Informatik II

Oliver Jack

Fachhochschule Jena
Fachbereich Elektrotechnik und Informationstechnik

Sommersemester 2010

Inhalt

Vorlesung 11. Exkurs: Der C-Präprozessor

Makros und der C-Präprozessor

- Bedingte Compilierung / Include Guards
- Beispiel Quicksort

Vordefinierte Makros

- Operatoren und Direktiven
- Nochmal Quicksort

Prüfung auf definierte Markros

```
#ifdef bezeichner
/* falls bezeichner definiert ist */
...
#else
/* falls bezeichner nicht definiert ist */
...
#endif

#ifndef bezeichner
/* falls bezeichner nicht definiert ist */
...
#else
/* falls bezeichner definiert ist */
...
#endif
```

Prüfung auf gültige Ausdrücke

```
#if konst_ausdruck
/* falls konst_ausdruck TRUE (!=0) ist */
...
#elif konst_ausdruck
/* nur ANSI-C */
...
#else
/* falls konst_ausdruck FALSE (==0) ist */
...
#endif
```

In konst_ausdruck ist möglich:

```
defined ( bezeichner )
```

Selber Effekt wie #ifdef

Benutzung bedingter Compilierung...

```
/* printExample.c */
#include <stdio.h>
#define EX(k) k+3.14159
#define PR(a) printf("%d␣␣",(int)(a))
#define PRINT(a) PR(a); putchar('\n')
#define PRINT2(a,b) PR(a), PRINT(b)
#define PRINT3(a,b,c) PR(a); PRINT2(b,c)
#define MAX(a,b) (a<b ? b : a)
```

...Benutzung bedingter Compilierung

```
int main() {                              /* Praeprozessor  */
  int i,x=1,y=2;
  PRINT( 2*EX(2) );
  for( i=0; i<=4; i+=2 )
    PRINT2( i, 5*i+32 );
#ifndef PRX
  PRINT3( MAX(x++,y),x,y );
  PRINT3( MAX(x++,y),x,y );
#else
  PRINT3( MAX(x++,++y),x,y );
  PRINT3( MAX(x,y),x,y );
#endif
  return 0;
}
```

Präprozessorausgabe

gcc -E printExample.c (#ifndef PRX == true)

```
int main() {
  int i,x=1,y=2;
  printf("%d␣␣",(int)(2*2 +3.14159));
  putc('\n', stdout);
  for( i=0; i<=4; i+=2 )
      printf("%d␣␣",(int)(i)), printf("%d␣␣",(int)(5*i+32));
      putc('\n', stdout);
  printf("%d␣␣",(int)((x++<y ? y : x++)));
  printf("%d␣␣",(int)(x)), printf("%d␣␣",(int)(y));
  putc('\n', stdout);
  printf("%d␣␣",(int)((x++<y ? y : x++)));
  printf("%d␣␣",(int)(x)), printf("%d␣␣",(int)(y));
  putc('\n', stdout);
  return 0;
}
```

Präprozessorausgabe

gcc -E -DPRX printExample.c (#indef PRX == false)

```
int main() {
  int i,x=1,y=2;
  printf("%d␣␣",(int)(2*2 +3.14159));
  putc('\n', stdout);
  for( i=0; i<=4; i+=2 )
      printf("%d␣␣",(int)(i)), printf("%d␣␣",(int)(5*i+32));
      putc('\n', stdout);
  printf("%d␣␣",(int)((x++<++y ? ++y : x++)));
  printf("%d␣␣",(int)(x)), printf("%d␣␣",(int)(y));
  putc('\n', stdout);
  printf("%d␣␣",(int)((x<y ? y : x)));
  printf("%d␣␣",(int)(x)), printf("%d␣␣",(int)(y));
  putc('\n', stdout);
  return 0;
}
```

Das Problem der mehrfachen Inklusion von Header-Dateien

- Datei grandfather.h

```
struct foo {
    int member;
};
```

- Datei father.h

```
#include "grandfather.h"
```

- Datei child.c

```
#include "grandfather.h"
#include "father.h"
```

- Problem: child.c inkludiert zweimal die Header-Datei grandfather.h, was einen Compiler-Fehler verursacht, da `struct foo` doppelt definiert ist.

Benutzung von Include Guards

- Datei grandfather.h

```
#ifndef H_GRANDFATHER
#define H_GRANDFATHER

struct foo {
    int member;
};

#endif
```

- Datei father.h

```
#include "grandfather.h"
```

- Datei child.c

```
#include "grandfather.h"
#include "father.h"
```

- Die erste Inklusion von grandfather.h definiert das Macro `H_GRANDFATHER`.
- Wenn child.h die Datei grandfather.h ein zweites mal inkludiert, schlägt der Test `#ifndef` fehl.
- Der Präprozessor ignoriert alles bis zum `#endif`.
- Keine mehrfache Definition von `struct foo`, das Programm wird korrekt kompiliert.

Quicksort

```
#include <stdlib.h>
/* Definiert Prototyp der Quicksort-Funktion */
void qsort(void * base, size_t num, size_t size,
int (* comparator)(const void *, const void *));
```

Argumente

1. zu sortierendes Feld,
2. Anzahl der Elemente des Feldes,
3. Anzahl der Bytes für jedes Feldelement,
4. Vergleichsfunktion für die Feldelemente.

Vergleichsfunktion

```
int compare(const void *, const void *)
```

Argumente

1. Zeiger auf erstes zu vergleichendes Feldelement,
2. Zeiger auf zweites zu vergleichendes Feldelement.

Rückgabewert

- Zahl kleiner null, falls erstes Element kleiner als zweites Element,
- null, falls beide Elemente gleich,
- Zahl größer null, falls erstes Element größer als zweites Element.

Testprogramm für qsort()

Zunächst ohne Makros

Funktionsweise

1. Feld mit Zahlen füllen,
2. Feld ausgeben,
3. Feld mit qsort() sortieren,
4. Feld ausgeben.

Programm (Vorspann)

```
#include <stdio.h>
#include <stdlib.h> /* qsort() und Zufallszahlen */
#include <time.h> /* fuer Zufallszahlengenerator */

#define    N    11    /* Groesse des Felds */

typedef enum {before, after} when;

/* Vergleichsfunktion */
int cmp(const void *vp, const void *vq);
void fill_array(double *a, int n);
void prn_array(when val, double *a, int n);
```

Programm (Hauptprogramm)

```
int main(void) {
  double a[N];

  fill_array(a, N);
  prn_array(before, a, N);
  qsort(a, N, sizeof(double), cmp);
  prn_array(after, a, N);
  return 0;
}
```

Programm (Vergleichsfunktion)

```
int cmp(const void *vp, const void *vq) {
  const double *p = vp;
  const double *q = vq;
  double diff = *p - *q;

  return ((diff <= 0.0) ?
          ((diff < 0.0) ? -1 : 0) : +1);
}
```

Programm (Feld füllen)

```
void fill_array(double *a, int n) {
  int i;

  srand(time(NULL)); /* rand() initialisieren */
  for (i = 0; i < n; ++i)
    a[i] = (rand() % 1001) / 10.0;
}
```

Programm (Feld ausgeben)

```
void prn_array(when val, double *a, int n) {
  int i;

  printf("%s\n%s%s\n", "---",
      ((val == before) ? "Before␣" : "After␣"),
      "sorting:");
  for (i = 0; i < n; ++i) {
    if (i % 6== 0)
      putchar('\n');
    printf("%8.1f", a[i]);
  }
  putchar('\n');
}
```

Ausgabe

```
---
Before sorting:

    69.5     23.1      0.8     14.9     37.5
    79.7     40.3     56.5     20.2     94.2
    28.2
---
After sorting:

     0.8     14.9     20.2     23.1     28.2
    37.5     40.3     56.5     69.5     79.7
    94.2
```

Programm mit Makros (Header sort.h)...

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>

#define M    24         /* Groesse von a[] */
#define N     8         /* Groesse von b[] */
/* Diesmal gem. Nachkommastellen sortieren */
#define fractional_part(x)    (x - (int) x)
#define random_char()      (rand() % 26 + 'a')
#define random_float()     (rand() % 100 / 10.0)
```

...Programm mit Makros (Header sort.h)...

Entweder Feld mit Zeichen oder mit Gleitkommazahlen füllen

```
#define   FILL(array, sz, type)      \
   if (strcmp(type, "char") == 0)    \
      for (i = 0; i < sz; ++i)       \
         array[i] = random_char();   \
   else                              \
      for (i = 0; i < sz; ++i)       \
         array[i] = random_float()
```

...Programm mit Makros (Header sort.h)

```
#define    PRINT(array, sz, fmt_string)  \
   for (i = 0; i < sz; ++i)                \
      printf(fmt_string, array[i]);        \
   putchar('\n')

/* Vergleichsfunktionen fuer qsort() */
int compare_fractional_part(const void *,
                            const void *);
int lexico(const void *, const void *);
```

Vergleichsfunktionen compare.h

```
#include "sort.h"

/* Vergleich der Nachkommastellen */
int compare_fractional_part(const void *vp,
                            const void *vq) {
  const float *p = vp, *q = vq;
  float x;

  x = fractional_part(*p) - fractional_part(*q);
  return ((x < 0.0) ? -1 : (x == 0.0) ? 0 : +1);
}
```

Vergleichsfunktionen compare.h

```
/* Lexikorafischer Vergleich */
int lexico(const void *vp, const void *vq) {
  const char *p = vp, *q = vq;
  return (*p - *q);
}
```

Programm mit Makros (Hauptprogramm)

```
#include "sort.h"
int main(void) {
    char a[M]; float b[N]; int i;
    srand(time(NULL));
    FILL(a, M , "char");
    PRINT(a, M, "%-2c");
    qsort(a, M, sizeof(char), lexico);
    PRINT(a, M, "%-2c");
    printf("---\n");
    FILL(b, N, "float");
    PRINT(b, N, "%-6.1f");
    qsort(b, N, sizeof(float),
          compare_fractional_part);
    PRINT(b, N, "%-6.1f");
    return 0;
}
```

Ausgabe

```
x k g u o y h d b r g m u n o e g g h j i v d p
b d d e g g g g h h i j k m n o o p r u u v x y
---
9.6   7.2   6.6   3.3   5.7   3.5   7.1   3.1
7.1   3.1   7.2   3.3   3.5   6.6   9.6   5.7
```

Operator

Beispiel

```
#define message_for(a, b) \
  printf(#a " and " #b ": We love you!\n")
int main(void) {
  message_for(Carole, Debra);
  return 0;
}
```

Makroexpansion („stringization“)

```
printf("Carole" " and " "Debra" " : We love you!\n"
/* entspricht */
printf("Carole and Debra : We love you!\n")
```

Operator

Beispiel

```
#define X(i) x ## i
X(1) = X(2) = X(3);
```

Makroexpansion („merge“)

x1 = x2 = x3;

Fest vordefiniert in ANSI C

__LINE__ Zeilenmummer der aktuellen Zeile in Programm,

__FILE__ Name der Programmdatei,

__DATE__ Übersetzungsdatum der Programmdatei,

__TIME__ Übersetzungszeit der Programmdatei,

__STDC__ Erkennungsmerkmal eines ANSI C-Compilers. Ganzzahlige Konstante. Falls Wert 1 ist, handelt es sich um einen ANSI C-konformen Compiler.

Direktive #error

- Erzeugung eines Compiler-Fehlers,
- Ausgabe des entsprechenden Textes.

Beispiel

```
#define A_SIZE 5
#define B_SIZE 7
/*
...
*/
#if A_Size < B_Size
  #error "Incompatible␣sizes"
#endif
```

Benutzung vordefinierter Makros

```
#if __STDC__ == 1
#define COMPILED_ON __DATE__ "␣" __TIME__
#else
#error Myprog requires a conforming \
Standard C compiler, \
this one is not
#endif
```

Makro assert()

Motivation

- Unterstüztung der Fehlersuche bei der Programmentwicklung.
- Überprüfung von Zusicherungen.
- Systemabhängig implementiert.

```
#include <assert.h>
void f(char *p, int n) {
  ...
  assert(p != Null);
  assert(n > 0 && n < 7);
  ...
}
```

Eine mögliche assert()-Definition

```
#include <stdio.h>
#include <stdlib.h>
#if defined(NDEBUG)
  #define assert(ignore) ((void) 0) /*ignorieren */
#else
  #define assert(expr) \
    if (!(expr)) { \
      printf("\n%s%s\n%s%s\n%s%d\n\n", \
        "Assertion failed: ", #expr, \
        "in file ", __FILE__, \
        "at line ", __LINE__); \
      abort(); \
    }
#endif
```

Direktive #undef

Motivation

- Rücknahme einer Makrodefinition.
- Benutzung von Funktionen statt Makros.

Beispiel

- int isalpha(int c) existiert als Makro und Funktion (ctype.h)
- Benutzung der Funktion

```
#include <ctype.h>
#undef isalpha
if (isalpha(c) { /*...*/ }
```

- Alternativ: (isalpha)(c);

Direktive #line...

Motivation

- Anzeige von Zeilennummern und Dateinamen des Quelltexts bei Codegeneratoren.
- „Umnummerierung“ der Quelltextzeilennummern.

```
#line Ganzzahlkonstante "Dateiname"
```

Dateiname ist optional

...Direktive #line...

Beispiel cweb (literate programming)

Datei easter.w

```
01 @ Print Easter Sunday date. E.g., Easter Sunday 1999 ...
02 ``{\tt 04.04.1999}''.
03 @c
04 void printedate(edate date) {
05   printf("%02d.%02d.%d\n", date.day, date.month, date.yearr);
06 }
```

Datei easter.c (ctangle easter.w)

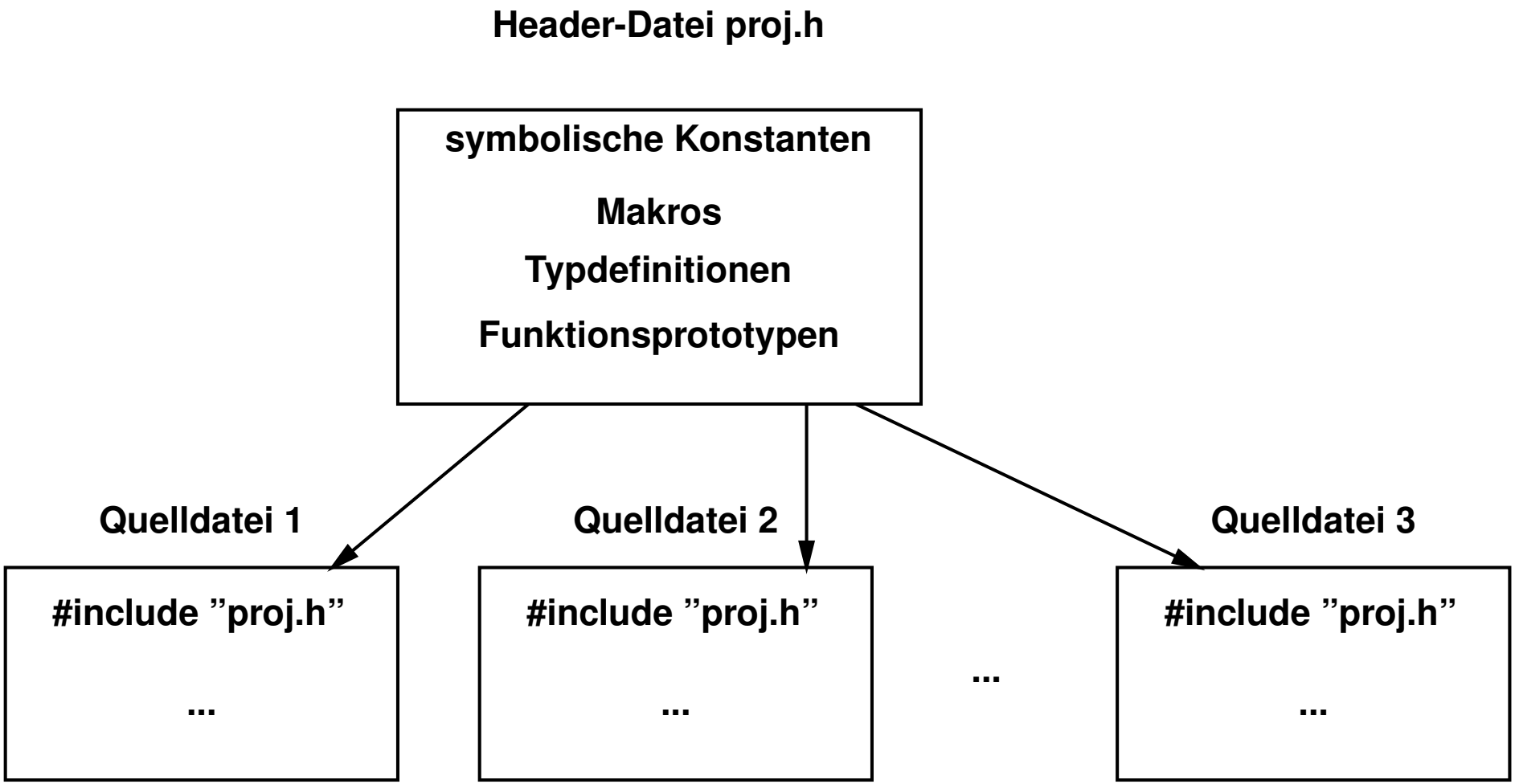
```
12 /*:2*//*4:*/
13 #line 103 "./easter.w"
14
15 void printedate(edate date){
16 printf("%02d.%02d.%d\n",date.day,date.month,date.yearr);
17 }
```

...Direktive #line

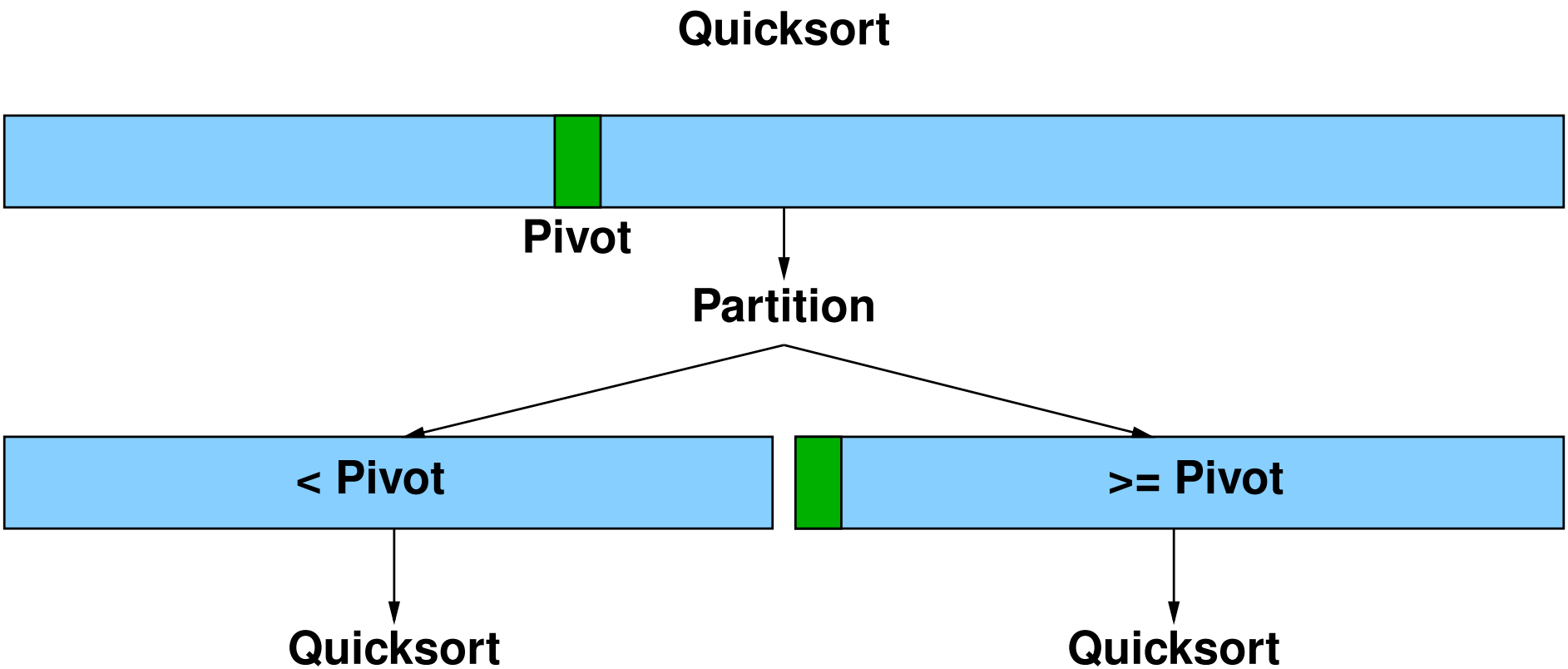
Fehlermeldung (gcc easter.c)

```
./easter.w: In function ‘printedate’:
./easter.w:105: error: structure has \
                  no member named ‘yearr’
```

Organisation eines C-Programms



Prinzip


Quicksort
Pivot
Partition
< Pivot
>= Pivot
Quicksort
Quicksort

Vorspann

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

#define   N   300
#define swap(x, y) { int t; t = x; x = y; y = t; }
#define order(x, y) if (x > y) swap(x, y)
#define o2(x, y) order(x, y)
#define o3(x, y, z) o2(x, y); o2(x, z); o2(y, z)

typedef   enum {yes, no}   yes_no;

static yes_no   find_pivot(int *left, int *right,
                           int *pivot_ptr);
static int *partition(int *left, int *right,
                      int pivot);
void quicksort(int *, int *);
```

Hauptprogramm

```
int main(void) {
    int    a[N], i;

    srand(time(NULL));
    for (i = 0; i < N; ++i)
        a[i] = rand() % 1000;
    quicksort(a, a + N - 1);
    for (i = 0; i < 7; ++i)
        printf("%4d", a[i]);
    printf("␣.....");
    for (i = N - 8; i < N; ++i)
        printf("%4d", a[i]);
    putchar('\n');
    return 0;
}
```

Quicksort

```
void quicksort(int *left, int *right)
{
    int    *p, pivot;

    if (find_pivot(left, right, &pivot) == yes) {
        p = partition(left, right, pivot);
        quicksort(left, p - 1);
        quicksort(p, right);
    }
}
```

Pivot finden...

```
static yes_no find_pivot(int *left, int *right,
                         int *pivot_ptr) {
   int   a, b, c, *p;

   a = *left;  /* Wert links */
   b = *(left + (right - left) / 2) /* Wert mitte *
   c = *right;   /* Wert rechts */
   o3(a, b, c);  /* sortieren */
   if (a < b) {  /* pivot gefunden */
      *pivot_ptr = b;
      return yes;
   }
```

...Pivot finden

```
    if (b < c) { /* pivot gefunden */
        *pivot_ptr = c;
        return yes;
    }
    for (p = left + 1; p <= right; ++p)
        if (*p != *left) {
         *pivot_ptr = (*p < *left) ? *left : *p;
         return yes;
        }
    return no;  /* alle Werte gleich */
}
```

Partitionieren

```
static int *partition(int *left, int *right,
                      int pivot) {
    while (left <= right) {
        while (*left < pivot)
         ++left;
        while (*right >= pivot)
         --right;
        if (left < right) {
         swap(*left, *right);
         ++left;
         --right;
        }
    }
    return left;
}
```

Programmausgabe

```
0    2   13   15   18   18   25 ..... 971 973 973 978 984
```