Grundlagenpraktikum Rechnerarchitektur (GRA)

Tutorübung

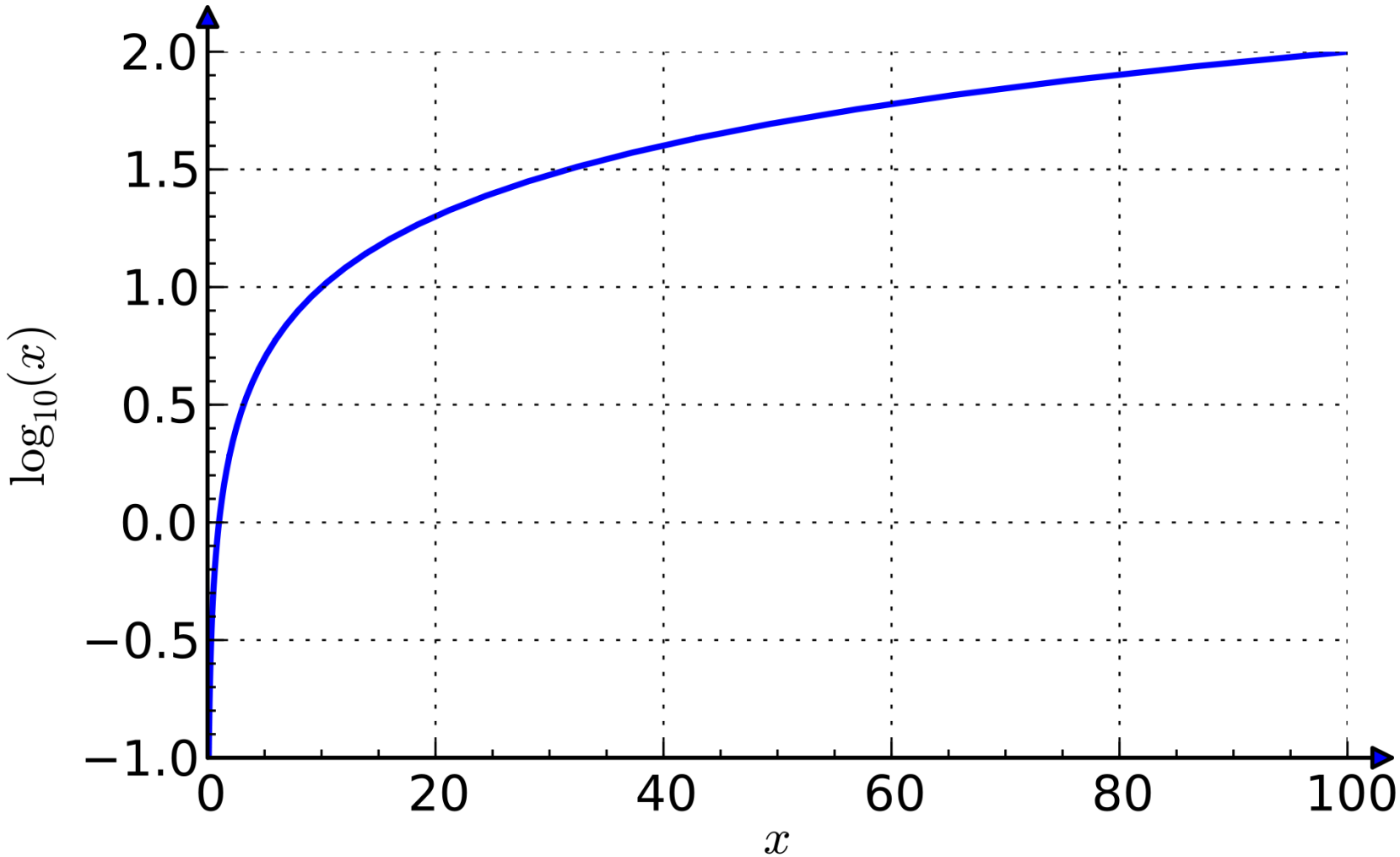
Moritz Beckel

30.05.2022 16:00 / 03.06.2022 15:00

Hausaufgaben

- Circ
- Numquad
- Round
- Quiz

Hausaufgaben



T6.1 Konvertierung von Strings zu Zahlen

Vorlage: `https://gra.caps.in.tum.de/m/stringtonum.tar`

1. Die Funktionen `atol`, bzw. `atof` um Strings in `long`, bzw. `double` Werte zu konvertieren kennen Sie bereits. Warum ist deren Verwendung oft problematisch? Sehen Sie sich hierzu die relevanten `man` pages an.

2. Basierend auf den eben betrachteten `man` pages: Welche weiteren Funktionen bieten sich für die Konvertierung von Strings in `long`, bzw. `double` Werte womöglich an?

3. Finden Sie die Bedeutung der Parameter `const char* nptr` und `char** endptr` *der* `strtol` *und* `strtod` *Funktionen heraus.*

T6.1 Konvertierung von Strings zu Zahlen

4. Konvertieren Sie nun mit `strtod` die Nutzereingabe in einen `double` Wert. Wie können Sie folgende Fehler abfangen?
   - Die Nutzereingabe repräsentiert keine gültige Fließkommazahl.
   - Die übergebene Fließkommazahl passt nicht in den Wertebereich eines `double`.

Hausaufgaben

P6.1 Nutzung von SIMD in Assembler: Saxpy [3 Pkt.]

Im Folgenden werden wir die SSE-Erweiterungen für die Optimierung einer häufig verwendeten Operation der linearen Algebra[4] nutzen. Die Funktion saxpy mit folgender Signatur führt folgende Operation durch:

```
void saxpy(size_t n, float alpha, const float x[n], float y[restrict n])
```

$$\vec{y} \leftarrow \alpha \cdot \vec{x} + \vec{y} \qquad \vec{x}, \vec{y} \in \mathbb{R}^n, \alpha \in \mathbb{R}$$

Hausaufgaben

P6.2 Nutzung von SIMD in C: Sdot [3 Pkt.]

Implementieren Sie die Funktion `sdot` unter Nutzung von SIMD-Intrinsics in C, welche das Skalarprodukt von zwei Vektoren berechnet. Die Funktion hat folgende Signatur und führt die untenstehende Berechnung durch:

```
float sdot(size_t n, const float x[n], const float y[n])
```

$$dot = \vec{x}^T \cdot \vec{y} \qquad \vec{x}, \vec{y} \in \mathbb{R}^n$$

Empfehlung: Akkumulieren Sie jeweils das Produkt von vier benachbarten Werten in einem Vektorregister, dessen Elemente Sie nach Ende Ihrer Block-Schleife aufsummieren.