



C-Programmierung (Teil 2)

Themenüberblick

1. Nützliche Funktionen/Befehle

- Zahlen erhöhen/verringern
- fflush()
- fgets()
- atoi()

2. Spezielle Formen von Datentypen

- Globale Variablen
- 2-dimensionale Arrays

3. Aussagen und if-Abfragen

- Aussagen
- if-Abfragen
- else

4. Schleifen

- while-Schleife
- for-Schleife

1. Nützliche Funktionen/Befehle

Zahlen erhöhen/verringern

```
int a = 0;
```

- `a++;`

- ♦ Erhöht a um 1 → `a = 1`

- `a--;`

- ♦ Verringert a um 1 → `a = -1`

- `a += n;`

- ♦ Erhöht a um n → `a = a + n`

- `a -= n;`

- ♦ Verringert a um n → `a = a - n`

fflush()

- Aktualisiert die Standardeingabe / Standardausgabe
- `fflush(stdin);`
 - ♦ Sinnvoll : vor `scanf()`
- `fflush(stdout);`
 - ♦ Sinnvoll : vor `printf()`

fgets()

- Funktion zum Einlesen von Strings
- `scanf ("%s", String);`
liest String nur bis zum nächsten Leerzeichen ein
- `char *fgets(char *String, int Länge, FILE *Eingabe)`

Liest String einer maximalen Länge aus einer Eingabe ein und speichert ihn in `String`.

fgets() – Beispiel

```
1  #include <stdio.h>
2
3  int main() {
4      char String[20];
5      fgets(String, 20, stdin);
6      printf("%s", String);
7      return 0;
8  }
```

- Liest einen String von der Standardeingabe ein und gibt ihn wieder aus
- fgets() fügt am Ende des Strings `\n` und `\0` ein

atoi()

- Formt String in **int** um (klappt nur wenn String nur aus Zahlen besteht)
- Benötigt eine weitere Bibliothek
 - ♦ `#include <stdlib.h>`
- **int** atoi(**char** *String)

Gibt String als **int**-Wert zurück

atoi() – Beispiel

```
1  #include <stdio.h>
2  #include <stdlib.h>
3
4  int main(){
5      char String[2] = "42";
6      int a = atoi(String);
7      printf("%i",a);
8      return 0;
9  }
```

- Ausgabe : 42
- Grund : Inzwischen ist die 42 ein Integer-Wert → %i

2. Spezielle Formen von Datentypen

Globale Variablen

- Variablen, die in keiner Funktion vorhanden sind
- Jede Funktion kann darauf zugreifen
 - Name darf in keiner Funktion für neue Variable verwendet werden
- Vorteil : Keine Parameterübergabe nötig
- Geeignet für feste Größen (z.B. Länge, Breite, Höhe)

Globale Variablen – Beispiel

```
1  #include <stdio.h>
2
3  double Porto = 4.00;
4
5  double berechne_Kosten(double Preis){
6      double Kosten = Preis + Porto;
7      return Kosten;
8  }
9
10 int main(){
11     double Preis;
12     printf("Wie hoch ist der Preis?\n");
13     fflush(stdout);
14     scanf("%lf",&Preis);
15     printf("\nDas Porto beträgt %.2lf Euro\n",Porto);
16     Preis = berechne_Kosten(Preis);
17     printf("Die Gesamtkosten sind %.2lf Euro\n",Preis);
18     return 0;
19 }
```

2-dimensionale Arrays

- Sehr ähnlich wie 1-dimensionale Arrays

- Initialisierung :

```
int 2DArray[2][3] = { {2, 4, 1},  
                      {6, 0, 9} } ;
```

- Zugriff :

- ♦ `2DArray[0][1]` Greift auf 1-te Zeile und 2-te Spalte zu → Wert 4
- ♦ `2DArray[1][2]` Greift auf 2-te Zeile und 3-te Spalte zu → Wert 9

3. Aussagen / if-Abfragen

Aussagen

- Wahrheitsgehalt von Aussagen ist interessant
- Wichtige Aussagen für zwei Terme a und b
 - ♦ $a == b$ a ist gleich b
 - ♦ $a != b$ a ist ungleich b
 - ♦ $a < b$ $a > b$ a ist kleiner bzw. größer als b
 - ♦ $a \leq b$ $a \geq b$ a ist kleiner bzw. größer gleich b
 - ♦ $a \&\& b$ a und b
 - ♦ $a || b$ a oder b

Wahr oder falsch

- Aussage ist wahr $\Leftrightarrow \text{Aussage} \neq 0$
- Aussage ist falsch $\Leftrightarrow \text{Aussage} = 0$
- Insb. : Wird Variable a als Aussage behandelt, gilt :
 - ♦ $a = \text{wahr}$ wenn $a \neq 0$
 - ♦ $a = \text{falsch}$ wenn $a = 0$

if-Abfragen

- **if** (Aussage) MacheEtwas;

- **if** (Aussage) {
 MacheDies;
 MacheDas;
}

- Bedeutet :

Wenn Aussage wahr ist, dann mache etwas.

if / else

- **if** (Aussage) MacheEtwas;
else MacheEtwasAnderes;

- **if** (Aussage) {
 MacheDies;
 MacheDas;
} **else** {
 MacheEtwasAnderes;
}

Bedeutung:

Wenn Aussage wahr ist,
dann mache etwas

Sonst mache etwas
anderes

if/else – Beispiel

```
1  #include <stdio.h>
2
3  int main(){
4      int a;
5      printf("Bitte eine ganze Zahl eingeben\n");
6      fflush(stdout);
7      scanf("%i",&a);
8      if(a>0) printf("Die Zahl ist positiv\n");
9      |     else if(a==0) printf("Die Zahl ist gleich 0\n");
10     |     else printf("Die Zahl ist negativ\n");
11     return 0;
12 }
```

4. Schleifen

while-Schleife

- **while** (Aussage) Aktion1;
- **while** (Aussage) {
 Aktion1;
 Aktion2;
}
- Bedeutet :

Solange die Aussage Wahr ist, führe Aktionen aus.
Prüfe nach jedem Durchlauf, ob Aussage wahr ist.

while-Schleife – Beispiel

```
1  #include <stdio.h>
2
3  int main(){
4      srand ((unsigned) time(NULL)); // Erzeugt Zufallszahl
5      int a = rand() % 1001;         // zwischen 0 und 1000
6      int b;
7      int geheim=1;
8      printf("Raten Sie eine Zahl zwischen 0 und 1000\n");
9      while(geheim){
10         fflush(stdin);
11         fflush(stdout);
12         scanf("%i",&b);
13         if(b>a) printf("Die Zahl ist zu gross\n");
14         if(b<a) printf("Die Zahl ist zu klein\n");
15         if(b==a) geheim = 0;
16     }
17     printf("\n* * * * *");
18     printf("\n* Sie haben die Zahl erraten! *");
19     printf("\n*          %4i          *",a);
20     printf("\n* * * * *");
21     return 0;
22 }
```

for-Schleife

- **for** (a;b;c) Aktion1;
- **for** (a;b;c) {
 Aktion1;
 Aktion2;
}
- a = Vorbedingung (z.B. Variable auf Wert setzen)
- b = Invariante (gilt diese Invariante, wiederhole)
- c = Aktion nach jedem Durchlauf der Schleife

for-Schleife – Beispiel

```
1  #include <stdio.h>
2
3  int main() {
4      int i;
5      int array[50];
6      for(i=0;i<50;i++) array[i]=i;
7      printf("array[42]=%i\n",array[42]);
8      return 0;
9  }
```

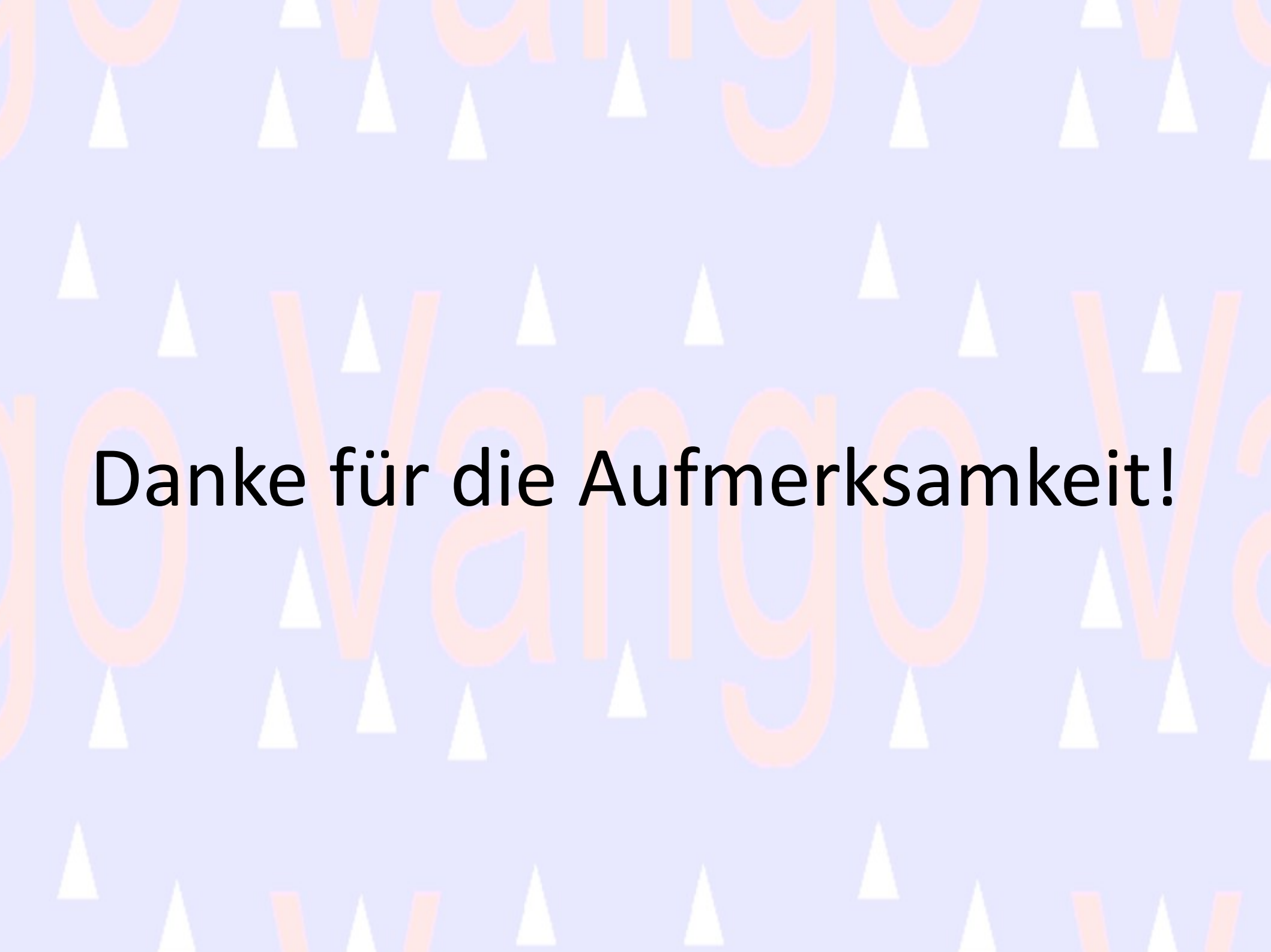
- Füllt das Array mit den Zahlen 0 bis 49
- Bei $i=50$ ist die Invariante nicht mehr wahr, also bricht die **for**-Schleife ab und das Programm läuft weiter

break

- **break;**
- Die Schleife bricht sofort ab – unabhängig von den Bedingungen

- Beispiel:

```
1  #include <stdio.h>
2
3  int main() {
4      int i;
5      for(i=1;i<100;i++){
6          if(i==42) break;
7      }
8      printf("%i",i);
9  }
```



Danke für die Aufmerksamkeit!