Inhaltsverzeichnis

1 Die einzelnen Bytes einer Int-Variablen

Erstellen Sie ein Programm, welches die einzelnen Bytes einer Int-Variablen (positive Ganzzahl) ausliest.

```
Dieses Programm liest mittels eines Zeigers die einzelnen Bytes einer Integer-Variablen aus.
Geben Sie eine positive ganze Zahl ein >9876543
Die Zahl 9876543 in hexadizimal Format: 96B43F
Die Adresse der 0te Byte: 008FF7E0, der Wert = 3F
Die Adresse der 1te Byte: 008FF7E1, der Wert = B4
Die Adresse der 2te Byte: 008FF7E2, der Wert = 96
Die Adresse der 3te Byte: 008FF7E3, der Wert = 00
```

Tipps:
- Definieren Sie einen Zeiger vom Typ unsigned char *pch und initialisieren Sie pch mit der Adresse der Int-Variablen.
- Die Anzahl Bytes kann mittels sizeof-Operator ermittelt werden: sizeof(int)
- Verwenden Sie die Formatierung %02X um die einzelnen Bytes als zweistellige Hexadezimalzahlen darzustellen.

2 Zeigeranalyse

Ziel:
Der Lernende kann einfache Zeigermanipulationen korrekt analysieren.

Problemstellung:

Folgender kleine Programmausschnitt sei gegeben:

```
int zahlA = 3, zahlB = 5;
int *ptrA = &zahlA, *ptrB = &zahlB, *ptrC = ptrA;
int** pptrA = &ptrA;

printf("zahlA = %u, zahlB = %u, *ptrA = %u, *ptrB = %u, *ptrC = %u \n",
    zahlA, zahlB, *ptrA, *ptrB, *ptrC);
printf("Ausgabe 1 : %u \n", ptrA == &zahlA);
printf("Ausgabe 2 : %u \n", **pptrA);
printf("Ausgabe 3 : %u \n", 7 * *ptrA / *ptrB + 7);

printf("Ausgabe 4 : %u \n", *((char*)ptrB + 1));
printf("Ausgabe 5 : %u \n", *(ptrC = &zahlB) *= *ptrA + 1);
```

```
printf("zahlA = %u, zahlB = %u, *ptrA = %u, *ptrB = %u,
*ptrC = %u \n", zahlA, zahlB, *ptrA, *ptrB, *ptrC);
```

Aufgabe:
Analysieren Sie das Programm und geben Sie an, was auf dem Bildschirm erscheint!

```
zahlA = , zahlB = , *ptrA = , *ptrB = , *ptrC =
Ausgabe 1 :
Ausgabe 2 :
Ausgabe 3 :
Ausgabe 4 :
Ausgabe 5 :
zahlA = , zahlB = , *ptrA = , *ptrB = , *ptrC =
```

3 Memory-Byte (Falkutativ)

Ziel:
Der Lernende kann via Zeiger beliebige Speicherstellen adressieren und auslesen
Der Lernende kann eine Zeigerarithmetik ausführen
Der Lernende kann einen Typecast auf Zeigertypen durchführen

Problemstellung:
Mit Zeigern kann praktisch beliebig auf Speicherbereiche zugegriffen werden. Mit dieser kleinen Uebung soll ein
erster „Gehversuch“ mit Zeigern gemacht werden.

Aufgabe:
Es ist eine Funktion *GetMemByte* zu erstellen, welcher eine Adresse und ein Offset1 übergeben werden kann und welche dann das Byte an dieser Adresse+Offset zurück gibt.
z.B.

```
double EineZahl = 1000;
byte EinByte;
:
EinByte = GetMemByte( &EineZahl, 3 );
:
```

Damit würde das 4.Byte (Offset = 3) der gebrochenen Zahl *EineZahl* zurück gegeben.
Damit die Funktion universell eingesetzt werden kann, soll die Funktion mit einem generischen-Zeiger als Parameter arbeiten.
Mit dieser Funktion soll dann der Inhalt einer Integer- und einer Float-Variable byteweise ausgegeben werden.
z.B.

```
Testprogramm für die Funktion 'GetMemByte'
Dump der Int-Variable: 12,34,56,78,
Dump der Float-Variable: c2,c8,00,00,
```

Hinweise:
Mit der C-Direktive *sizeof( )* kann die Speichergrösse – in Byte's – einer Variable oder eines Datentypes ermittelt werden.
Mit *typedef unsigned char byte;* wird ein eigener Datentype 'byte' definiert. D.h. wenn im Programm der Datentyp *byte* verwendet wird, setzt der Compiler dann dafür *unsigned char* ein.

Mit *void *Ptr* wird ein sogenannter generischer Zeiger definiert. Diese Variable kann eine Adresse auf irgend ein Datentyp aufnehmen. Soll aber auf den Inhalt zugegriffen werden, muss zuerst eine Type-cast durchgeführt werden.

Historie

Dokument erstellt	X.Cheng	01.09.2021