Hochschule Niederrhein	PE2 / SE2	Dozenten
Fachbereich 03	Übung 1	Rethmann / Brandt
Bachelor Informatik	SoSe 2021	Stockmanns / Quix

Modulare Programmierung in C

Lernziele

Anwenden der modularen Programmierung in C sowie Vertiefen der Kenntnisse über die Gültigkeit und Sichtbarkeit von Variablen. Modulare Programmierung beschreibt die Aufteilung eines Programms in Module, die einzeln geplant, programmiert und getestet werden können.

Aufgabe 1:

Implementieren Sie eine Datenstruktur `stack_t` mittels modularer Programmierung. Der Stack soll bei Bedarf automatisch vergrößert werden, sodass er beliebig viele Werte vom Typ `float` speichern kann. Die Datenstruktur soll folgende Operationen unterstützen:

- `stack_t *createStack();`
  erzeugt einen leeren Stack
- `char isEmpty(stack_t *s);`
  prüft, ob der Stack `s` leer ist
- `void push(stack_t *s, float value);`
  legt den Wert `value` auf den Stack `s`
- `float top(stack_t *s);`
  liefert das zuletzt eingefügte Element des Stacks `s`
- `void pop(stack_t *s);`
  entfernt das zuletzt eingefügte Element vom Stack `s`
- `char getError(stack_t *s);`
  liefert den Inhalt der Fehlervariablen
- `void destroyStack(stack_t *s);`
  zerstört den Stack und gibt belegten Speicherplatz frei

Wenn auf einem leeren Stack ein `top` oder `pop` ausgeführt wird, soll eine Fehlervariable gesetzt werden.

Schreiben Sie ein Programm in C, dass einen in umgekehrter polnischer Notation gegebenen arithmetischen Ausdruck einliest, mit Hilfe des Stacks auswertet und das Ergebnis anzeigt. Unter `https://de.wikipedia.org/wiki/Umgekehrte_polnische_Notation` finden Sie Informationen zu der umgekehrten polnischen Notation und wie man mit deren Hilfe den Wert arithmetischer Ausdrücke berechnen kann.

Zusatz-Aufgabe Z1:

Implementieren Sie einen Datentyp `complex_t` mittels modularer Programmierung. Komplexe Zahlen[1] bestehen aus Real- und Imaginärteil und werden z.B. in der Elektrotechnik verwendet. Bei der Darstellung einer sinusförmigen Wechselspannung als komplexe Größe und entsprechenden Darstellungen für Widerstände, Kondensatoren und Spulen vereinfachen sich die Berechnungen des elektrischen Stromes, der Wirk- und der Blindleistung in einer Schaltung.

Der Datentyp soll folgende Operationen unterstützen:

- `complex_t *getComplex(double re, double im);`
  erzeugt eine komplexe Zahl mit entsprechendem Real- und Imaginärteil
- `complex_t *addComplex(complex_t *a, complex_t *b);`
  addiert die beiden komplexen Zahlen `a` und `b`
- `complex_t *subComplex(complex_t *a, complex_t *b);`
  subtrahiert die komplexe Zahl `b` von `a`
- `complex_t *mulComplex(complex_t *a, complex_t *b);`
  multipliziert die beiden komplexen Zahlen `a` und `b`
- `complex_t *divComplex(complex_t *a, complex_t *b);`
  dividiert die komplexe Zahl `a` durch `b`
- `complex_t *cleanComplex(complex_t *c);`
  gibt den durch die komplexe Zahl `c` belegten Speicher frei

Schreiben Sie ein Programm, mit dem Sie die obigen Operationen testen können.

[1] `https://de.wikipedia.org/wiki/Komplexe_Zahl`