DAP2 Praktikum für ETIT und IKT, SoSe 2018, Kurzaufgabe K3

Fällig am 14.05 um 18:00

Es gelten die bekannten Programmierregeln in unbarmherziger Härte.

Kurzaufgabe K3.1 (1 Punkt)

- Implementieren Sie den `QuickSort` Algorithmus als

```
template <class T> void QuickSort(vector<T> &A)
```

- Dieser sortiert das Feld `A[0] ... A[n-1]` aufsteigend. Er ruft den eigentlichen, rekursiv implementierten Algorithmus auf. Der Pseudocode des rekursiven Kerns von `QuickSort` ist wie folgt:

```
function QuickSort(A,l,r)
      if l<r then
            i = l
            j = r
            pivot = A[floor((l+r)/2)]
            while i <= j do
                  while A[i] < pivot do
                        i = i+1
                  while A[j] > pivot do
                        j = j-1
                  if i <= j then
                        Tausche A[i] mit A[j]
                        i = i+1
                        if j >0 then j = j-1
            QuickSort(A,l,j)
            QuickSort(A,i,r)
```

  Wenn `n` Elemente sich in `A` befinden, so beginnt die Rekursion mit `QuickSort(A,0,n-1);`

- Bauen Sie `QuickSort` in den herunter geladenen Quelltext von `Sorting.cpp` ein. Der Pseudocode ist nur ein Gerüst, aus dem Sie eine korrekte Funktion in C++ realisieren müssen.
- Wie schon bei den letzten Langaufgabe besprochen, verwenden Sie bitte den Typ `size_t` statt `int` um Indexoperationen durchzuführen. Beachten Sie, dass `size_t` nie negativ wird.
- Verwenden Sie `assert` um Pre- und Post-Conditions zu überprüfen!
- Überprüfen Sie die Richtigkeit Ihrer Implementierung.
- Fertigen Sie einen Plot für die Laufzeit bei zufällige Daten mit Matlab an.

Kurzaufgabe K3.2 (1 Punkt)

Wenn ein Fahrkartenautomat Wechselgeld ausgibt, so möchte man den Betrag sicherlich nicht in lauter 1-Cent-Münzen ausgezahlt bekommen. Im Normalfall möchte man den Betrag mit möglichst wenigen Münzen bzw. Scheinen erhalten. Der Automat hat also ein Optimierungsproblem zu lösen. Der Automat kennt die Wertigkeiten $w_k \ldots w_1$ der Währung, wobei $w_k > w_{k-1} > \ldots > w_1$ und $w_1 = 1$ ist. Der Automat kennt natürlich auch den Betrag B, der zurückgezahlt werden muss. Alle Zahlen sind ganzzahlig und müssen positiv sein. Der Automat soll nach dem *gierigen Münzwechsel Algorithmus* vorgehen:

Ist $B \geq w_k$ dann gibt er n Münzen der Wertigkeit w_k aus mit $0 \leq B - n \cdot w_k < w_k$ und berechnet $B' = B - n \cdot w_k$. Danach führt er das gleiche für B' und w_{k-1} aus, usw… Der Automat hat einen unendlich großen Münzvorrat.

- Schreiben Sie ein Programm `CoinChange` das diese umgangssprachliche Beschreibung umsetzt. Es erwartet zwei Eingabeparameter. Der erste Parameter soll ein ganzzahliger Wert > 0 sein, der dem Betrag *B* entspricht. Der zweite Parameter soll ein String sein, der für die Währung steht und die Stückelung vorgibt.
- Wir lassen zwei Währungen zu: „Euro“ und „Knopf“. Die Stückelung der beiden Währungen ist:
  - Euro : 200, 100, 50, 20, 10, 5, 2, 1
  - Knopf: 200, 100, 50, 20, 10, 5, 4, 2, 1
- Der Münzwechselalgorithmus soll eine eigene

```
void Change(int Betrag, vector<int> Steuckelung, vector<int> &Muenzen)
```

Funktion sein, der der `Betrag` und die `Stueckelung` der ausgewählten Währung übergeben wird. Ebenfalls erhält Sie die Referenz auf `Muenzen`, das die Anzahl der zu verwendenden Münzen zurückliefert. `Change` wird vom Hauptprogramm aufgerufen.

Beispiel: Für *B* = 455 und "Euro", soll `Muenzen = {2,0,1,0,0,1,0,0}` sein.

- Bezüglich unsinniger Eingaben, Überprüfungen zur Laufzeit, Speicherlöcher, Vermeidung alter C-Funktionen, etc. gilt das übliche. Halten Sie sich möglichst genau an das Aussehen der Beispielläufe unten!
- Beispielläufe für `CoinChange`:

```
> CoinChange
Usage:
CoinChange Value Euro|Knopf
  Value: integer value
  Euro/Knopf decides between available coin values.

> CoinChange -1 Knopf
First Parameter Value must be at least 1

>CoinChange 79 Knopf
Second argument must be either Euro or Knopf

>CoinChange 79 euro
79 euro is:
1 x 50  +  1 x 20  +  1 x 5  +  2 x 2 euro

>CoinChange 79 Knopf
79 knopf is:
1 x 50  +  1 x 20  +  1 x 5  +  1 x 4 knopf
```