Vorlesung3

Plan der Vorlesung

- Variable und Speicheradresse.
- Adresse und Adresseoprator.
- Speicheradresse
- Zeigervariablen und einfache Variablen
- Vereinbarung von Zeigern

Die Speicheradresse

- Jede Speicheradresse entspricht einem Byte (8 Bits) des Speicherplatzes
- Jede Variable besitzt eine eigene Speicheradresse.
- Die Speicheradresse ist eindeutig für jede Variable.

Variable ⟷ Speicheradresse

Speicheradresse	FF02
Wert	-1023

Arrays und Speicheradresse

```
int a[7] = {234, -872, 0, 2, 1009, 2191, -7427}
```

WERTE	234	-872	0	2	1009	2191	-7427
ADRESSE	FF02	FF06	FF0A	FF0E	FF12	FF16	FF1A
POSITION	0	1	2	3	4	5	6

Ermittlung der Adresse

Wie kann man die Adresse einer Variable ermitteln?

Durch den **Operator &**

```
&Variablename
```

Vorlesung2_5.cpp

Variablen und Adresse

Aufgabe:
Es soll ein Programm erstellt werden.
Das Programm sollte Variablen von verschiedenen Typen einlesen und deren Werte sowie deren Adressen auf dem Bildschirm ausgeben.

Vorlesung3_1.cpp

Zeiger und Adresse

Aufgabe:
Es sollte ein Programm entwickelt werden ,das Programm sollte Variablen von unterschiedlichen Datentypen einlesen und deren Adressen in Variablen speichern und letztendlich deren Inhalt auf dem Bildschirm ausgeben.

Lösungsweg

- Einlesen von Variablen.
- Adressen von Variablen Speichern:Wie?und Wo?
- Inhalt Variablen Ausgeben.

Vorlesung3_2.cpp

Was sind Zeiger ?

- -Nichts, wovor man sich fürchten muss! **Zeiger** (engl. **Pointer**) sind ganz gewöhnliche Variable.
- -Im Gegensatz zu Variablen vom Typ char, int,long, float, double usw. wollen sie aber keine Werte von diesen Datentypen aufnehmen, sondern **Adressen** !

-Zeiger speichern nur Adresse.

Vereinbarung von Zeiger

```
int  * p_1;     // Zeiger von int
double  *p_2; //Zeiger von double
char *p_3;      //Zeiger von char
short *p_4;     //zeiger von short
```

Vereinbarung von Zeiger

- Allgemein wird also eine Zeigervariable (kurz: ein Zeiger) so definiert:

```
Typ   *p;
```

Man sagt: "p ist ein Zeiger auf ein Objekt vom Datentyp Typ."

- Der Typ eines Zeigers ist: Typ * (oder ohne Leerzeichen: Typ*)

Vereinbarung von Zeiger

- Der Typ eines Zeigers ist: Typ * (oder ohne Leerzeichen: Typ*)

-p_1 hat den Typ int*

-p_2 hat den Typ double*

-p_3 hat den Typ char*

Zuweisung von absoluten Adressen

Im Rahmen der Mikrocontoller-Programmierung wird mit Hilfe von Zeigern auf bekannte, d.h. absolute RAM Adressen zugegriffen. Bei Systemen mit "Memory Mapped I/O" greift man darüber hinaus auf Port- bzw. Registeradressen von Hardware-Bausteinen genauso über Zeiger zu wie auf RAM-Adressen.

Solchen I/O–Adressen sind im Gegensatz zu Variablenadressen keine Namen zugeordnet.Man kennt diese Adressen ganz einfach, weil sie entweder vorgegeben sind oder weil man sie selber einstellt. In all diesen Fällen muss man also Zeigern bekannte(= absolute) Adressen zuweisen!

Vorlesung3_3.cpp

Zuweisung von absoluten Adressen

p_1 = 0x0065FDF8;Compiler-Fehlermeldung:

"int kann nicht in int gewandelt werden!"*

Vorlesung3_3.cpp

Zuweisung von absoluten Adressen

p_1 = (int)0x0065FDF8;Erfolg*

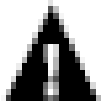
Vorlesung3_3.cpp

Zuweisung von absoluten Adressen

p_1 = (int)0x0065FDF8;*

*p_2 = (double *)0x0065FDF8;*

*p_3 = (char *)0x0065FDF8;*



Alle drei Zeiger zeigen jetzt auf dieselbe Speicheradresse 0065FDF8, aber p_1 "sieht" dort einen 4-Byte–int-Wert, p_3 einen 1-Byte–char–Wert und p_2 einen 8-Byte–double–Wert!

Vorlesung3_3.cpp

Lesen und Schreiben von Daten Über Zeiger

Problem:Schreiben Sie ein Programm,das ein Zeiger auf einer gültigen Adresse von Typ int enthält. Das heißt er zeigt auf nicht geschützten Speicherbereich.

Lesen Sie bitte die Werte des Betreffenden Typs aus dem Speicher.

Schreiben Sie bitte auch im Speicher.

Lösungsweg

-Zeiger auf int vereinbaren.

-Variable von int vereinbaren.

-Adresse der variable in Zeiger speichern.

-Auf die Variable durch den Zeiger zugreifen.

*Lesen.

*schreiben.

Vorlesung3_4.cpp

Verweisoperator *

Je nach Tagesform nennen ihn die Leute auch

- Zugriffsoperator,
- Dereferenzierungsoperator oder
- Indirektionsoperator.

Verweisoperator *

***p**

bedeutet einen lesenden oder schreibenden Zugriff auf den Speicherbereich, auf den der Zeiger p zeigt.

Achtung: 1 Symbol – 3 Gebräuche

Das Symbol * können wir finden in 3 Zusammenhänge:

- Als Multiplikationsoperator: `d = (3 + b) * 5 - c;`
- In der Deklaration des Datentyps Zeiger: `int *a;`
- Als Verweisoperator bei der Dereferenzierung des Zeigers: `*d = *c + 5;`

Unsere erste Funktion mit Zeigern

Ziel: Eine Funktion, die eine Variable von *main* verändert. Ohne Rückgabe!

Bisher: übliche Funktion

- Nur durch Rückgabe können wir eine Variable im *main* verändern.

  Typischer Aufruf

```
c = funktion (a, b);
```

Vorlesung3_5.cpp

Übergabe per Wert

- Keine Möglichkeit ohne Adressen eine in *main* deklarierte Variable zu verändern.

```
funktion (a, b, c);
```

Durch diesen Aufruf c wird gleich bleiben

Vorlesung3_7.cpp

Erklärung

- Durch einen üblichen Aufruf übergeben wir nur Werte in der Funktion
- Die Variablen haben einen Gültigkeitsbereich
  - Die Variablen in *main* können nicht anhand des Namens in einer Funktion.
  - Umgekehrt auch für die Variablen in der Funktionen
- Die Speicheradresse einer Variable ist trotzdem überall gültig!

Von nun an: Nutzung von Zeigern

- Wenn die Adresse der Variable wird in die Funktion übergeben, kann man die Variable direkt in der Funktion verändern.

  Typischer Aufruf mit Adressenübergabe

```
funktion (a, b, &c);
```

  - Von den Variablen a und b nutzt man nur den Wert. Die können nicht in der Funktion verändert werden
  - Von der Variable c übergibt man die Adresse, deshalb c ist verändbar

Vorlesung3_6.cpp