OPR Praktikum
Übung: Grundlagen

Thomas Mahr

29. Februar 2016

1 Motivation

Das OPR-Fach setzt Kenntnisse in C voraus. Mit dieser Übung sollen einige Grundlagen wiederholt werden, die Ihnen die Arbeit mit C++ erleichtern:

- Verständnis des Übersetzungsvorgangs in C und C++
- Benutzung der Kommandozeile
- Zeiger

2 Voraussetzungen

Informatik 1 und Informatik 2

3 Vorbereitung

1. Stellen Sie sicher, dass Sie den C Übersetzungsvorgang[1] verstanden haben!
   - Was ist der Präprozessor?
   - Was ist der Compiler?
   - Was ist der Assembler?
   - Was ist der Linker?
2. Stellen Sie sicher, dass Sie auf den Rechnern im Praktikumsraum oder auf Ihrem mitgebrachten Rechner sowohl auf der Kommandozeile als auch in einer integrierten Entwicklungsumgebung übersetzen können.
3. Stellen Sie sicher, dass Sie auf den Rechnern im Praktikumsraum oder auf Ihrem mitgebrachten Rechner mit einem Debugger arbeiten können.

[1] http://wap-pool.math.uni-bayreuth.de/prog/compilierung.html

4 Aufgabe: Fehler

4.1 Präprozessor

- Erstellen Sie ein Programm, bei dem der Präprozessor auf einen Fehler stößt.
- Erklären Sie den Fehler.
- Beheben Sie den Fehler.

4.2 Compiler

- Erstellen Sie ein Programm, bei dem der Compiler auf einen Fehler stößt.
- Erklären Sie den Fehler.
- Beheben Sie den Fehler.

4.3 Assembler

- Stellen Sie durch Setzen eines geeigneten Compiler-Schalters den vom Compiler erzeugten Assembler-Code dar.

4.4 Linker

- Erstellen Sie ein Programm, bei dem der Linker auf einen Fehler stößt.
- Erklären Sie den Fehler.
- Beheben Sie den Fehler.

4.5 Laufzeit

- Erstellen Sie ein Programm, das zu einem Laufzeitfehler führt.
- Erklären Sie den Fehler.
- Beheben Sie den Fehler.

5 Aufgabe: Zeiger und Debugger

Gegeben ist dieses Programm:

```
1  #include<stdio.h>
2  #include<stdlib.h>
3
4  int main()
5  {
6    const int nx=3;
7    const int ny=2;
8    int** matrix;
9
10   matrix = (int**)malloc(ny*sizeof(int*));
11   for(int y=0; y<ny; y++)
12   {
13     matrix[y] = (int*)malloc(nx*sizeof(int));
14     for(int x=0; x<nx; x++)
15     {
16       matrix[y][x] = (y+1)*10 + x+1;
17     }
18   }
19
20   for(int y=0; y<ny; y++)
21   {
22     for(int x=0; x<nx; x++)
23     {
24       printf("%i ",matrix[y][x]);
25     }
26     printf("\n");
27   }
28
29   for(int y=0; y<ny; y++)
30   {
31     free(matrix[y]);
32   }
33   free(matrix);
34 }
```

Das Programm führt zu dieser Ausgabe:

```
11 12 13
21 22 23
```

5.1 Debugger

Ermitteln Sie mittels eines Debuggers, an welchen Speicheradressen diese Variablen stehen: `matrix ,matrix[0],matrix[1],matrix[0][0],matrix[0][1],matrix[0][2],matrix[1][0] ,matrix[1][1],matrix[1][2]`

5.2 Speicherbelegung

Stellen Sie die Speicherbelegung tabellarisch dar:

Adresse	Inhalt an der Adresse	Name der Variablen

5.3 Änderung des Programms

Stellen Sie das Programm auf eine 3-dimensionale Matrix um!