C-Einführung

1 Programmaufbau

2 Headerfiles

3 Funktionen

4 Arrays

1 Programmaufbau

```
1  // Programmbeginn -> Headder
2  #include <stdio.h>
3
4  //globale Konstanten definieren
5  int iPi=3.141592653;
6  //Funktionen deklarieren
7  int  fctAddieren(int, int);
8
9  //Hauptprogramm
10 int main()
11 {
12    //allgemeine Variablen initialisieren
13    int iErgebnis;
14    int iA, iB;
15    //Programmblöcke, Loops, Input,...
16    scanf("%d%d", &iA, &iB);
17    iErgebnis=fctAddieren(iA, iB);
18    fctAusgabe(iErgebnis);
19    //Programmende
20    return 0;
21 }
22 //Definition der Funktionen
23 int fctAddieren(int iZahl1, int iZahl2)
24 {
25    int iSumme;
26    iSumme=iZahl1*iZahl2;
27    return iSumme;
28 }
```

- Headerfiles
- globale Konstanten & Funktionen
- int main()
- {return 0;}
- Funktionen

2 Headerfiles

- Code Dateien
- Bereitstellen von Funktionen
- übersichtlich
- Arbeitsaufteilung

Beispiele:

- stdio.h
  - printf
  - scanf
- math.h
  - sin(x)
  - exp(x)

2 Headerfiles

2 Arten:

- von C-Programmbibliothek vorgegeben
  - #include <file>
- selbst geschrieben
  - #include “file”

```
 1  // Headertitel
 2  #ifndef MYHEADER_H
 3  #define MYHEADER_H
 4
 5  //Funktionen decalarieren
 6  int fctRechnen(int,int);
 7
 8  //Funktionen definieren
 9  int fctRechnen(int iX,int iY)
10  {
11    //Code
12  }
13
14  //Beenden des Headers
15  #endif
```

3 Funktionen

- Kombination von Befehlen
- Nutzen:
  - Mehrfachverwendung
  - Programmgliederung
- main() → Hauptfunktion
  - Aufrufen von Unterfunktionen
  - aus Headerdatei
  - im Code definierte Funktionen

3 Funktionen

- Deklarierung:
  - Rückgabetypen Name (Eingabetypen);
  - Wichtig für Compiler

```
1  // Programmbeginn -> Headder
2  #include <stdio.h>
3
4  //Funktionen deklarieren
5  int  fctAddieren(int, int);
6  void fctAusgabe(int);
```

3 Funktionen

- Definition:
  - Rückgabetypen Name (Eingabetypen mit Parameternamen) {Body}

```
 1  //Definition der Funktionen
 2  int fctAddieren(int iZahl1, int iZahl2)
 3  {
 4     int iSumme;
 5     iSumme=iZahl1*iZahl2;
 6     return iSumme;
 7  }
 8  void fctAusgabe(int iAusgabe)
 9  {
10     printf("%d",iAusgabe);
11     return;
12  }
```

3 Funktionen

- Rückgabetypen
  - void → keine Rückgabe
  - char → Zeichenrückgabe
  - int → Werterückgabe
- Aussgabe → return
- Inhalte nur lokal
  - Variablen außerhalb nicht veränderbar
  - Ausnahme → Pointer

3 Funktionen - Programmbeispiel

```c
// Programmbeginn -> Headder
#include <stdio.h>


//globale Konstanten definieren
int iPi=3.141592653;
//Funktionen deklarieren
int  fctAddieren(int, int);
void fctAusgabe(int);

//Hauptprogramm
int main()
{
   //allgemeine Variablen initialisieren
   int iErgebnis;
   int iA, iB;

   //Hauptprogramm schreiben
   scanf("%d%d", &iA, &iB);
   iErgebnis=fctAddieren(iA, iB);
   fctAusgabe(iErgebnis);

   //Programmende
   return 0;
}

//Definition der Funktionen
int fctAddieren(int iZahl1, int iZahl2)
{
   int iSumme;
   iSumme=iZahl1*iZahl2;
   return iSumme;
}
void fctAusgabe(int iAusgabe)
{
   printf("%d",iAusgabe);
   return;
}
```

3 Funktionen - Programmbeispiel

```c
//Deklaration der Funktion
void fctVertauschen(int *iX, int *iY);

int main ()
{
   //lokale Varriablen
   int iA = 100;
   int iB = 200;

   printf("vor dem Tausch iA : %d\n", iA );
   printf("vor dem Tausch iB : %d\n", iB );

   //Funktionsaufruf, &iA -> Adresse von iA
   fctVertauschen(&iA, &iB);

   printf("nach dem Tausch iA : %d\n", iA );
   printf("nach dem Tausch iB : %d\n", iB );

   return 0;
}

// Definition der Funktion
void fctVertauschen(int *iX, int *iY)
{
   int iTemp;
   iTemp = *iX;     //Wert an Adresse iX (iA)
   *iX = *iY;       //iY in iX
   *iY = iTemp;      //iTemp in iY

   return;
}
```

- mit Pointer

- Ausgabe:

```
vor dem Tausch iA : 100
vor dem Tausch iB : 200
nach dem Tausch iA : 200
nach dem Tausch iB : 100
```

4 Arrays

- Erstellen von Elementen des selben Datentyps
- ein- und mehrdimensional
- Deklaration:
  - 1-dim: Datentyp Name [Größe]
  - 2-dim: Datentyp Name [Zeile][Spalte]
- Zugriff:
  - 1-dim: Name [Position im Array]
  - 2-dim: Name [Zeile] [Spalte]

4 Arrays - Programmbeispiel

```c
#include <stdio.h>

int main ()
{
   int iArray[ 10 ];
   int i,j;

   //Werte in das Array schreiben
   for ( i = 0; i < 10; i++ )
   {
      iArray[ i ] = i + 100; // Element i = i+100
   }

   //Ausgabe des Arrays
   for (j = 0; j < 10; j++ )
   {
      printf("Element[%d] = %d\n", j, iArray[j] );
   }

   return 0;
}
```

- Ausgabe:

```
Element[0] = 100
Element[1] = 101
Element[2] = 102
Element[3] = 103
Element[4] = 104
Element[5] = 105
Element[6] = 106
Element[7] = 107
Element[8] = 108
Element[9] = 109
```

4 Arrays

- Parameter von Funktionen → Pointer
- Rückgabe einer Funktion → static
- Dynamische Größe → Pointer

Quellen

- www.Tutorialspoint.com/cprogramming
- www.Wissrech.ins.uni-bonn.de
- www.Codepad.org