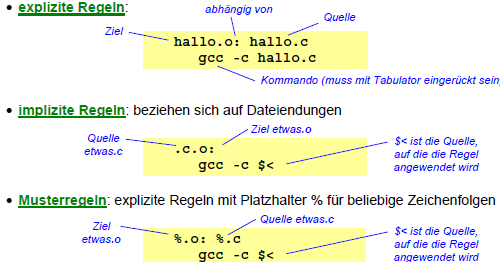
*Typ* \**Zeigername* **= (***Typ***\*)** malloc**(sizeof (***Typ***));***Typ* \**Zeigername* = (*Typ*\*) calloc(*Feldgröße*, sizeof (*Typ*));

|  |  |  |
| --- | --- | --- |
| **enum month {jan = 1, feb, mar, apr, may, jun, jul, aug, sep, oct, nov, dec};**  **enum month aMonth = mar;**  **enum int\_or\_string {type\_int, type\_string};**  **struct** struct\_with\_union  **{enum** int\_or\_string u\_type**;**  **union**  **{int** i**; char** \*s**;}** u**;};** | **struct struct\_with\_union x;**  x**.**u\_type **=** type\_int**;**  x**.**u**.**i **= 1;**  x**.**u\_type **=** type\_string**;**  x**.**u**.**s **= "Hallo";**  **→ bei pointern auf struct**  **. bei struct an sich** | **#include <stdio.h>**  **#include <stdlib.h> /\* calloc, malloc, free... \*/**  **#include <stddef.h> /\* NULL, size\_t, ... \*/**  **#include <string.h>**  **#include <fcntl.h> /\* *open, O\_RDONLY, O\_WRONLY, O\_CREAT, O\_EXCL* \*/**  **#include <**sys**/**stat**.**h**> /\*** *mode\_t, S\_IRUSR | S\_IWUSR | S\_IRGRP | S\_IROTH* **\*/**  **#include <**unistd**.**h**> /\*** *read, write* **\*/**  **#include <**errno**.**h**> /\*** *errno* **\*/** |
| ***argv Feld von String-Pointern (argument vector).***  *argc* ***+ 1*** *Feldgröße (argument count).*  *argv****[0]*** *Programm-Name (Kommando)*  *argv****[1]*** *erstes Kommandozeilen-Argument*  *argv****[****argc* ***– 1]*** *letztes Kommandozeilen-Argument*  *argv****[****argc****]*** *0 (NULL-Pointer)* | Final = const  private = static  int main(int argc, char \*argv[])  scanf(„%s“, &bla);  → & nicht bei arrays[ ],oder \*\*  int a;  int \*b;  a = 1;  b = &a; | ***#ifndef NAME\_H***  ***#define*** *NAME\_H*  ***... /\**** *Inhalt der Header-Datei Name.h* ***\*/***  ***#endif***  ***#include “header.h“***  void test(int \*x, int (\*test\_in\_test) (const char\*))  {x = test\_in\_test(„blah“);} |

|  |  |
| --- | --- |
| Fehlerbehandlung:  int feof(FILE \*fp);liefert einen von 0 verschiedenen Wert, wenn das Eingabeende erreicht wurde  int ferror(FILE \*fp); liefert einen von 0 verschiedenen Wert, wenn ein Fehler aufgetreten ist  void perror(const char \*prefix); gibt prefix gefolgt von der Fehlermeldung des aktuellen Fehlers auf stderr aus  void clearerr(FILE \*fp); Setzt den Eingabeende- und Fehlerzustand zurück | statische lib: ar rs libbeispiel.a function.o global.o  dynamische lib: gcc -shared function.o global.o -o libbeispiel.so  binden: gcc localglobalvar.o -L. –lbeispiel -o localglobalvar  archiv: tar czf beispiel.tar.gz localglobalvar libbeispiel.so |



|  |  |
| --- | --- |
| FLAGS = -g -W -Wall -ansi -pedantic  PREPROCESSOR = gcc -E  COMPILER = gcc -c  LINKER = gcc  OBJECTS = \  localglobalvar.o \  function.o \  global.o  all: localglobalvar  clean:  rm -f localglobalvar $(OBJECTS) $(OBJECTS:.o=.i)  localglobalvar: $(OBJECTS)  $(LINKER) $(FLAGS) -o localglobalvar $(OBJECTS)  localglobalvar.o: localglobalvar.i  $(COMPILER) $(FLAGS) localglobalvar.i | function.o: function.i  $(COMPILER) $(FLAGS) function.i  global.o: global.i  $(COMPILER) $(FLAGS) global.i  localglobalvar.i: localglobalvar.c function.h global.h  $(PREPROCESSOR) $(FLAGS) localglobalvar.c -o localglobalvar.i  function.i: function.c function.h global.h  $(PREPROCESSOR) $(FLAGS) function.c -o function.i  global.i: global.c global.h  $(PREPROCESSOR) $(FLAGS) global.c -o global.i  # Makefile fuer Softwaresystem hallohallo  PACKAGES=hallo hallo2  all clean:  for p in $(PACKAGES); do \  (cd $$p && $(MAKE) $@); \  done |

|  |  |  |
| --- | --- | --- |
| **unterschiede java, c++** | **Unterschiede c, c++** | **Unterschiede c, java** |
| in C++ sind Klassen auch als Werttyp verwendbar,  in C++ Operator-Overloading möglich, in C++ kein Garbage-Collector, Exceptions in Java  in C++ muss man Operatoren für Klassen selber schreiben | in C++ gibt es Klassen  in C++ Konstruktoren und Destruktoren, in C++ Datenkapselung , in C++ kann man Methoden überladen, in C++ gibt es namespaces | ANSI-C Zeiger bieten sehr viel mehr Möglichkeiten als Java Referenzen  ANSI-C kennt keinen echten Feld-Typ  ANSI-C kennt keinen echten String-Typ |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Absturz:** unerwartetes Programmende, z.B. wegen Speicherzugriffsfehler  **Endlosschleife:** das Programm scheint zu "hängen", aber es läuft und läuft und läuft ...  **Speicherüberlauf:** der ganze Rechner wird langsam,weil das Programm sämtlichen Speicher belegt hat  **Fehlverhalten:** das Programm tut nicht, was es tun soll, liefert z.B. falsche Ergebnisse | 1) Reproduzierbarkeit finden,Debugger:*Programm mit gleichen Eingaben im Debugger laufen lassen oder core-Datei untersuchen*  *2) Programm im Debugger unterbrechen*  *oder Programm "abschießen" (kill -6 ...), um untersuchbare core-Datei zu erhalten*  *3)valgrind:speicherverwaltungstool: wo viel speicher reserviert wird → fehlerquelle*  *4) Fehler reproduzieren und isolieren(Blackbox), Debugger: mit breakpoints untersuchen(whitebox)*  **#include <**iostream**> //** *std::cout, std::cin, std::hex, std::endl, operator<<, operator>>*  **int main()**  **{**  std**::**cout **<< "**Dezimalzahl eingeben: **";**  **int** zahl**;**  std**::**cin **>>** zahl**;**  std**::**cout **<< "**Hexadezimalzahl: **" <<** std**::**hex **<<** zahl **<<** std**::**endl**;}**   |  |  |  | | --- | --- | --- | |  | ILP32 | LP64 | | char | 8 | 8 | | short | 16 | 16 | | int | 32 | 32 | | long | 32 | 64 | | long long | 64 | 64 | | float | 32 | 32 | | double | 64 | 64 | | Pointer | 32 | 64 | |

|  |  |  |
| --- | --- | --- |
| [int](http://clc-wiki.net/wiki/int) [strcmp](http://clc-wiki.net/wiki/strcmp)([const](http://clc-wiki.net/wiki/const) [char](http://clc-wiki.net/wiki/char)\* s1, [const](http://clc-wiki.net/wiki/const) [char](http://clc-wiki.net/wiki/char)\* s2)  {  [while](http://clc-wiki.net/wiki/while)(\*s1 && (\*s1==\*s2))  s1++,s2++;  [return](http://clc-wiki.net/wiki/return) \*([const](http://clc-wiki.net/wiki/const) [unsigned](http://clc-wiki.net/wiki/unsigned) [char](http://clc-wiki.net/wiki/char)\*)s1-\*([const](http://clc-wiki.net/wiki/const) [unsigned](http://clc-wiki.net/wiki/unsigned) [char](http://clc-wiki.net/wiki/char)\*)s2;}  char \*s; const char \*t;  u = (char\*)malloc(strlen(s)+strlen(t)+1);  if(u!=NULL){  strcat(strycpy(u,s),t); | void strcpy2(char \*dest[], const char \*source[]) {  int i = 0;  while ((dest[i] = source[i]) != '\0') {  i++;  } } | char \*strcat(char \*dest, const char \*src)  {  size\_t i,j;  for (i = 0; dest[i] != '\0'; i++)  ;  for (j = 0; src[j] != '\0'; j++)  dest[i+j] = src[j];  dest[i+j] = '\0';  return dest;  } |