

PROGRAMMIEREN II

DHBW Stuttgart Campus Horb INF2017

AGENDA FÜR DIE HEUTIGE VORLESUNG

- Erstellen von Zufallszahlen
- Structs im C Sinne
- Pointer

ZUFALLSZAHLEN

- Computer beherrschen in der Regel nur Pseudozufallszahlen
- Als **Pseudozufall** wird bezeichnet, was zufällig erscheint, in Wirklichkeit jedoch berechenbar ist
- Deterministische Algorithmen erzeugen bei gleichen Seed of die gleiche Zahlenfolge
- Seed ist der Startwert für den Generator, häufig wird hier der aktuelle Timestamp genutzt

RANDOM NUMBERS

```
#include <random>
#include <iostream>

int main()
{
    //Will be used to obtain a seed for the random number engine
    std::random_device rd;
    //Standard mersenne_twister_engine seeded with rd()
    std::mt19937 gen(rd());
    std::uniform_int_distribution<> dis(1, 6);
    // role the dice 10 times
    for (int n = 0; n < 10; ++n)
        //Use dis to transform the random unsigned int generated
        //by gen into an int in [1, 6]
        std::cout << dis(gen) << ' ';
    std::cout << '\n';
}
```

ERSTE PROGRAMMIER- AUFGABE

- Die Aufgabe in CppLotto.cpp aus dem bekannten GitHub

STRUCTS IM C SINNE

- Ein struct ist eine zusammengesetzte Datentypdeklaration, die eine physikalisch gruppierte Liste von Variablen definiert

STRUCT EXAMPLE I

```
struct tag_name {  
    type member1;  
    type member2;  
    /* declare as many members as desired,  
but the entire structure size must be known to the compiler. */  
};
```

STRUCT EXAMPLE 2

```
typedef struct tag_name {  
    type member1;  
    type member2;  
} struct_alias;
```

-

STRUCT EXAMPLE 3

```
struct account {  
    int account_number;  
    char *first_name;  
    char *last_name;  
    float balance;  
};
```

```
struct account s;
```

STRUCT EXAMPLE 4

```
typedef struct account {  
    int account_number;  
    char *first_name;  
    char *last_name;  
    float balance;  
} account_t;
```

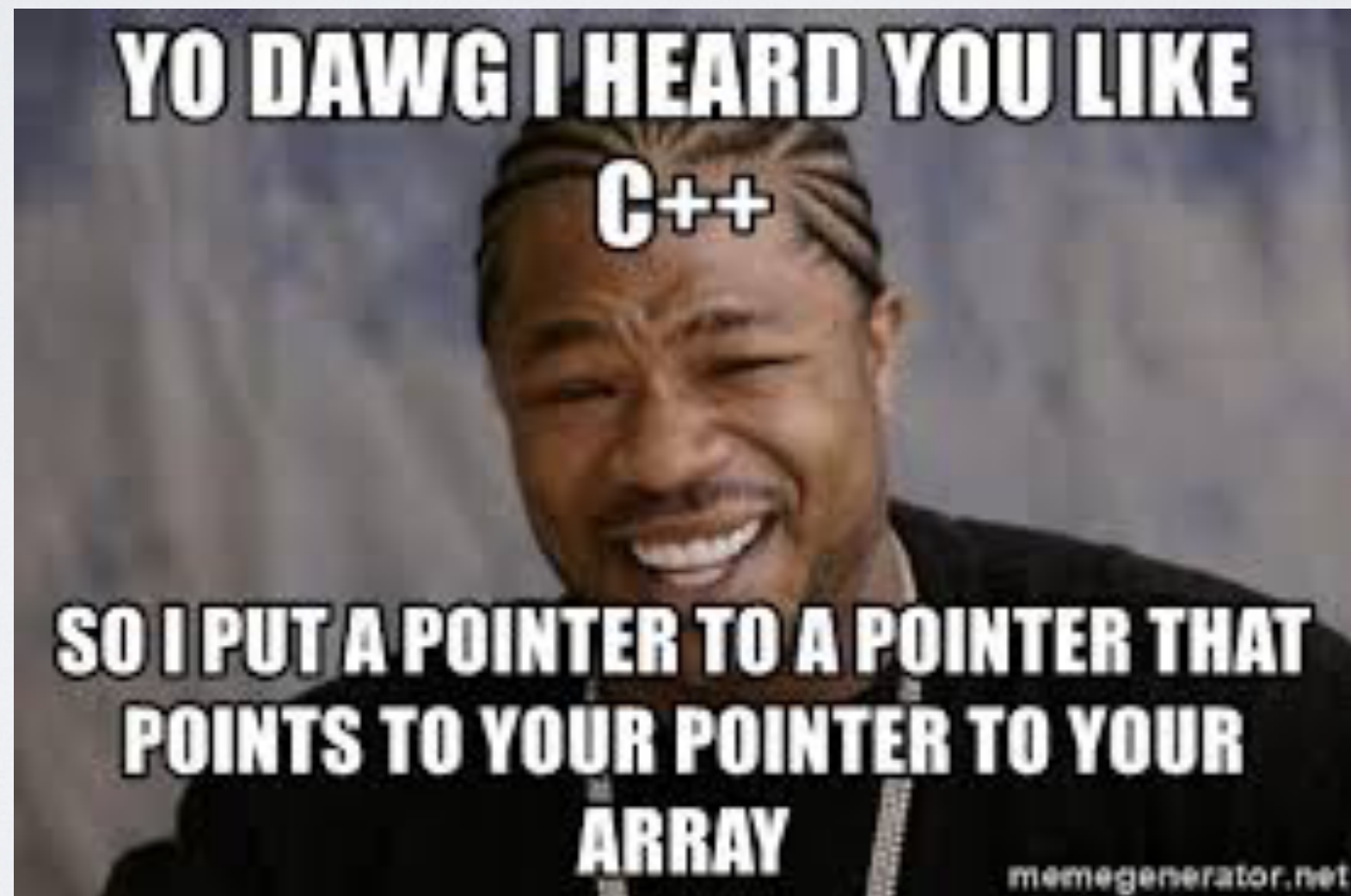
```
account_t s;
```

-

EIN PAAR FAKTEN ZU STRUCTS

- Structs sind aus anderen Datentypen zusammengefasste Datentypen
- Die Größe von Structs muss zur Compilezeit feststehen
- Structs können mit typedef als eigener Typ definiert werden

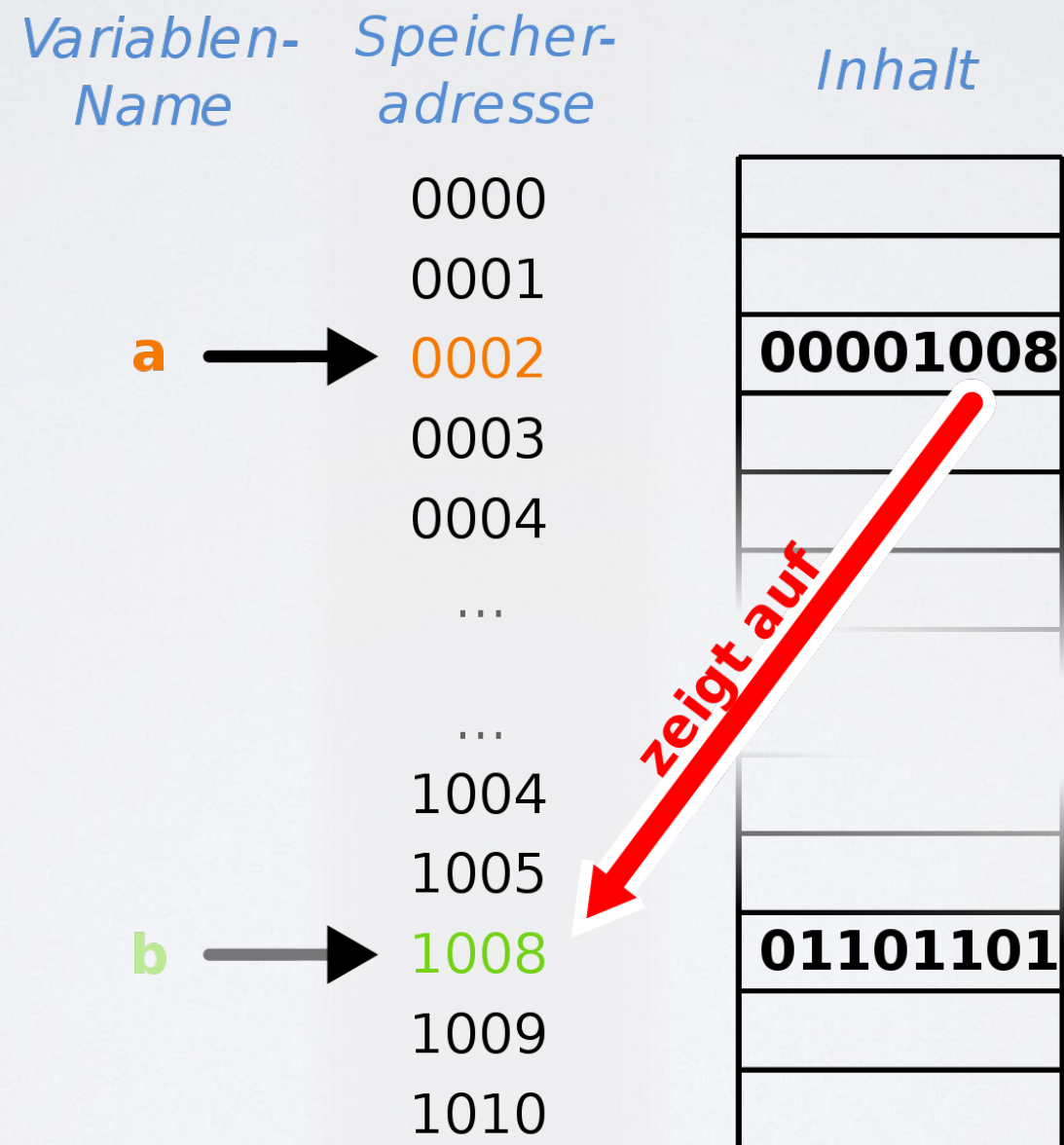
POINTER



ALLGEMEINES ZU POINTERN

- Pointer (Zeiger) zeigen auf Speicherstellen
- In C++ kann man Pointer Typisieren, dadurch weiß die Anwendung wie groß der Speicher ist
- Im Gegensatz zur Referenz müssen Zeiger explizit dereferenziert werden
- Besonderer Pointer ist der nullptr oder NULL

ZEIGER SCHAUBILD



ZEIGER BEISPIEL I

```
#include <stdio.h>
int main()
{
    long x = 9532523;
    long *x_ptr = &x;
    printf("Address: \n\t %p \n contains value:\n\t %d\n original val is:
\n\t %d\n",
        x_ptr, *x_ptr, x);
}
```

Torstens-MacBook-Pro-2:day2 torstenhopf\$./pointers

Address:

0x7ffee05deb48

contains value:

9532523

original val is:

9532523

POINTERS AND ARRAYS

```
// in c and c++ arrays and pointers are somehow familiar with  
// each other. The pointer links to the first element of  
the
```

```
// array  
char buf[50];  
char* string = buf;
```

```
// you can even use array operators on the pointer  
// this uses internally the sizeof(char) to do this  
string[0] = 'a';  
string[1] = 'b';
```

•

INDIREKTER ZUGRIFF

```
char letter = 'b';  
char *letterPointer = &letter;  
*letterPointer = '3';  
printf("Omg letter does not contain a letter anymore! but %c\n",  
      letter);
```


ZEIGERARTITHMETIK

```
char buf[50];  
char *string = buf;  
// will print all chars of the buf  
for (char *p = buf; *p; ++p)  
{  
    printf("%c", *p);  
}
```

ANONYME ZEIGER

```
void *iCanPointToEverything = 0;  
char *iAmAString = "Vingardium Liviosa";  
// no problem  
iCanPointToEverything = iAmAString;  
//iAmAString = iCanPointToEverything; would need an explicit cast
```

•

BESONDERE ZEIGER

```
char *c_ptr = NULL;  
char *dont_dereference = nullptr;  
char *i_am_also_null = 0;
```


NULLPOINTER

- Wobei NULL meist `void *ptr = 0;`
- `nullptr` in C++11 eingeführtes literal
- 0 erklärt sich selbst
- Wieso ist eine `if(ptr)` Prüfung nicht die ganze Wahrheit?

REGELN DES POINTER CLUBS

- Schrödingers Pointer
- Traue keinem Pointer den du nicht selbst invalidiert hast
- Setze Pointer wenn du sie noch nicht oder nicht mehr benutzt auf nullptr(!)

AUFGABE UND HAUSAUFGABE FÜR DAS NÄCHSTE MAL

- Erweitern der Hero App!
 - Der Hero ist ab jetzt nicht mehr nur ein String sondern ein (C) struct bestehend aus char(20) für den Namen, und einer Zufallszahl zwischen 10-30 für die Lebenspunkte, sowie einem Pointer auf einen Struct Waffe
 - Die Waffe erhält einen von - bis Schaden von rand(3) bis Ergebnis+3
 - Wenn man einen Helden aus der Liste der erstellten oder vorgefertigten Charaktere wählt, wird ein Kampf gegen einen zufälligen Gegner aus der Liste gestartet
 - Beim Kampf schlagen beide Charaktere abwechselnd mit den Waffen aufeinander ein bis einer keine Lebenspunkte mehr hat, dieser Charakter ist dann tot
 - Es beginnt der Charakter mit dem längerem Namen