Fachhochschule Bingen

Programmieren 2

Organisation: Klausur, Studienleistung
Wiederholung & Ergänzung: Macros
ALDA-Exkurs: Sortieren

Prof. Dr. Maximilian Mengel,
Professur Programmiermethodik,
Grundlagen der Informatik und Multimedia
Gebäude 1, Raum 212
Tel.: 06721-409 152
E-Mail: mengel@fh-bingen.de

Organisation

- Programmieren 2:
  - Skript / Folien / Übungen:
    - C - Skript: Wiedling / Mengel
    - Folien im Web
    - Übungsblätter im Web
  - Literatur C: Referenzhandbuch, z.B.:
    - Kerninghan/Ritchie: Programmieren in Ansi C, Hanser Verlag
  - Literatur C++ (Gemäß ISO/ANSI) z.B.:
    - H. Schildt: C++ Ent-Packt, MITP-Verlag, ISBN 3-8266-0731-7
    - U. Breymann: C++, Einführung und professionelle Programmierung. Hanser, ISBN 3-446-22330-4

Organisation

- Programmieren 2:
  - Vorlesung:
    - z.T. mit Laptop
    - => Programme im Web
  - Fragen:
    - Gebäude 1, Raum 212
  - Sprechzeiten
    - Nach Vereinbarung:
      Tel.: 06721-409 152
      E-Mail: mengel@fh-bingen.de

Organisation

- Programmieren 2:
  - **Klausur:**
    - 02.07.03 (siehe Internet)
    - Skript, Folien, Mitschrift, 3 Bücher, **keine Altmeister**
  - Praktikum Organisation:
    - 6 Gruppen, jeweils alle 2 Wochen
    - Termine: 2x Di, 14:00, 2x Di 15:45, 2x Do, 14:00 Uhr
    - Eintragen in Listen!
  - **Praktikum Anerkennung:**
    - mindestens drei von sechs zu lösenden Aufgaben

Wiederholung: #define

- Definition von symbolischen Konstanten

  <define> ::= # define <Name> [<Ersatz-Text>]
- Beispiel

```
#define PI 3.1415926
#define ZU_LANGER_KONSTANTENNAME_FUER_ EINE_ZEILE \
 25
#define MAXIMUM 1000
#define M (PI/8.)
```


Ergänzung: #define für Macros

- Definition von Macros

  <define> ::= # define <Name>(<Parameterliste>) <Ersatz-Text>

  wobei die Parameterliste eine oder mehrere Parameter umfassen kann, die dann innerhalb des Ersatztextes benutzt werden können
- Beispiel:

```
#define SQR(x) ((x)*(x))
#define min(x,y) \
 (((x) < (y))? (x) : (y))
```

- Falsch:

```
#define SQR1(x) ( x * x )
#define SQR2(x) (x) * (x)
```


ALDA Exkurs: Sortierprobleme

- Eines der häufig zu lösenden Probleme ist das Sortieren von Daten:
  - Sortieren von Zahlen
    - Beurteilung von Meßwerte
    - Postleitzahlen
  - Alphabetisches Sortieren
    - Adressverwaltung
    - Telefonbuch

Allgemeines Problem

- Problem:
  - Eine Feld Z mit N Datenwerten vom Type DT soll so sortiert werden, daß die Datenwerte in Z gemäß einer für den Datentyp DT gültigen Relation (z.B. < oder >) auf- bzw. absteigend angeordnet sind
- Spezifikation
  - Vorbedingung

    $\{Z[1], Z[2], ...Z[\mathrm{N}] \in DT \cap \otimes \text{ist Relation für } DT\}$
  - Nachbedingung

    $\{Z[1], Z[2], ...Z[\mathrm{N}] \in DT \cap \otimes \text{ist Relation für } DT \cap Z[1] \otimes Z[2] \cap Z[2] \otimes Z[3] \cap ... \cap Z[\mathrm{N}-1] \otimes Z[\mathrm{N}]\}$

Sortieren mittels Vertauschen

- Idee:
  - Vertausche solange benachbarte Werte gemäß der gegebenen Relation bis keine Vertauschung mehr notwendig ist
- Algorithmen: z.B. Bubblesort

```
void BubbleSort(int anzahl, int *z)
{
for(;anzahl>0; --anzahl)
  for (int i=0; i<anzahl;++i)
       if (z[i] > z[i+1]) swap (&z[i],&z[i+1]);
}
```


Teile und Herrsche / Divide and Conquer

- Grundsätzliches Vorgehen:
  - Wenn die Problem nicht trivial Lösbar ist (trivial ist z.B.: Sortiere ein Feld der Größe 1), dann
    - Zerteile das zu lösende Problem in zwei (möglichst gleich große) Teilprobleme
    - Löse die Teilprobleme wieder nach dem selben Lösungsansatz
    - Erstelle auf Basis der Teillösungen die Gesamtlösung
- Algorithmen:
  - Quicksort
  - Mergesort

Quicksort

- Prozedur (rekursiv)

```
void QuickSort(int anzahl, int z[])
{
   int wert =z[Anzahl/2], li=0, re=Anzahl-1;
   if (Anzahl > 1)
   {
     do {
          while (z[li] < wert && li<re) ++li;
          while (z[re] > wert && li<re) --re;
          if (li<re) {swap(&z[li],&z[re]); ++li; --re;}
     } while (li<re);
     if (li > 1 && li < Anzahl) qsort(li,z);
     if (Anzahl-li > 1) qsort(Anzahl-li,&z[li]);
   }
}
```


Mergesort (Algorithmus)

- Besteht Feld aus eine Element => fertig
- Besteht Feld aus mehr als einem Element, dann:
  - Teile Feld in zwei gleich große Teilfelder und rufe Sortieralgorithmus für beide Teilfelder rekursiv auf
  - Erstelle aus den sortierten Teilfeldern Ergebnisfeld, indem:
    - Solange in beiden Teilfeldern noch Elemente:
      - Vergleiche beiden ersten Elemente gemäß Relation
      - Übernimm größeres bzw. kleineres Element in Ergebnisfeld und lösche es im Teilfeld
    - Übernimm alle Elemente des nicht leeren Teilfeldes in das Ergebnisfeld.