Aufgabenstellung

Einleitung

Diese Aufgabe betrachtet Algorithmen auf ein- und zwei-dimensionalen *float*-Feldern. Ein-dimensionale Felder werden jeweils als Parameter *float *data* mit der Größe *int size* und zwei-dimensionale Felder als Parameter *float **data* mit den Größen *int size1* für die erste und *int size2* für die zweite Dimension übergeben.

Hinweise

- Prüfen Sie Parameter auf ihre Gültigkeit wie in der Aufgabenstellung angegeben. Sind keine expliziten Parametereinschränkungen angegeben, ist auch keine Prüfung notwendig.
- Eine Exception wird durch das Schlüsselwort `throw` geworfen und es muss dem Konstruktor der Exception eine Fehlernachricht übergeben werden: `std::range_error("Hallo Welt, ich bin ein Fehler.")`

Funktionen

Die genaue Beschreibung der zu implementierenden Funktionen entnehmen Sie der Datei `task.cpp`. Die einzelnen Funktionen werden aus der Datei `main.cpp` aufgerufen. Jede Funktion ist unabhängig von den anderen Funktionen und kann separat implementiert werden, wobei die vorgegebene Reihenfolge nicht eingehalten werden muss.

Algorithmus für `down_sampling`

Der zu implementierende Algorithmus für die Funktion `down_sampling` ist durch den folgenden Pseudocode gegeben. Im Pseudocode verwenden wir die Notation `variable[i]`, um wie in C++ auf das i-te Element des Feldes zuzugreifen.
Die Funktion erhält als Parameter ein *float*-Feld *data*, dessen Größe *size* und eine natürliche Zahl *M*. Die Rückgabe ist ein *float ** auf ein mit *new* neu allokiertes Feld.

```
Erzeuge auf dem Stack eine Variable K vom Typ int und initialisiere diese mit dem Wert 5.
Erzeuge auf dem Stack ein statisch allokiertes float-Feld namens
  filter der Größe K, welches mit den Werten { 0.05, 0.2, 0.5, 0.2, 0.05 }
  (an den Stellen 0 bis K - 1) initialisiert ist.
Erstelle eine Variable processed vom Typ float * und allokiere darin mit new
  Speicher für size/M Elemente.
Zähle eine Variable "i" von 0 bis einschließlich "size/M - 1" hoch {
  Erzeuge eine Variable sample vom Typ float, die mit 0 initialisiert ist.
  Zähle eine Variable "j" von 0 bis einschließlich "K - 1" hoch {
    Falls (i*M+j < size) {
      Addiere auf den Wert von sample den Wert von data[i*M+j] * filter[j].
    }
  }
  Weise processed[i] den Wert von sample zu.
}
Liefere processed als Rückgabe zurück.
```