The GAME Engineers

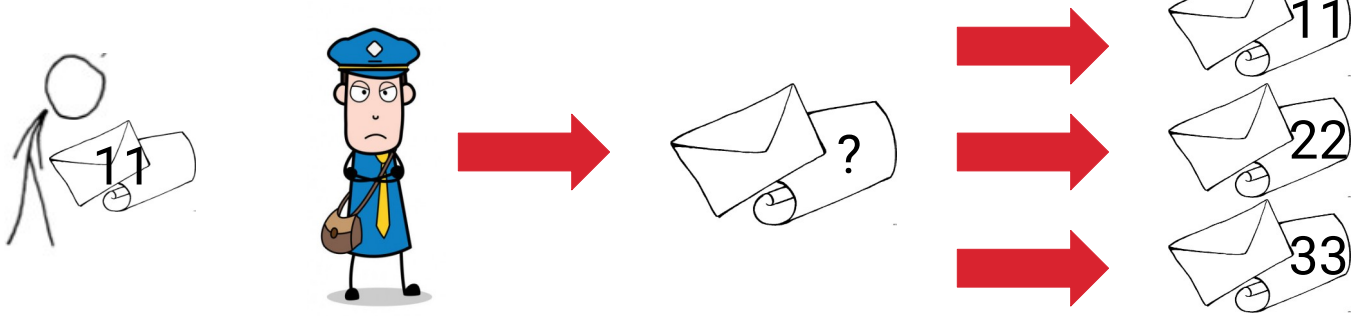
CG #4 – Teil 1: Pointer

Inhalt

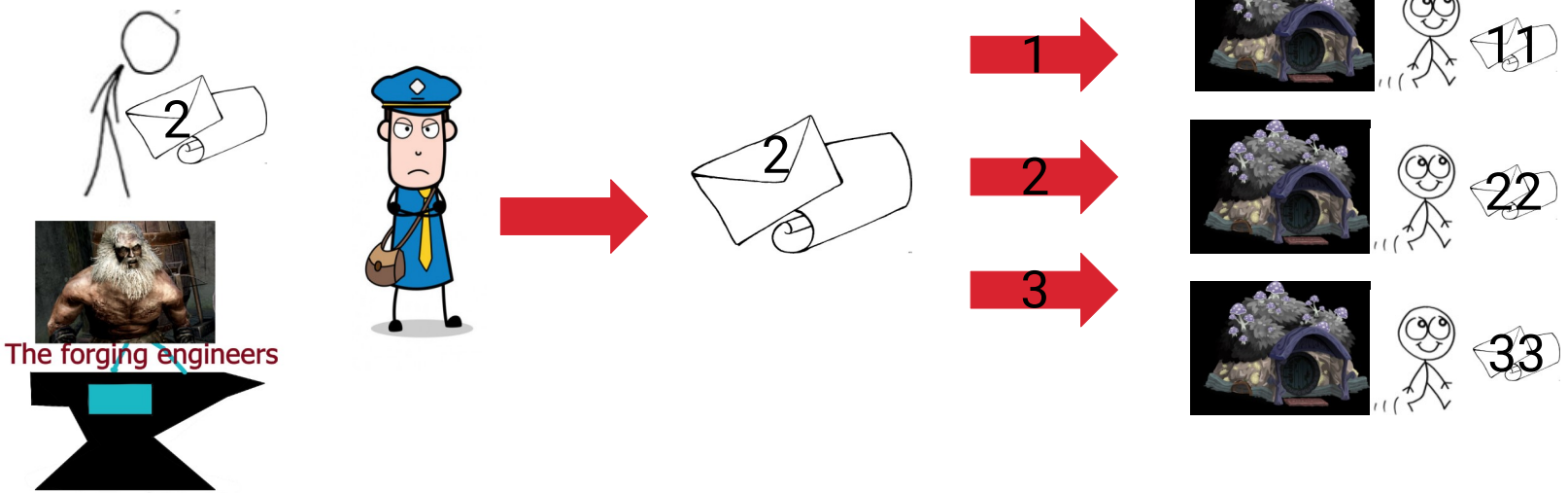
- Was sind Pointer?
  - Bildlich
  - PC-Welt

- Vor- und Nachteile

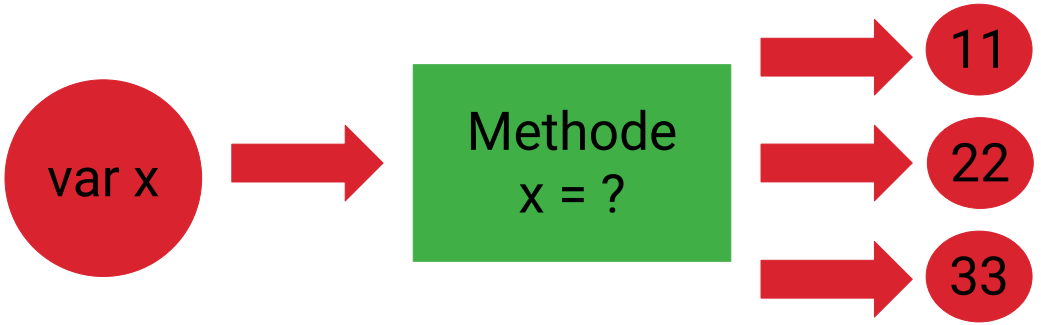
Was sind Pointer? – Bildlich

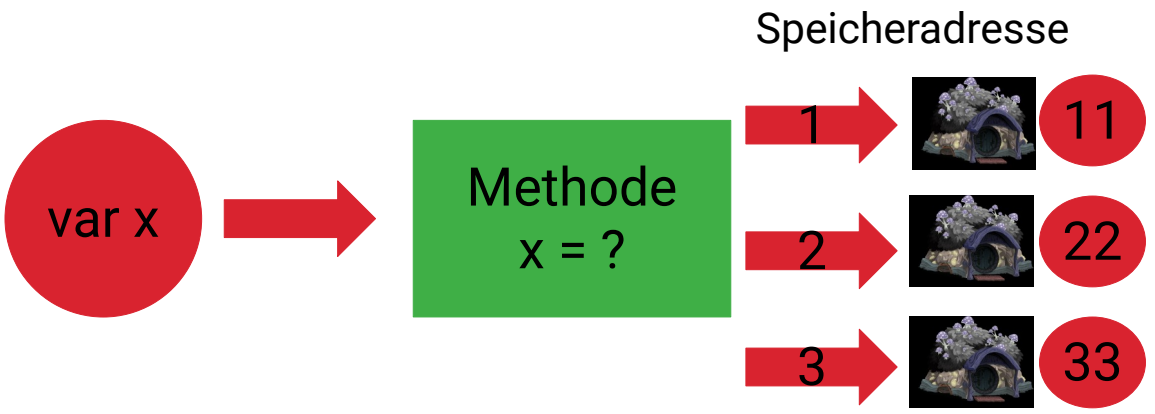


Änderung ?



Was sind Pointer? – PC Welt



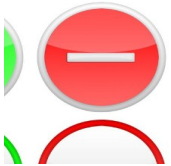


Besonderheit

null

Vor und Nachteile

- Speicherplatz sparen
- Speicherplatz freigeben
- Lebensdauer selbst festlegen
- Objekt muss nicht kopiert werden
- Mehrere Pointer auf die gleiche Variable
- Direkt in den Speicher schreiben



- Komplizierter und fehleranfälliger Umgang
- Keine Datenkontrolle
- Speicher wird überfüllt
- Daten beschädigt
- Daten können weit auseinander sein
- Typische Lücke für Malware

The GAME Engineers

CG #4 – Teil 2: Einführung in C

Inhalt

- C
  - Was und wozu?
  - C++ und C# ?
  - Compiler
  - Dateiendungen
  - Dateiaufbau

- Datentypen
  - Zeichen
  - signed und unsigned
  - Ganzzahlen
  - Gleitkommazahlen
  - Deklaration und Definition
  - Initialisierung
  - Pointer

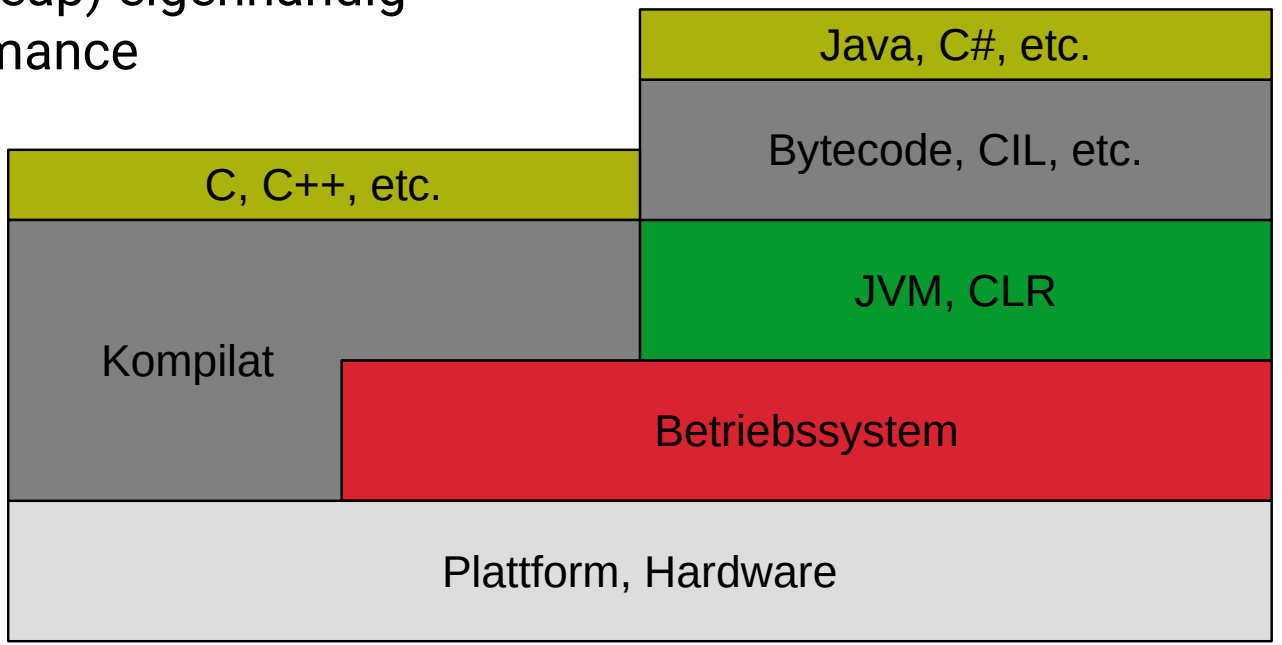
- Einfache Ausgaben
  - printf
  - Ausflug: Java

C – Was und wozu?

- C ist maschinennahe Sprache
  - Was heißt das?
  - Embedded, Treiber, Betriebssysteme, …
  - Speicherverwaltung (Heap) eigenhändig
  - Häufig bessere Performance

- Breite Unterstützung

- Linux, Windows, etc.
  u.a. in C geschrieben



C – C++ und C#?

- C++ ist ursprünglich eine Erweiterung von C gewesen
  - ursprünglich eine Erweiterung von C (um echte Objektorientierung)
  - Mittlerweile eigene Programmiersprache

- C#
  - OOP
  - ähnlich zu Java
  - Common Intermediate Language
  - Common Language Runtime

C – Compiler

- GCC
  - GNU Compiler Collection ≠ GNU C Compiler
  - Aber: GNU C Compiler in GNU Compiler Collection enthalten

- MingW
  - Windows Portierung von GCC

- Nutzung: **gcc datei1.h datei1.c datei2.h datei2.c … -o ExecDateiName**

C – Dateiendungen

- Nutzung: gcc datei1**.h** datei1**.c** datei2.h datei2.c … -o ExecDateiName
  - Zwei Dateiendungen:
    - .c – C Source Code
    - .h – Header Dateien

- Header Dateien beschreiben den Inhalt einer .c-Datei
  - Welche Funktionen sind darin
  - Welche Konstanten / Strukturen werden darin verwendet
  - Etc.
  - In späteren Präsentationen mehr

C – Dateiaufbau - .c

“import”

```c
#include <stdlib.h>
#include <stdio.h>

int main(void)
{
    printf("Hallo\n");

    int *a = malloc(sizeof(int));
    scanf(" %d", a);

    printf("%d", *a);

    return 0;
}
```

main Funktion

Befehle

Datentypen – Ganzzahlen

```c
int main(void)
{
    // 1 Byte vorzeichenlos
    // -> 0 - 255 (ASCII-Zeichen)
    char zeichen = 'a';

    return 0;
}
```

Datentypen – signed und unsigned

- Bei Java sind wir es gewohnt, dass durchgängig Zweikomplement angewendet wird

- Bei C grundsätzlich auch,
  - aber durch Schlüsselworte “signed” und “unsigned” kann man Zahlen als vorzeichenlos (unsigned) oder vorzeichenbehaftet (signed) behandeln
  - signed ist Standard und muss nicht mitangegeben werden

Datentypen – Ganzzahlen – short und int

```
// 2 Byte
// -> -32.768 – 32.767
short zahl1 = 12345;

// -> 0 – 65.535
unsigned short zahl2 = 12345;
```

```
// min. 2 Byte; meist 4 Byte
// -> -2.147.483.648 - 2.147.483.647
int zahl1 = 12345;

// -> 0 - 4.294.967.269
unsigned int zahl2 = 12345;
```

Datentypen – Ganzzahlen – long und long long

„long“ ist dabei
kurz für “long int”

```
// min. 4 Byte;
// meist 4 Byte bei Windows,
// 8 Byte bei Unixoiden
// -> -9,2e18 – 9,2e18
long zahl1 = 12345;

// -> 0 – 18,5e18
unsigned long zahl2 = 12345;
```


```
// min. 8 Byte
// -> -9,2e18 – 9,2e18
long long zahl1 = 12345;

// -> 0 – 18,5e18
unsigned long long zahl2 = 12345;
```

Datentypen – Gleitkommazahlen – float, double und long double

```
// 4 Byte
// -> -3,4e38 – 3,4e38
float zahl1 = 12345.0f;
```

```
// 8 Byte
// -> -1,7e308 – 1,7e308
double zahl1 = 12345.0d;
```

```
// min. 10 Byte
// -> -1,7e4932 – 1,7e4932
long double zahl1 = 12345.0d;
```

Datentypen – Deklaration und Definition

Java:

```java
int a;

System.out.println(a);
```

Kompiliert es?

```
variable a might not
have been initialized
```

C:

```c
int a;

printf("Ausgabe: %d", a);
```

Kompiliert es?

Welche Ausgabe
Erwartest du?

```
Ausgabe: 16
```

Datentypen – Deklaration und Definition

Zahlen:

```
int a; a = 0;
int b = 0;

float c = 3.0f;
double d = 4.0;
```

```
signed int a;

unsigned int b;
```

String:

```
char a[] = "Hallo";

char *b = "Hallo";
```

Arrays:

```
int a[3];
int b[] = {0, 1, 2};

// mehr hierzu folgt
// noch
```

Datentypen – Pointer

```
pointer a;
```

Enthält Speicheradresse als Wert

Adresse	Wert
0000	
0001	
0002	5
0003	
0004	
0005	0002
0006	
0007	
...	...

Zeigt auf Speicheradresse

Datentypen – Pointer

```c
int *a;

a = malloc(sizeof(int));
*a = 5;

printf("%p\n%d", a, *a);
```

```
Adresse: 0000000000A31460
Inhalt: 5
```

Datentypen – Pointer

```c
int *a;
int b = 101;

a = &b;

printf("Adresse: %p ; %p\n", a, &b);
printf("Inhalt: %d ; %d", *a, b);
```

```
Adresse: 000000000061FE14 ; 000000000061FE14
Inhalt: 101 ; 101
```

Einfache Ausgaben – printf

```c
#include <stdio.h>

int main(void)
{
    printf("Hallo! Ich heisse Niklas");

    return 0;
}
```

Einfache Ausgaben – printf

```
int a = 5;

printf("Was ist in a? " + a);
```

```
int a = 5;

printf("Was ist in a? %d", a);
```

Einfache Ausgaben – printf

%d, %i	int, short, long	Ganzzahlen
%x, %X	int, short, long	Ganzzahlen in Hex
%f	float, double	Gleitkommazahlen
%e, %E	float, double	Gleitkommazahlen in Zehnerpotenz
%c	char	Zeichen
%s	char *	String

Einfache Ausgaben – printf

```c
int a = 255;

printf("%d, %i, %x, %X", a, a, a, a);
```

```
255, 255, ff, FF
```

```c
double a = 255.0;

printf("%f, %e, %E", a, a, a);
```

```
255.000000, 2.550000e+002, 2.550000E+002
```

Einfache Ausgaben – printf

```c
char a = 'b';
char *b = "McMuffin";

printf("%c, %s", a, b);
```

```
b, McMuffin
```

Einfache Ausgaben – printf

```
int a = 51, b = 12345;
```

```
printf("%d\n", a);
printf("%d", b);
```

```
51
12345
```

```
printf("%5d\n", a);
printf("%5d", b);
```

```
   51
12345
```

```
printf("%05d\n", a);
printf("%05d", b);
```

```
00051
12345
```

Einfache Ausgaben – printf

```
double a = 1.0/3;
```

```
printf("%f", a);
```

```
0.333333
```

```
printf("%.2f", a);
```

```
0.33
```

```
printf("%5.3f\n", a);
printf("%6.3f", a);
```

```
0.333
 0.333
```

Einfache Ausgaben – printf

```
printf("Das Produkt kostet %.2f Euro\n"
       "und mit %d %% MwSt. kostet es %.2f Euro.",
       10.0, 19, 10.0 * 1.19);
```

```
Das Produkt kostet 10.00 Euro
und mit 19 % MwSt. kostet es 11.90 Euro.
```

Einfache Ausgaben – Ausflug: Java

```
System.out.printf("%d, %.3f, %s", 10, 2/3d, "Uff");
```

```
10, 0,667, Uff
```

```
var a = "%s ist %d Jahre alt".formatted("Niklas", 22);

System.out.println(a);
```

```
Niklas ist 22 Jahre alt
```


```
var a = String.format("%s ist %d Jahre alt", "Niklas", 22);
```

Ende

Fragen?