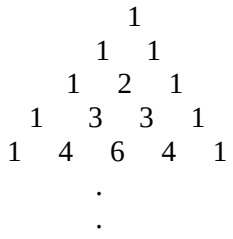
Aufgabe 13

Schreiben Sie eine Funktion `int binomial(int n, int k)`, die den Binomialkoeffizienten nach der Formel

$$\binom{n}{k} = \frac{n!}{k!\,(n-k)!}$$

berechnet. Verwenden Sie zur Berechnung der Fakultät, die aus der Vorlesung bekannte Funktion.

Schreiben Sie eine Variante der Funktion binomial, die eine Rekursion für die Binomialkoeffizienten verwendet, die sich aus dem Pascalschen Dreieck und dem binomischen Lehrsatz leicht herleiten lässt. Welchen Vorteil hat diese rekursive Variante gegenüber der Berechnung nach obiger Formel?



Pascalsches Dreieck .

Jeder Eintrag ergibt sich als Summe der beiden benachbarten Einträge in der Zeile darüber. Links und rechts wird jeweils mit 1 aufgefüllt. Die Einträge in der n-ten Zeile entsprechen den Faktoren, die in der Auflösung des Ausdrucks $(a+b)^n$ auftreten. So ist zum Beispiel $(a+b)^4 = a^4 + 4a^3b + 6a^2b^2 + 4ab^3 + b^4$. Da andererseits $(a+b)^n = \sum_{k=0}^{n} \binom{n}{k} a^{n-k}b^k$, erhält man leicht eine Rekursion für die Binomialkoeffizienten. (Hier werden die Eigenschaften des Pascalschen Dreiecks einfach vorausgesetzt. Genau genommen hat das Pascalsche Dreieck diese Eigenschaften wegen der Gültigkeit der Rekursion.)

Schreiben Sie eine weitere Variante der Funktion binomial, die folgende Formel verwendet (wenn Sie immer abwechselnd eine Division und eine Multiplikation ausführen, kann die gesamte Berechnung in ganzen Zahlen durchgeführt werden):

$$\binom{n}{k} = \prod_{i=1}^{k} \frac{n-k+i}{i}$$

Welche der drei Versionen der Funktion binomial würden Sie in der Praxis einsetzen?

Aufgabe 14

Schreiben Sie die Funktionen `int mult(int m, int n)` und `int power(int m, unsigned int n)`, die das Produkt m*n bzw. die Exponentialfunktion m^n der ganzen Zahlen m und n berechnen. Verwenden Sie dazu weder Schleifen, noch den Operator * (Multiplikation). Auch das Verwenden irgendwelcher externer Routinen (z.B. Mathematikbibliotheken) ist nicht erlaubt! Beachten Sie, dass int-Werte auch negativ oder gleich 0 sein können.
Hinweis: m*n = m+m*(n-1) für n>0 bzw. $m^n=m*m^{n-1}$

Aufgabe 15

Die Funktionen f(n), g(n) und h(n) sind für natürliche Zahlen n wie folgt definiert:

```
f(0) = 0
g(0) = 0
h(0) = 1
f(n) = h(n)+g(n-1)
g(n) = 2*f(n)
h(n) = n*h(n-1)
```

Schreiben Sie ein Programm, das die Berechnung der Funktion f für ein beliebiges n erlaubt.

Aufgabe 16

Schreiben Sie eine Funktion `int stack()`, durch deren Aufruf der Stack zum Überlaufen gebracht wird. (Anmerkung: Dieser Überlauf, der normalerweise als "stack overflow" bezeichnet wird, wird auf den Übungsrechnern durch den Abbruch des Programms und die Ausgabe der Fehlermeldung "Segmentation fault" angezeigt.) Wie oft muss die Funktion stack() rekursiv aufgerufen werden, bis es zum stack overflow kommt? Wie können Sie das beschleunigen und wie können Sie aus Ihren Ergebnissen die Größe des Stacks abschätzen?

Aufgabe 17

Schreiben Sie eine Funktion `bool intTest(int n, char c)`, die prüft, ob die Ziffer c in der Zahl n vorkommt. Verwenden Sie in der Funktionsdefinition keine Schleifen.

Aufgabe 18

Schreiben Sie eine Funktion, die eine natürliche Zahl als Parameter enthält und alle Ziffern der Zahl von rechts nach links gelesen und jeweils durch ein Leerzeichen voneinander getrennt ausgibt, dann einen Stern und alle Ziffern der eingegebenen Zahl von links nach rechts gelesen und jeweils durch ein Leerzeichen getrennt. Verwenden Sie keine Schleife in der Funktion.

```
z.B. Eingabe: 12345
Ausgabe: 5 4 3 2 1 * 1 2 3 4 5
```

Aufgabe 19

Schreiben Sie eine Funktion, die eine Zeichenkette als Parameter erhält und eine "vermixte" Zeichenkette retourniert. Dabei sollen im Ergebnis zunächst alle Zeichen mit ungeradem Index im Parameter in der ursprünglichen Reihenfolge enthalten sein und daran anschließend alle Zeichen mit geradem Index im Parameter in umgekehrter Reihenfolge. Verwenden Sie weder Schleifen noch Routinen aus externen Bibliotheken (z.B. strlen). Beispiel:

```
Parameter: "abcdefghi"
Returnwert: "bdfhigeca"
```

Aufgabe 20 *

Schreiben Sie ein Programm, das eine natürliche Zahl n einliest und die expandierte Form des Ausdrucks $(a+b)^n$ ausgibt. Also z.B.:

```
Eingabe: 3
Ausgabe: a^3+3a^2b+3ab^2+b^3
```

(Vergleiche Aufgabe 13).

Aufgabe 21 *

Schreiben Sie ein Programm, das eine Zahl einliest und alle möglichen Permutationen der in der Zahl vorkommenden Ziffern ausgibt. (Sie können davon ausgehen, dass jede Ziffer nur einmal eingegeben wird.)

```
z.B. Eingabe: 123
Ausgabe: 123 132 213 231 312 321
```

Aufgabe 22 *

Wie Aufgabe 21, allerdings dürfen Ziffern auch mehrfach auftreten.

Aufgabe 23 *

Wie Aufgabe 21, aber für Zeichenketten

```
z.B. Eingabe: abc
Ausgabe: abc acb bac bca cab cba
```

Aufgabe 24 *

Wie Aufgabe 23, allerdings dürfen Zeichen auch mehrfach auftreten