KV Betriebssysteme
Einführung in C

KV Betriebssysteme Preprocessor

DI. Dr. Peter René Dietmüller

CIS-Software - Peter René Dietmüller
Lorcherstraße 2a, 4470 Enns

Preprocessor

- In C wird vor dem eigentlichen Compiler ein sogenannter Präprozessor gestartet.
- Der Präprozessor wertet bestimmte Kommandos im Code aus und manipuliert den Quellcode. Der C-Compiler selbst bekommt bereits modifizierten Quellcode.
- Präprozessor-Anweisungen sind durch ein vorangestelltes # zu erkennen.
- Präprozessor-Anweisungen haben am Ende keinen Strichpunkt!

#define, #undef (I)

- **Bedeutung**
  - Mit #define können in C Konstanten nachgeahmt werden. Durch #define wird der Präprozessor angewiesen alle Vorkommen des definierten Symbols durch den angegebenen Text zu ersetzen.
  - Texte innerhalb von Zeichenketten werden durch den Preprocessor nicht verändert.
- **Syntax**
  - #define symbol text
  - #undef symbol

#define, #undef (II)

- **Beispiel**
  - #define MAX 100
  - Definiert das Symbol MAX und weist den Preprozessor an, alle Vorkommen von MAX durch den Wert 100 zu ersetzen. Ausgenommen davon sind Zeichenketten und Kommentare.
- **TRUE und FALSE sind nicht definiert. Meist werden sie mit einem #define folgendermaßen deklariert.**
  - #define FALSE 0
  - #define TRUE 1
  - #define TRUE (!FALSE)

#define, #undef (III)

Quellcode

```
#define  MAX    100
#define  FALSE  0
#define  TRUE   (!FALSE)

int tiles[MAX];
...
for (i = 1; i <= MAX; i++)
    tiles[i] = TRUE;
printf("MAX=%d", MAX);
```

Nach Preprocessor

```
int tiles[100];
...
for (i = 1; i <= 100; i++)
    tiles[i] = (!0);
printf("MAX=%d", 100);
```


Konstanten

- **2 Möglichkeiten**
  - #define
  - Schlüsselwort const
- **Schlüsselwort const**
  - Variablen können vor Änderung geschützt werden.
  - „const“ wird vor allem für Parameter verwendet.
  - Es handelt sich nicht um Konstanten im klassischen Sinn, weil Speicherplatz angelegt wird.
- **Beispiele**
  - const int a = 1;
  - #define K 2

KV Betriebssysteme
Einführung in C

#include

- **Bedeutung**
  - Inkludiert die angegebene Datei.
- **Syntax**
  - #include <xxx.h>
  - #include "xxx.h"
- **Beispiel**
  - #include <stdio.h>
  - #include "module.h"
  - Inkludiert die Datei stdio.h aus dem INCLUDE-Verzeichnis des Compilers und die Datei module.h aus dem aktuellen Verzeichnis.

Importieren von Modulen (I)

- Geschieht durch das Inkludieren von Header-Files (Extension .h), in denen die Schnittstelle eines Moduls definiert ist. Das Modulkonzept wird später noch genauer erklärt.
- Um zum Beispiel das Modul STDIO (Standard Input/Output) zu importieren, muß man am Beginn des Programms folgende Zeile aufnehmen.
  - #include <stdio.h>

Importieren von Modulen (II)

```
/* -- 100 Mal Würfeln -- */
#include <stdlib.h>
#include <stdio.h>

int main(void) {
   int i;
   srand(0);                    /* Zufallszahlen init. */
   printf("100 Mal Würfeln:\n");
   for (i = 0; i < 100; i++)
      printf("%2d", rand() % 6 + 1);
   return 0;
}
```


#ifdef, #ifndef, #else, #endif (I)

- **Bedeutung**
  - Mit diesen Preprocessorbefehlen kann eine bedingte Compilierung durchgeführt werden. Das heißt Teile des Codes können ausgeblendet werden.
- **Syntax**
  - #ifdef symbol /* bzw. #if defined(symbol) */
  - #ifndef symbol /* bzw. #if !defined(symbol) */
  - #endif
  - if-Abfrage auf Definition bzw. Nicht-Definition; Code nach ifdef und vor endif wird nur übersetzt, falls symbol definiert ist.

#ifdef, #ifndef, #else, #endif (II)

- **Beispiel**
  - char version[100];

```
#ifdef WIN2000
    strcpy(version, "Windows 2000");
#end
#ifdef WINNT
    strcpy(version, "Windows NT 4.0");
#end
#ifdef WIN98
    strcpy(version, "Windows 98");
#end
```


KV Betriebssysteme
Array

DI. Dr. Peter René Dietmüller

CIS-Software - Peter René Dietmüller
Lorcherstraße 2a, 4470 Enns

KV Betriebssysteme
Einführung in C

Array - Modell

Vornamen

0	Christian
1	Istvan
2	Johannes
3	Günther
4	Martin
	...
N-1	Peter

Syntax

- **Deklaration**
  - **BaseType Identifier "["Elements"]" {"["Elements"]"}**
- **Beispiele**
  - **int werte[100]; char alphabet[256];**
- **Zugriff**
  - **ArrayName[index]**
- **Beispiele**
  - **c = alphabet[255]; i = werte[1];**
  - **aber: c = alphabet[256]; /* ?? */**

Regeln

- **Die Anzahl der Elemente muß bei der Definition durch einen konstanten Ausdruck bestimmt sein (auch in Verbindung mit dem sizeof-Operator)**
- **Es gibt zur Laufzeit keine Möglichkeit die Anzahl der Elemente eines Arrays zu bestimmen! (siehe aber sizeof-Operator)**
- **Der erste Index eines Arrays ist immer 0! Die zehn Elemente eines Arrays test[10] sind also test[0], test[1], ..., test[9]!**
- **Zuweisung eines Arrays an ein anderes resultiert NICHT in einem Kopieren der Elemente.**

Initialisierung

- **char hex[16];**
- **hex[0] = '0'; hex[1] = '1'; ... hex[15] = 'F';**

- **char hex[16] = {'0', '1', '2', '3', '4', '5', '6', '7', '8', '9', 'A', 'B', 'C', 'D', 'E', 'F'};**

- **int values[3] = {1, 2}; /* Rest wird 0 gesetzt! */**

Parameter (I)

- **Arrays werden als Parameter entgegen den C-Gepflogenheiten Call-by-Reference übergeben (eigentlich: Call-by-Value Übergabe der Array-Adresse!)**
- **Wenn es nicht dem Anlegen von Speicherplatz für einen Array dient, so kann die Größe der ersten Dimension offen gelassen werden (Open Array), z.B. bei Arrays als Funktionsparametern**

Parameter (II)

```
void ArrayAenderung(int werte[]) {
    werte[1] = 5;
}

void ArrayTest(void) {
    int werte[11] = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10};
    printf("%d", werte[1]);          /* 1 */
    ArrayAenderung(werte);
    printf("%d", werte[1]);          /* 5 */
}
```

Parameter (III)

```
int writeName (char name[]) {
      /* 'name' .. offenes, eindimensionales Array */
}

int getPairs (int p[][2]) {
      /* 'p' .. offenes, zweidimensionales Array */
}
```


Array - Mehrdimensional

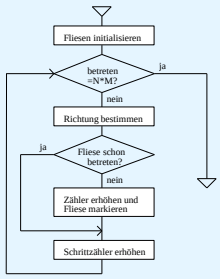
- Mehrdimensionale Arrays werden so gespeichert, daß sich jeweils der letzte Index am schnellsten ändert ("Reihen-Spalten-Ordnung")
- Bei Deklaration und Zugriff muß jede Dimension in eigene Klammern "[" und "]" gestellt werden ([i1][i2], nicht [i1,i2])!
- Beispiel
  - int pairs[4][3] = {{1,2},{3,4}}; /* Rest wird 0 */
  - Speicherabbild

[0][0]	[0][1]	[0][2]	[1][0]	[1][1]	..	[3][2]

Array - Beispiel (I)

- Aufgabenstellung
  - Ein Käfer torkelt in einem rechteckigen Raum mit N*M Fliesen. In jedem Zeitschritt tritt er auf irgendeine der Nachbarfliesen (wenn er an die Wand stößt, bleibt er auf seiner Fliese).
  - Gesucht ist die Anzahl der Schritte, die der Käfer braucht, um alle Fliesen mindestens einmal zu betreten.
- Quellcode
  - Source\Kaefer.c

Array - Beispiel (II)



KV Betriebssysteme
Datenstrukturen

DI. Dr. Peter René Dietmüller

CIS-Software - Peter René Dietmüller
Lorcherstraße 2a, 4470 Enns

Strukturen (I)

- Bedeutung
  - Zusammenfassung verschiedener Datentypen
  - Jedes Feld wird durch einen Namen referenziert.
- Deklaration
  - struct [tag] { member-list } [declarators];
  - [struct] tag declarators;
  - „tag“ ist der Name eines Datentyps.
  - „declarators“ sind die Namen von Variablen.
- Zugriff
  - Variablenname "." Feldname

KV Betriebssysteme
Einführung in C

Strukturen (II)

- **Beispiel**
  - struct TADDRESS {
    char szStreet[30];
    int iNumber;
    int iZip;
    char szCity[20];
    };

    struct TADDRESS aAddress =
    {"Altenbergerstraße", 69, 4040, "Linz"};

    aAddress.iZip = 4020;

Strukturen (III)

- **Beispiel zu Arrays und Strukturen**
  - struct TADDRESS {
    char szStreet[30];
    int iNumber;
    int iZip;
    char szCity[20];
    };

    struct TADDRESS address[20];

    address[0].iZip = 1010;

Strukturen (IV)

- **Operationen mit Strukturen**
  - Übergabe an Funktion (Call by value)
  - Rückgabe aus Funktion
  - Zuweisung an eine Struktur (Kopie)
  - Anwendung von Adress- und sizeof-Operator
- **Alignment**
  - Ausrichtung der Daten im Speicher
  - struct { char c; long l; } TEST;
    printf("%ld\n", sizeof(struct TEST));
  - Dieses Beispiel kann 5, 6 oder 8 liefern.

Bitfelder

- **Bedeutung**
  - Aufteilung eines Feldes in einzelne Bits / Bitgruppen
- **Beispiel**
  - struct CELL // Declare CELL bit field
    {
    unsigned character : 8; // 00000000 ????????
    unsigned foreground : 3; // 00000??? 00000000
    unsigned intensity : 1; // 0000?000 00000000
    unsigned background : 3; // 0???0000 00000000
    unsigned blink : 1; // ?0000000 00000000
    } screen[25][80]; // Array of bit fields

Unions

- **Bedeutung**
  - Zusammenfassung verschiedener Datentypen
  - Jedes Feld wird durch einen Namen referenziert.
  - Alle Felder belegen denselben Speicherplatz!
- **Syntax**
  - union [tag] { member-list } [declarators];
  - [union] tag declarators;
- **Beispiel**
  - union TEST { short sZahl; long lZahl; };
    uTest.lZahl = 65537;
    printf("%d\n", uTest.sZahl); /* ?? */

Aufzählungen

- **Bedeutung**
  - Deklaration einer vordefinierten Menge von Werten
  - Jeder Wert bekommt einen Namen.
  - Der erste Wert ist 0.
- **Syntax**
  - enum [tag] {enum-list} [declarator];
  - enum tag declarator;
- **Beispiele**
  - enum tage { mon, die, ... } woche;
  - enum tage { mon = 1, die, ... } woche;
  - enum boolean { FALSE, TRUE };

Typdeklaration

IS SOFTWARE

- **Bedeutung**
  - **Für einen Typ wird ein Synonym vergeben.**
- **Syntax**
  - **typedef type-declaration synonym;**
- **Beispiele:**
  - **typedef struct {**
    char strasse[30];
    char ort[20];
    } TADRESSE;
    TADRESSE adresse;
  - **typedef int (*funcptr)();**