Aufgabenblatt 6: Funktionen

Kapitel 1: Praktikumsaufgaben

6.1.1: Potenzfunktion ✔

Erstellen Sie eine Funktion, die $y = x^p$für eine beliebige Gleitkommazahl x und eine nicht-negative ganzzahlige Potenz p berechnet.

Erstellen Sie ein Hauptprogamm, in dem Sie die Funktionswerte für die Potenzfunktion in einem bestimmten Bereich mit einer bestimmten Schrittweite ausgibt. Die Bereichsgrenzen und die Schrittweite sollen dabei vom Benutzer eingegeben werden.

Feedback

Du hast am 10.11.16, 13:01 folgendes Feedback zu dieser Aufgabe abgegeben:

bearbeitet: komplett
Schwierigkeit: 3/10
Spaß: 4/10
Zeit: 8min

6.1.2: Exponentialfunktion ✔

Die Exponentialfunktion e^xkann durch die Potenzreihe $e^x = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \frac{x^4}{4!} + \ldots$ berechnet werden.

Erstellen Sie eine C-Funktion, die die Exponentialfunktion über ihre Potenzreihe näherungsweise berechnet.

Erstellen Sie einen Testrahmen, in dem Ihre Implementierung mit der Implementierung aus der Runtime-Library (exp) verglichen wird.

Feedback

Du hast am 10.11.16, 14:37 folgendes Feedback zu dieser Aufgabe abgegeben:

bearbeitet: komplett
Schwierigkeit: 4/10
Spaß: 7/10
Zeit: 20min

6.1.3: Binomialkoeffizient ✔

Der Binomialkoeffizient $\binom{n}{k}$(sprich "n über k") ist durch die Formel

$$\binom{n}{k} = \frac{n!}{k!(n-k)!} = \frac{n \cdot (n-1) \cdot (n-2) \ldots (n-k+3) \cdot (n-k+2) \cdot (n-k+1)}{1 \cdot 2 \cdot 3 \ldots (k-2) \cdot (k-1) \cdot k}$$

definiert.

Erstellen Sie eine Funktion, die Binomialkoeffizienten nach dieser Formel berechnet! Implementieren sie die Funktion so, dass möglichst große Binomialkoeffizienten ganzzahlig berechnet werden können, ohne dass bei Zwischenergebnissen ein Überlauf stattfindet.

Testen Sie ihre Funktion, indem Sie das Pascalsche Dreieck

$$\begin{array}{ccccccccc}
 & & & & \binom{0}{0} & & & & \\
 & & & \binom{1}{0} & & \binom{1}{1} & & & \\
 & & \binom{2}{0} & & \binom{2}{1} & & \binom{2}{2} & & \\
 & \binom{3}{0} & & \binom{3}{1} & & \binom{3}{2} & & \binom{3}{3} & \\
\binom{4}{0} & & \binom{4}{1} & & \binom{4}{2} & & \binom{4}{3} & & \binom{4}{4}
\end{array}$$

auf dem Bildschirm ausgeben.

Feedback

Du hast am 17.11.16, 10:21 folgendes Feedback zu dieser Aufgabe abgegeben:

bearbeitet: komplett
Schwierigkeit: 3/10
Spaß: 5/10
Zeit: 15min

6.1.4: Stringfunktionen ✔

Erstellen Sie eigene Implementierungen für die folgenden Funktionen aus der Runtime-Library:

- `atoi`
- `strcmp`
- `strcat`
- `strstr`

Nennen Sie diese Funktionen `myatoi`, `mystrcmp`, `mystrcat` und `mystrstr` und geben Sie diesen Funktionen die gleiche Schnittstelle wie den entsprechenden Funktionen der Runtime-Library. Erstellen Sie einen Testrahmen, in dem Sie alle Funktionen testen. Implementieren Sie Ihre Funktionen und den Testrahmen in unterschiedlichen Quellcode-Dateien und verwenden Sie eine Headerdatei für die Funktionsprototypen.

Feedback

Du hast am 21.11.16, 15:53 folgendes Feedback zu dieser Aufgabe abgegeben:

bearbeitet: komplett
Schwierigkeit: 5/10
Spaß: 8/10
Zeit: 25min

Kapitel 2: Vertiefung und Selbsttest

6.2.1: Schiffe versenken

Sie kennen das Spiel "Schiffe versenken", bei dem ein Spieler die in einer in Planquadrate unterteilten Ebene verborgenen Schiffe seines Gegners aufspüren und versenken muss. Erstellen Sie ein Programm, mit dem Sie "Schiffe versenken" spielen können! Das Programm sollte zunächst die Höhe und die Breite des Spielfeldes und die Anzahl der Schiffe vom Benutzer erfragen:

```
Anzahl Zeilen (1-20): 8
Anzahl Spalten (1-20): 10
Anzahl Schiffe (1-9): 4
```

Dann sollte der Benutzer die Schiffspositionen durch Angabe der Startkoordinaten, der Länge und der Richtung des Schiffs eingeben können. Das Programm gibt zwischendurch immer die aktuelle Belegung des Spielfelds aus, damit der Benutzer sieht, wo er noch Schiffe positionieren kann:

```
1.Schiff: C7-3-R

   ABCDEFGHIJ
 1 ----------
 2 ----------
 3 ----------
 4 ----------
 5 ----------
 6 ----------
 7 --111-----
 8 ----------

2.Schiff: G2-5-U

   ABCDEFGHIJ
 1 ----------
 2 ------2---
 3 ------2---
 4 ------2---
 5 ------2---
 6 ------2---
 7 --111-----
 8 ----------
```

Die Eingabe erfolgt dabei immer in der Form:

Spaltenbuchstabe (a,b, ...) Zeilennummer (1,2,...)- *Länge (1-5) - Richtung (R oder r, U oder u)*

Das Programm überprüft, ob an der gewählten Stelle ein Schiff positioniert werden kann, weist den Benutzer auf eventuelle Fehler hin und fordert im Fehlerfall eine Neueingabe an.

Wenn alle Schiffe positioniert sind, wechselt das Programm in den Ratemodus und fordert den Benutzer fortlaufend auf, Positionen (Zeile, Spalte) zu raten. Nach jedem Rateversuch zeigt das Programm den aktuellen Spielstand an. Dazu verwendet es die folgenden Zeichen:

```
- in diesem Feld ist noch nichts passiert
0 hier wurde geraten aber nichts getroffen
X hier wurde geraten und getroffen (Schiff ist aber noch nicht versenkt)
# Schiff versenkt
```

Eine vollständige Spielstandsanzeige könnte wie folgt aussehen:

```
   ABCDEFGHIJ
 1 ------0---
 2 -----0x0--
 3 -0----x---
 4 -----0----
 5 -0--------
 6 ---0------
```

```
7 -0###---0-
8 -0----0---
```

Zerlegen Sie das Problem konsequent in Teilprobleme und verwenden Sie Funktionen zur Lösung der Teilaufgaben!

Falls Sie noch Fragen zur Aufgabenstellung haben, nutzen Sie das ausführbare Programm in der Anlage als Vorlage für Ihre Lösung.

6.2.2: Zahlenmemory

Sie kennen das Spiel "Memory". Erstellen Sie ein Programm, mit dem ein Spieler auf einem 4x4-Spielfeld "Zahlenmemory" spielen kann. Auf dem Spielfeld stehen verdeckt 8 Zahlenpaare (Zahlen 1 - 8). Der Spieler deckt jeweils zwei Felder auf. Sind die Zahlen auf den Feldern gleich, hat er ein Paaar gefunden und das Paar bleibt aufgedeckt. Ungleiche Zahlen werden wieder verdeckt und der Spieler hat den nächsten Versuch. Es wird so lange gespielt, bis alle Paare aufgedeckt wird.

Versuchen Sie das Spielfeld durch Zufallszahlen so zu initialisieren, dass dem Spieler die Zahlen unbekannt sind. Dazu verwenden Sie die Funktionen `srand` und `rand` aus der Runtime-Library.

In der Gestaltung der Bedienoberfläche sind Sie völlig frei, Sie können der Einfachheit halber davon ausgehen, dass der Benutzer immer korrekte und sinnvolle Eingaben macht.