OPR Praktikum

Übung: Grundlagen

Thomas Mahr

2

9. Februar 2016

1

Motivation

Das OPR-Fach setzt Kenntnisse in C voraus. Mit dieser Übung sollen einige Grundlagen wieder-

holt 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 Übersetzungsvorgang1 verstanden haben!

•

•

•

•

Was ist der Präprozessor? Bearbeitet den Quellcode vor der Übersetzung, löscht Kommentare und fügt Textersetzungen ein. (g++ -E Programm.cpp)

Was ist der Compiler? Überprüft auf Fehler und wandelt er die Quellcode-Dateien in binäre Objekt-Dateien (Endung: .o) um

Was ist der Assembler? Erzeugt Maschinencode, der vom CPU ausgefürt warden kann

Was ist der Linker? Verknüpft die einzelnen Programm Komponenten (z.b. Bibliotheken) zu einem ausführbaren Gesamtprogramm zusammen

2

3

. Stellen Sie sicher, dass Sie auf den Rechnern im Praktikumsraum oder auf Ihrem mitge-

brachten Rechner sowohl auf der Kommandozeile als auch in einer integrierten Entwick-

lungsumgebung übersetzen können.

. Stellen Sie sicher, dass Sie auf den Rechnern im Praktikumsraum oder auf Ihrem mitge-

brachten Rechner mit einem Debugger arbeiten können.

1

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

1



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

4

.3 Assembler

•

Stellen Sie durch Setzen eines geeigneten Compiler-Schalters den vom Compiler erzeugten

Assembler-Code dar.

.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.

2

5

Aufgabe: Zeiger und Debugger

Gegeben ist dieses Programm:

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

25

26

27

28

29

30

31

32

33

34

#include<stdio.h>

#include<stdlib.h>

int main()

{

const int nx=3;

const int ny=2;

int\*\* matrix;

matrix = (int\*\*)malloc(ny sizeof(int ));

for(int y=0; y<ny; y++)

{

\*

\*

matrix[y] = (int )malloc(nx sizeof(int));

for(int x=0; x<nx; x++)

{

\*

\*

matrix[y][x] = (y+1)\*10 + x+1;

}

}

for(int y=0; y<ny; y++)

{

for(int x=0; x<nx; x++)

{

printf("%i ",matrix[y][x]);

}

printf("\n");

}

for(int y=0; y<ny; y++)

{

free(matrix[y]);

}

free(matrix);

}

Das Programm führt zu dieser Ausgabe:

1

2

1 12 13

1 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:

3

5

.3 Änderung des Programms

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

4

Adresse Inhalt an der Adresse Name der Variablen

Ein Bild, das Text, Screenshot, Schrift enthält.

Automatisch generierte Beschreibungdfdafdasf

matrix

(matrix[0])

(matrix[1])

(matrix[0][0])

(matrix[0][1])

(matrix[0][2])

(matrix[1][0])

(matrix[1][1])

(matrix[1][2])

0x798220

0x798260

0x7984c0

11

12

13

21

22

23

0x5ffe68

0x798220

0x798228

0x798260

0x798264

0x798268

0x7984c0

0x7984c4

0x7984c8

