Selbststudium 02 STACK

PROG C

Stefan R. Bachmann - bachmste

Zürich, 08.03.2015

Grundidee

Mein Grundgedanke war eine möglichst benutzerfreundliche Eingabe für den Stack zu gestalten. Daher habe ich ein "Pseudointerface" integriert. Dieses soll eine Auswahl an Möglichkeiten bieten um den Stack zu bearbeiten. Eine letzte Auswahl soll die Möglichkeit bieten in einen anderen Programmteil überzugehen. Der Einfachheit halber ist diese letzte Option als eine weitere Exit Option implementiert. Aus diesem Grund gibt es zwei Möglichkeiten das Programm zu verlassen. (Siehe Appendix A: FlowChart der main())

Pseudointerface

```
printf("STACK OPERATIONS AVAIBLE\n");
   while(option){
        (void) printf("-----------------------------------\n");
        (void) printf("          1 ----->   PUSH          \n");
        (void) printf("          2 ----->   POP           \n");
        (void) printf("          3 ----->   PRINT STACK   \n");
        (void) printf("          4 ----->   QUIT!         \n");
        (void) printf("-----------------------------------\n");

        (void) printf("Please enter your choice to proceed!\n");
```

case switch

Das Pseudointerface ist lediglich mit ein paar Zeilen printf implementiert. Nichts überaus spezielles. Für die Implementierung der Auswahl wird die Eingabe in einer simplen Variable gespeichert.
Um Zwischen den verschiedenen Funktionen zu switchen, habe ich verschiedene cases implementiert. Als default ist die print-Funktion definiert. Wie erwähnt soll case 4 als mögliche Erweiterung dienen. Ausserdem habe ich dieses Mal den Buffer geflusht, wie auf der letzten Zeile zu erkennen ist.

case switch implementation

```
        (void) printf("Please enter your choice to proceed!\n");
        (void) scanf("%d", &choice);
        switch (choice){
            case 1:
                push();
                break;
            case 2:
                pop();
                break;
            case 3:
                printStack();
                break;
            case 4:
                (void) printf("Goodbye!");
                return;

            default:
                printStack();
        }
        fflush(stdin);
        (void) printf("Would you like to continue? [0/1]?\n");
        (void) scanf(„%d", &option);
    }
```

push()/pop()

Die Implementierung dieser beiden Funktionen ist relativ ähnlich. Zu beachten ist vor allem, dass bei push() die position des "Zeigers" um 1 höher wird und bei der pop() Funktion um 1 wieder absinkt. Wichtig ist auch zu prüfen, ob der Stack voll ist oder nicht. Dazu wurde eine globale Variable STACK_LENGTH definiert und in die Headerdatei ausgelagert.

Weiter habe ich noch die Ausgabe bei pop(), wie auch bei printStack(), so angepasst, dass jeweils nur vier Nachkommastellen angezeigt werden.

push()/pop() implementation

```
// Function to push an element onto the stack
void push(void){
    double number;
    if(position == (STACK_LENGTH - 1)){
        (void) printf("Stack is full! \n");
        return;
    } else {
        (void) printf("Enter the element you want to push onto the stack, please. \n");
        (void) scanf("%lf", &number);
        position += 1;
        stack[position] = number;
    }
    return;
}

// Function to pop the top element of the stack
double pop(void){
    double number;
    if(position == -1){
        (void) printf("Stack is Empty\n");
        return (position);
    } else {
        number = stack[position];
        (void) printf("Poped element is = %.4lf \n", stack[position]);
        position -= 1;
    }
    return (number);
}
```

Headerdatei

Letztlich habe ich noch die ganzen globalen Variablen, wie auch die Initalisierung der diversen Funktionen in eine Header-datei ausgelagert. Die main() Funktion funktionierte immer noch ohne Probleme im Anschluss. Weiter ist ist der Unterschied zwischen static const und #define zu erläutern. static const lässt die Betrachtung der scope zu und ist typenspezifisch. lässt also zu, den Typ der Variable zu definieren. Allerdings habe ich gelesen, dass die static const in C keine wirklichen Konstanten sind. Das static sorgt spezifisch dafür, dass die scope auf das spezifische file bezogen ist. Die Variable kann damit nicht mehr mit "extern" implementiert werden.

Headerdatei

```
#define STACK_LENGTH 10

static double stack[STACK_LENGTH];
static int position = 0;

void push(void);
double pop(void);
void printStack(void);
```

Appendix
A

FlowChart der main() Funktion