

c/c++ und SDL für Anfänger

Rene Leitner

Open HW und SW Event

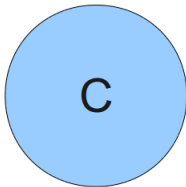
5. Dezember 2010

- ① Einleitung
- ② Entwicklungsumgebung
- ③ Grundlagen C
- ④ Codingstyle
- ⑤ Große Übung 1
- ⑥ Grundlagen C++
- ⑦ Große Übung 2
- ⑧ Grundlagen SDL
- ⑨ Große Übung 3
- ⑩ Offene Fragen

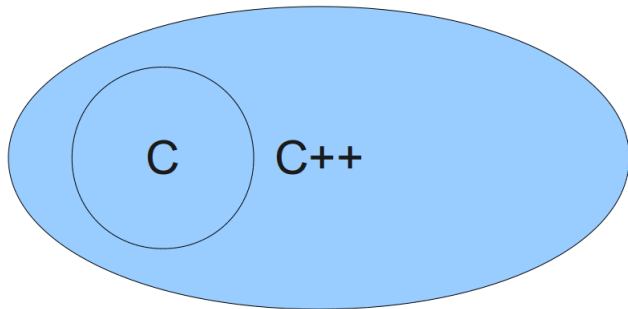
- Einleitung
 - **Vorstellungsrunde**
 - C/C++ und SDL Abgrenzung
 - Vom Code zum Ausführbaren Programm

- Einleitung
 - Vorstellungsrunde
 - **C/C++ und SDL Abgrenzung**
 - Vom Code zum Ausführbaren Programm

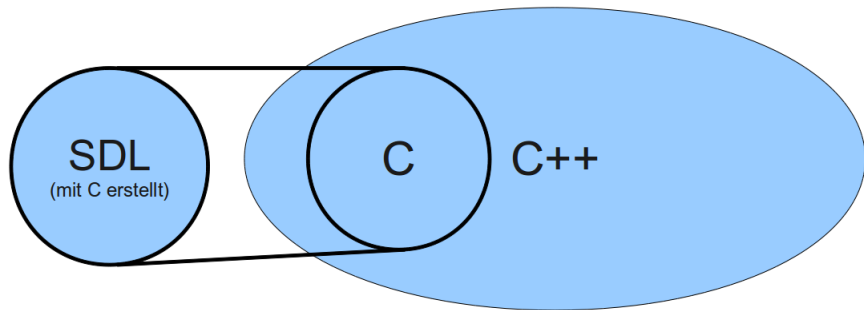
C/C++ und SDL Abgrenzung



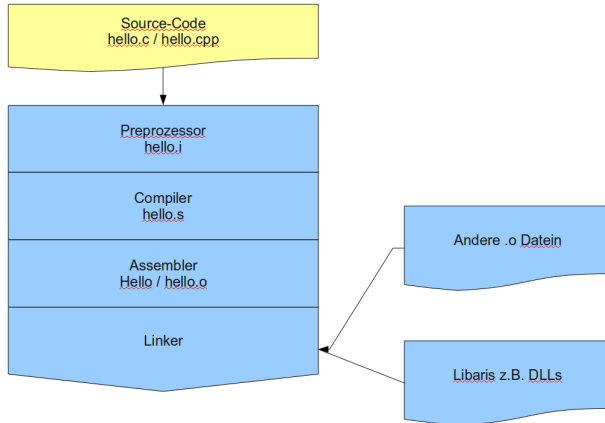
C/C++ und SDL Abgrenzung



C/C++ und SDL Abgrenzung



Vom Code zum Ausführbaren Programm



- Das ist unser Ausgangs Quellcode!

Hello World

```
#include <stdio.h>

main()
{
    // Gibt Hello World auf der Konsole aus
    printf("Hello World \n");
}
```

Preprozessor Output

- Nach dem Preprozessor wurde alle inkludierten Daten eingebunden.
- Ausgabe in der hello.i

Compiler Output

- Nach dem Compiler wurde der C Code in Assembler umgesetzt.

Hello World

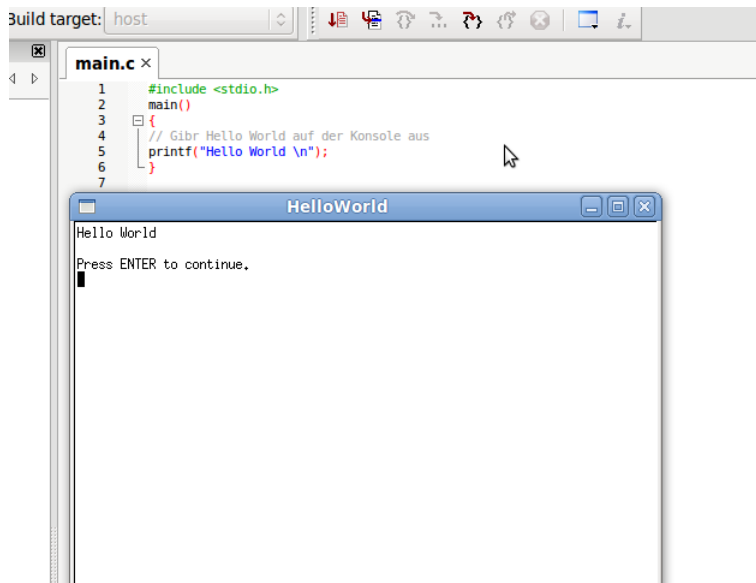
winzig

```
.file "main.c"
.section .rodata
.LCO:
.string "Hello World "
.text
.globl main
.type main, @function
main:
    pushl   %ebp
    movl    %esp, %ebp
    andl    $-16, %esp
    subl    $16, %esp
    movl    $.LCO, (%esp)
    call    puts
    leave
    ret
.size     main, .-main
.ident    "GCC: (Ubuntu 4.4.3-4ubuntu5) 4.4.3"
.section .note.GNU-stack,"",@progbits
```

Assembler Output

```
7± 45 4c 46 01 01 01 00 00 00 00 00 00 00 00 01 00 03 00 01 00 00 00 00 00 00 00 00 00
00 00 d0 00 00 00 00 00 00 00 34 00 00 00 00 00 28 00 0b 00 08 00 55 89 e5 83 e4 f0 83 ec
10 c7 04 24 00 00 00 00 e8 fc ff ff ff c9 c3 00 48 65 6c 6c 6f 20 57 6f 72 6c 64 20 00 00
47 43 43 3a 20 28 55 62 75 6e 74 75 20 34 2e 34 2e 33 2d 34 75 62 75 6e 74 75 35 29 20 34
2e 34 2e 33 00 00 2e 73 79 6d 74 61 62 00 2e 73 74 72 74 61 62 00 2e 73 68 73 74 72 74 61
62 00 2e 72 65 6c 2e 74 65 78 74 00 2e 64 61 74 61 00 2e 62 73 73 00 2e 72 6f 64 61 74 61
00 2e 63 6f 6d 6d 65 6e 74 00 2e 6e 6f 74 65 2e 47 4e 55 2d 73 74 61 63 6b 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 1f 00 00 00 01 00 00 00 06 00 00 00 00 00 00 00 34 00 00 00 17 00
00 00 00 00 00 00 00 00 00 00 04 00 00 00 00 00 00 00 1b 00 00 00 09 00 00 00 00 00 00
00 00 00 00 3c 03 00 00 10 00 00 00 09 00 00 00 01 00 00 00 04 00 00 00 08 00 00 00 25 00
00 00 01 00 00 00 03 00 00 00 00 00 00 00 4c 00 00 00 00 00 00 00 00 00 00 00 00 00 00
04 00 00 00 00 00 00 00 2b 00 00 00 08 00 00 00 03 00 00 00 00 00 00 00 4c 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 04 00 00 00 00 00 00 00 30 00 00 00 01 00 00 00 02 00 00 00
00 00 00 00 4c 00 00 00 0d 00 00 00 00 00 00 00 00 00 00 00 01 00 00 00 00 00 00 00 38 00
00 00 01 00 00 00 30 00 00 00 00 00 00 00 59 00 00 00 24 00 00 00 00 00 00 00 00 00 00
01 00 00 00 01 00 00 00 41 00 00 00 01 00 00 00 00 00 00 00 00 00 00 00 7d 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 01 00 00 00 00 00 00 00 11 00 00 00 03 00 00 00 00 00 00 00
00 00 00 00 7d 00 00 00 51 00 00 00 00 00 00 00 00 00 00 00 01 00 00 00 00 00 00 00 01 00
00 00 02 00 00 00 00 00 00 00 00 00 00 00 88 02 00 00 a0 00 00 00 0a 00 00 00 08 00 00 00
04 00 00 00 10 00 00 00 09 00 00 00 03 00 00 00 00 00 00 00 00 00 00 00 28 03 00 00 12 00
00 00 00 00 00 00 00 00 00 00 01 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 01 00 00 00 00 00 00 00 00 00 00 00 04 00 f1 ff 00 00 00 00 00 00 00 00 00 00
00 00 03 00 01 00 00 00 00 00 00 00 00 00 00 00 00 00 03 00 03 00 00 00 00 00 00 00 00 00
00 00 00 00 03 00 04 00 00 00 00 00 00 00 00 00 00 00 00 00 03 00 05 00 00 00 00 00 00 00
00 00 00 00 00 00 03 00 07 00 00 00 00 00 00 00 00 00 00 00 00 00 03 00 06 00 08 00 00 00
00 00 00 00 17 00 00 00 12 00 01 00 0d 00 00 00 00 00 00 00 00 00 00 00 10 00 00 00 00 6d
61 69 6e 2e 63 00 6d 61 69 6e 00 70 75 74 73 00 00 00 0c 00 00 00 01 05 00 00 11 00 00 00
02 09 00 00
```

Ausführbares Programm



The image shows a code editor window with a file named `main.c` open. The code is as follows:

```
1  #include <stdio.h>
2  main()
3  {
4      // Gibr Hello World auf der Konsole aus
5      printf("Hello World \n");
6  }
7
```

Below the code editor, a terminal window titled `HelloWorld` is open. It displays the output of the program:

```
Hello World
Press ENTER to continue.
```

The terminal window has a cursor on the line "Press ENTER to continue.".

- Entwicklungsumgebung
 - **Vorstellung**
 - Installation der Entwicklungsumgebung
 - Test

- leicht zu benutzen
- leicht zu installieren
- Existiert für Linux, Windows, Mac OS
- Erweiterbar durch Plugins
- Ressourcen schonend - nicht wie Eclipse
- Läuft auch auf anderen CPU Architekturen z.B. ARM OMAP
- Kostenlos
- Direkte SDL und Cross Compiler Unterstützung
- Deutsch und Englische Beschreibung

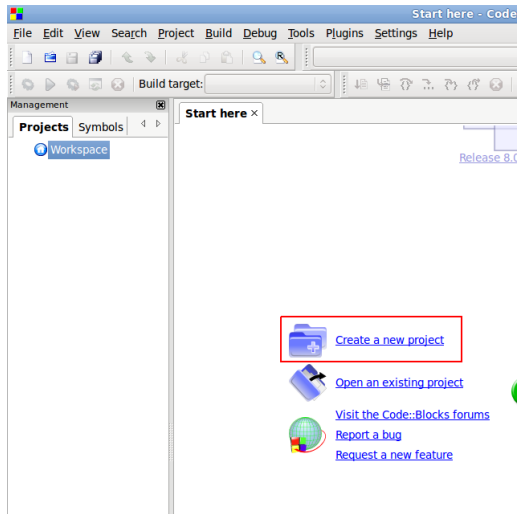
- Entwicklungsumgebung
 - Vorstellung
 - **Installation der Entwicklungsumgebung**
 - Test

Installation

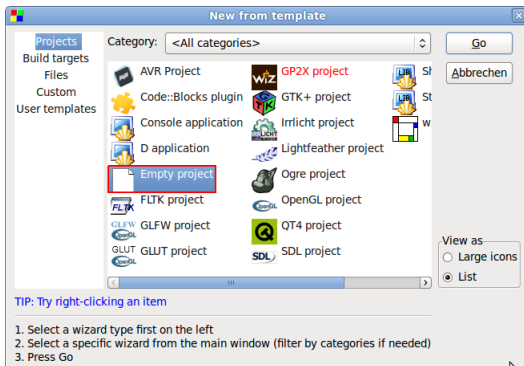
- Ubuntu
 - `sudo apt-get install build-essential codeblocks`
- Windows
 - <http://prdownload.berlios.de/codeblocks/codeblocks-10.05mingw-setup.exe>
- Mac OS
 - <http://prdownload.berlios.de/codeblocks/codeblocks-10.05-p1-mac.dmg>
- Andere
 - <http://www.codeblocks.org/downloads>

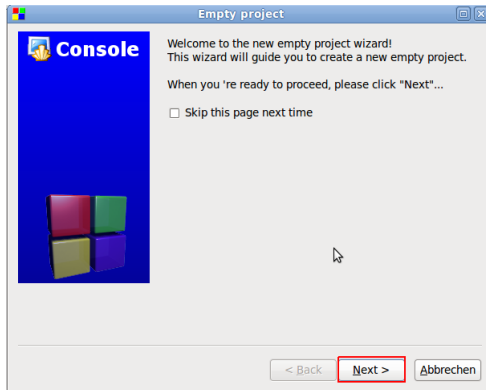
- Entwicklungsumgebung
 - Vorstellung
 - Installation der Entwicklungsumgebung
 - **Test**

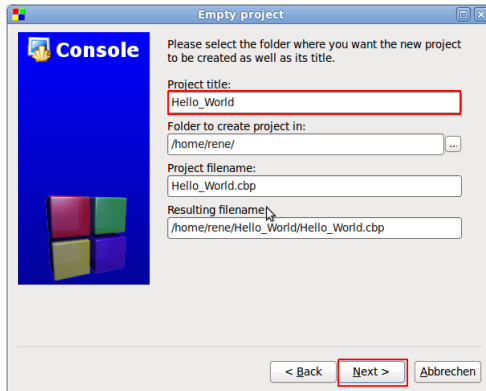
Test

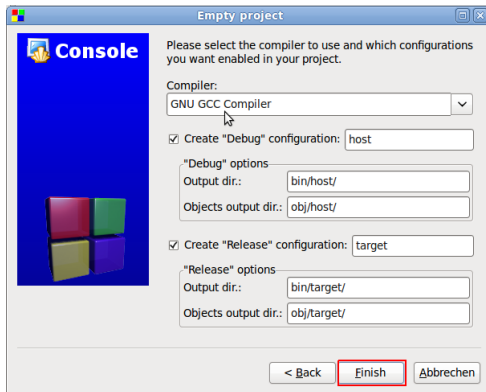


Test

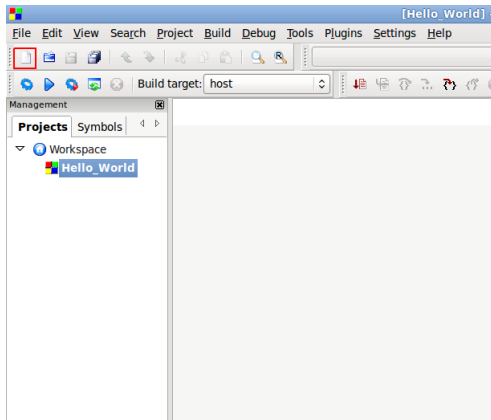


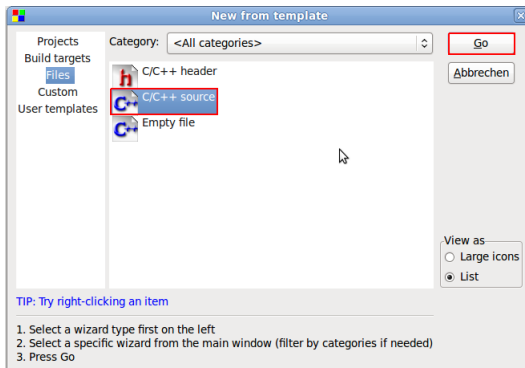


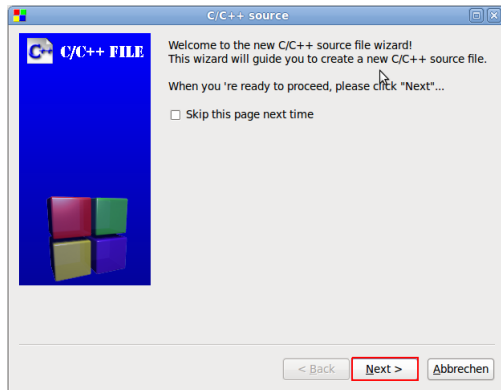




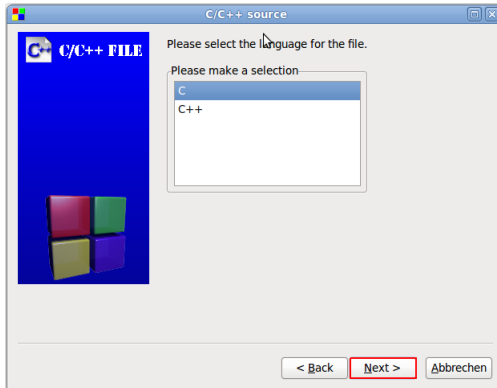
Test



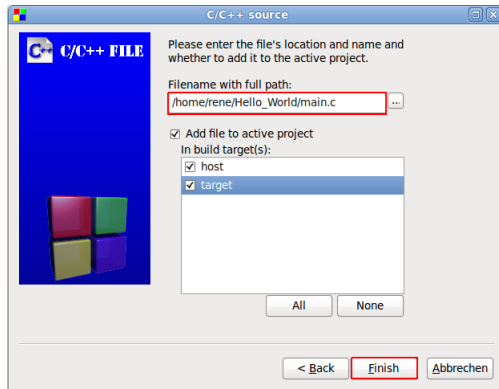




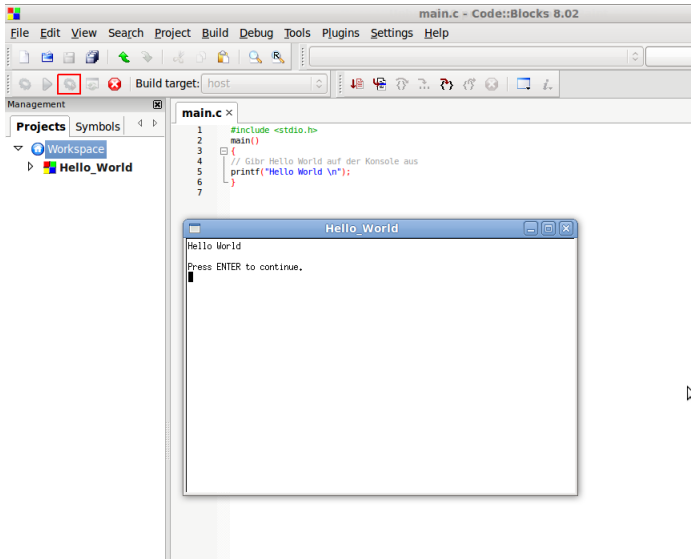
Test



Test



Test



- ① Einleitung
- ② Entwicklungsumgebung
- ③ **Grundlagen C**
- ④ Codingstyle
- ⑤ Große Übung 1
- ⑥ Grundlagen C++
- ⑦ Große Übung 2
- ⑧ Grundlagen SDL
- ⑨ Große Übung 3
- ⑩ Offene Fragen

- **Einleitung**
- Variablen
 - Einfache Datentypen
 - Erweiterte Datentypen
 - Arrays
 - Pointer
- Ausgabe
- Eingabe
- Bedingungen
- Schleifen
- Funktionen
- String-Operationen
- Dateien behandeln
- Dynamischer Speicher
- H-Dateien
- Übungsspiel Spiel

- In den 70ern von Dennis Ritchi erfunden - zur Entwicklung von Unix
- C kann man auf fast jedes System portieren - wirkliche System-Unabhängigkeit (nicht wie bei Java)
- ANSI C besteht aus einem festgelegten Befehlsumfang
- Linux hauptsächlich in C geschrieben
- Sprache ist abhängig von Groß- und Kleinschreibung

Einleitung

Hello World

```
#include <stdio.h>

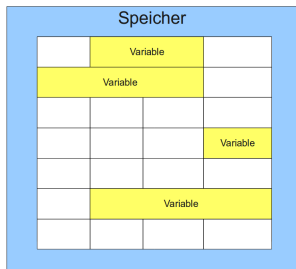
int main(int argc, char *argv[])
{
    // Gibt Hello World auf der Konsole aus
    printf("Hello World \n");
    return 0;
}
```

- Befehlszeilen werden mit ; beendet
- Befehlsblöcke werden in {} zusammen gefasst
- Logisch zusammenhängende Ausdrücke oder Variablen werden in () zusammengefasst
- Main ist die Funktion, die als erstes beim Programmstart ausgeführt wird

- Einleitung
- **Variablen**
 - Einfache Datentypen
 - Erweiterte Datentypen
 - Arrays
 - Pointer
- Ausgabe
- Eingabe
- Bedingungen
- Schleifen
- Funktionen
- String-Operationen
- Dateien behandeln
- Dynamischer Speicher
- H-Dateien
- Übungsspiel Spiel

Variablen

- Sind reservierte Bereiche im Hauptspeicher, zum Speichern von Daten zur Programmlaufzeit
- Es gibt mehrere Variablen-Typen
- Typen richten sich nach den zu speichernden Daten (Zahl, Buchstabe, usw.)
- Die verschiedenen Typen haben einen unterschiedlichen Speicherbedarf



Grundlagen C

- Einleitung
- Variablen
 - **Einfache Datentypen**
 - Erweiterte Datentypen
 - Arrays
 - Pointer
- Ausgabe
- Eingabe
- Bedingungen
- Schleifen
- Funktionen
- String-Operationen
- Dateien behandeln
- Dynamischer Speicher
- H-Dateien
- Übungsspiel Spiel

Variablen - Einfache Datentypen

Typ	Größe	Beschreibung
char	1 Byte	Ein alphanumerisches Zeichen
int	2 Byte	Eine ganze Zahl zwischen -32768 .. 32767
long	4 Byte	Eine ganze Zahl zwischen
	4 Byte	-2147483648 .. 2147483647
unsigned int	2 Byte	Eine ganze Zahl zwischen 0 .. 65535
float	4 Byte	Eine Kommazahl zwischen 1.17E-38 .. 3.4E38
double	8 Byte	Eine Kommazahl zwischen 2.2E-308 .. 1.8E308

Variablen - Einfache Datentypen II

- Deklaration (Bekanntmachen und Beschreiben einer Variable - Reservierung von Speicher)

Deklaration

```
<data_type> var_name;  
<data_type> var_name1, var_name2, ...;
```

Beispiel

```
int i;  
float x,y;
```

Variablen - Einfache Datentypen III

- Initialisierung (Der Variable einen vorab Wert zuweisen)

Initialisierung

```
int i=0;  
float x=0.0f, y=0.0f;
```

- Werte zuweisen

Zuweisung

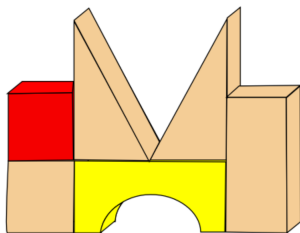
```
i = 1;  
x = 0.5f;
```

Grundlagen C

- Einleitung
- Variablen
 - Einfache Datentypen
 - **Erweiterte Datentypen**
 - Arrays
 - Pointer
- Ausgabe
- Eingabe
- Bedingungen
- Schleifen
- Funktionen
- String-Operationen
- Dateien behandeln
- Dynamischer Speicher
- H-Dateien
- Übungsspiel Spiel

Variablen - Erweiterte Datentypen

- Sind aus Standard-Typen zusammengesetzte Datentypen
- In C Strukturen genannt
- So lassen sich beliebige Typen erstellen



Variablen - Erweiterte Datentypen II

- Erstellung eines Datentyps bzw. einer Struktur

Struktur

```
struct name {  
    int i,  
    ...,  
    char c  
};
```

- Deklaration mit einer Struktur

Deklaration

```
struct name variable_1, ..., variable_m;
```

Variablen - Erweiterte Datentypen III

- Erstellen einer Struktur und Deklaration der Variablen

Deklaration

```
struct name {  
    int i,  
    ...,  
    char c  
}variable_1, ..., variable_m;
```

- Deklaration einer Variable(unter Verwendung einer Struktur) ohne Anlegen eines Datentyps/Struktur

Deklaration ohne Strukturerstellung

```
struct {  
    int i,  
    ...,  
    char c  
}variable_1, ..., variable_m;
```

Variablen - Erweiterte Datentypen V

- Werte zuweisen

Werte zuweisen

```
variable_1.i = 1;  
variable_1.c = 'c'
```

Grundlagen C

- Einleitung
- Variablen
 - Einfache Datentypen
 - Erweiterte Datentypen
 - **Arrays**
 - Pointer
- Ausgabe
- Eingabe
- Bedingungen
- Schleifen
- Funktionen
- String-Operationen
- Dateien behandeln
- Dynamischer Speicher
- H-Dateien
- Übungsspiel Spiel

Arrays

- Mit Arrays kann man Listen oder Tabellen abbilden
- Es werden Datenbereiche nacheinander im Speicher reserviert

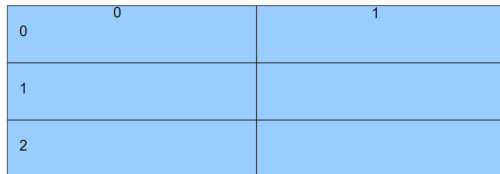
int i;



int i[3];



int i[3][2];



Arrays II

- Deklaration eines Array mit 5 Elementen (Liste)

Deklaration

```
int    i[5];
```

- In den Klammern wird angegeben wie oft das Element zur Verfügung stehen soll

Arrays III

- Initialisierung der ersten drei Elemente mit 1,2 und 3, der Rest wird mit dem Wert 0 Belegt

Deklaration und Initialisierung

```
int i[5]={1, 2, 3};
```

- Deklarieren von 3 Elementen und der gleichzeitigen Zuweisung von 1,2 und 3

Deklaration und Initialisierung

```
int i[]={1, 2, 3};
```

- Den 5. und letzten Element den Wert 3 zuweisen

Zuweisung

```
i[4] = 3;
```

- Es wird bei 0 anfangen zu zählen

- Deklarieren und initialisieren von Character Arrays

Deklaration und Initialisierung

```
char Letters[10]={'f','a','x',' ','m','o','d','e','m','\0'};  
char Text[10]="fax modem";  
//Zeileende automatisch hinzugefügt
```

- Deklaration eines Array mit 10 Zeilen und 4 Spalten - 10 Elemente (Tabelle)

Deklaration

```
int Matrix[10][4];
```

- In den Klammern wird angegeben wie oft das Element zur Verfügung stehen soll (Spalte und Zeile)

Arrays VII

- Initialisierung eines zweidimensionalen Arrays

Initialisierung

```
int Matrix[3][2]={  
    {1, 2},  
    {3, 4},  
    {5, 6}  
};
```

- Initialisieren eines zweidimensionalen Arrays ohne Angabe der ersten [4]

Initialisierung

```
int Matrix[][2]={  
    {1, 2},  
    {3, 4},
```

- Initialisieren von Character in einem zweidimensionalen Arrays

Initialisierung eines zweidimensionalen Arrays

```
char Matrix[4][4]={  
    {"Hell"},  
    {"HoHo"},  
    {"HiHo"},  
    {"Lolo"}  
};
```

Arrays VIII

- Wert 3 zuweisen

Zuweisung

```
Matrix[3][2] = 3;
```

- Es wird bei 0 anfangen zu zählen


Grundlagen C

- Einleitung
- Variablen
 - Einfache Datentypen
 - Erweiterte Datentypen
 - Arrays
 - **Pointer**
- Ausgabe
- Eingabe
- Bedingungen
- Schleifen
- Funktionen
- String-Operationen
- Dateien behandeln
- Dynamischer Speicher
- H-Dateien
- Übungsspiel Spiel

Pointer

- Ein Pointer enthält eine Speicher-Adresse zu einer Variable oder zu einem anderen Pointer

Adresse	Speicher	Datentyp
HEX 00		
HEX 01	3	Int (int i)
HEX 02		
HEX 03	HEX 01	Pointer (int *p)



Pointer II

- Anlegen einer Variable i und eines Pointer, Pointer werden mit * vor dem Variablennamen deklariert

Beispiel

```
int i = 3;  
Int *p;  
P = &i;
```

Pointer III

- Theoretische Ausgabe

Direkte Ausgabe von p

p - HEX 01

Mit * vor dem Pointernamen wird der Inhalt der Adresse ausgegeben

*p - 3

Gibt die Adresse des Pointers und von i aus

&p - HEX 05

&i - HEX 01

- Einleitung
- Variablen
 - Einfache Datentypen
 - Erweiterte Datentypen
 - Arrays
 - Pointer
- **Ausgabe**
- Eingabe
- Bedingungen
- Schleifen
- Funktionen
- String-Operationen
- Dateien behandeln
- Dynamischer Speicher
- H-Dateien
- Übungsspiel Spiel

Ausgabe

- Standard Ausgabe ist printf in der stdio.h

Ausgabe Text

```
printf("Hallo!\n");
```

Ausgabe von Variablen

```
printf("I ist %d und c %c und f %.2f\n", i,c,f);
```

<i>specifier</i>	Output	Example
c	Character	a
d or i	Signed decimal integer	392
e	Scientific notation (mantisse/exponent) using e character	3.9265e+2
E	Scientific notation (mantisse/exponent) using E character	3.9265E+2
f	Decimal floating point	392.65
g	Use the shorter of %e or %f	392.65
G	Use the shorter of %E or %f	392.65
o	Signed octal	610
s	String of characters	sample
u	Unsigned decimal integer	7235
x	Unsigned hexadecimal integer	7fa
X	Unsigned hexadecimal integer (capital letters)	7FA
p	Pointer address	B800:0000
n	Nothing printed. The argument must be a pointer to a signed int, where the number of characters written so far is stored.	
%	A % followed by another % character will write % to stdout.	

- ÜBUNG!!!!

Grundlagen C

- Einleitung
- Variablen
 - Einfache Datentypen
 - Erweiterte Datentypen
 - Arrays
 - Pointer
- Ausgabe
- **Eingabe**
- Bedingungen
- Schleifen
- Funktionen
- String-Operationen
- Dateien behandeln
- Dynamischer Speicher
- H-Dateien
- Übungsspiel Spiel

Eingabe

- Standard Input ist scanf in der stdio.h

Eingabe Zahl

```
scanf("%d",&i);
```

- Erwartet einen Pointer als Eingabevariable
- %* - Typen sind vergleichbar mit printf

Eingabe Zeichen

```
scanf("%c",array);
```

- Ohne & bei Arrays, da ohne [] automatisch die adresse des ersten Elements übergeben wird

- Einleitung
- Variablen
 - Einfache Datentypen
 - Erweiterte Datentypen
 - Arrays
 - Pointer
- Ausgabe
- Eingabe
- **Bedingungen**
- Schleifen
- Funktionen
- String-Operationen
- Dateien behandeln
- Dynamischer Speicher
- H-Dateien
- Übungsspiel Spiel

Bedingungen

- Trifft Entscheidung je nachdem ob eine Bedingung war oder falsch ist.

Entscheidung in C

```
if(i == 1){  
    //Mach was  
}
```

- Wenn nur die nächste Zeile nach der IF bearbeitet werden soll

Entscheidung in C

```
if(i == 1)  
    //Mach was
```

Vergleichsoperatoren

- $A == B$ - A gleich B nur ein $=$ funktioniert nicht
- $A < B$ - A kleiner B
- $A > B$ - A größer B
- $A \leq B$ - A kleiner oder gleich B
- $A \geq B$ - A größer gleich B
- $A \neq B$ - A ungleich B

Bedingungen III

- Das nicht Eintreffen einer Bedingung behandeln

Else

```
int i = 5;
if(i == 1){
    //Hier soll was passieren wenn i 1 ist
}else{
    //Hier soll was passieren wenn i nicht 1 ist
}
```

Bedingungen IV

- Auf verschiedene Ereignisse reagieren

Else If

```
int i = 5;
if(i == 1){
    // wenn i gleich 1
}else if(i == 5){
    // wenn i gleich 5
}else{
    //wenn i was anderes als 1 und 5
}
```

- Auf auf mehrere Bedingungen reagieren

AND OR

```
int i = 1;
int j = 3;
char c = 'a';
if((i = 1 and j < 6) or c = 'a'){
    // Mach was
}
```

Bedingungen VI

- Viele Werte Unterscheiden

Switch

```
switch(i){  
    case 2:  
        Wenn 2 dann mach hier  
        break;  
    case 4:  
        Wenn 4 dann mach hier  
        break;  
    default:  
        Wenn was anderes dann  
}
```

- ÜBUNG!!!!

Grundlagen C

- Einleitung
- Variablen
 - Einfache Datentypen
 - Erweiterte Datentypen
 - Arrays
 - Pointer
- Ausgabe
- Eingabe
- Bedingungen
- **Schleifen**
- Funktionen
- String-Operationen
- Dateien behandeln
- Dynamischer Speicher
- H-Dateien
- Übungsspiel Spiel

Schleifen

- Eine Schleife wiederholt einen Anweisungsblock solange, wie eine bestimmte Bedingung war ist.
- Anweisungsblock wird wiederholte solange Bedingung war ist

while

```
while(bedingung){  
    //Anweisungen  
}
```

- Anweisungsblock wird erst einmal durchlaufen, dann wird Bedingung geprüft

do while

```
Do{  
    //Anweisung  
}while(bedingung)
```

Schleifen II

Beispiel

```
int i = 0;
while(i < 10){
    //Anweisungen
    i++; // oder i = i +1
}
```

- Wenn man eine Variable automatisch addieren lassen möchte:

For Schleife

```
int i = 0;  
  
for(i = 1;i<10;i++){  
    //Anweisung  
}
```

- ÜBUNG!!!!

Grundlagen C

- Einleitung
- Variablen
 - Einfache Datentypen
 - Erweiterte Datentypen
 - Arrays
 - Pointer
- Ausgabe
- Eingabe
- Bedingungen
- Schleifen
- **Funktionen**
- String-Operationen
- Dateien behandeln
- Dynamischer Speicher
- H-Dateien
- Übungsspiel Spiel

Funktionen

- Zusammen Fassen von Code-Stücken in eine eigen Einheit
- Code-Stücke werden so wieder verwendbar
- Steigert die Übersichtlichkeit

Funktionsaufbau

```
<rückgabety> funktionsname(übergabewerte, ...){  
    //Code  
}
```

- Void als Rückgabewert bedeutet das die Funktion nichts zurück gibt

Funktionen II

Beispiel

```
int add_a_b(){
    int a = 2;
    int b = 3;
    return a + b;
}

int main(){
    int result = 0;
    result = add_a_b();
    printf("%d",result);
    return 0;
}
```

- Ausgabe: 5

Funktionen III

Beispiel Werteübergabe

```
int add_a_b(int a, int b){  
    return a + b;  
}
```

```
int main(){  
    int a = 2;  
    int b = 4;  
    int result = 0;  
    result = add_a_b(a,b);  
    printf("%d",result);  
    return 0;  
}
```

- Ausgabe: 6

Beispiel Werteübergabe II

```
int add_a_b(int a, int b){
    a = 8;
    return a + b;
}

int main(){
    int a = 2;
    int b = 4;
    int result = 0;
    result = add_a_b(a,b);
    printf("%d",a);
    return 0;
}
```

- Ausgabe von a: 2

Beispiel Referenzübergabe

```
int add_a_b(int *a, int b){
    *a = 8;
    return a + b;
}

int main(){
    int a = 2;
    int ba = 4;
    int result = 0;
    result = add_a_b(&a,b);
    printf("%d",a);
    return 0;
}
```

- Ausgabe von a: 8

Funktionen VI

Beispiel Referenzübergabe von Arrays

```
int add_a_b(int a[], int b){ // möglich auch int a[8]
    a[0] = 8;
    return a + b;
}

int main(){
    int a[9] = {0};
    int b = 4;
    int result = 0;
    result = add_a_b(a,b);
    printf("%d",a[0]);
    return 0;
}
```

- Ausgabe von a[0]: 8

Funktionen VII

Beispiel Prototyp

```
int add_a_b(int a[], int b); //Prototyp
int main(){
    int a[9] = {0};
    int b = 4;
    int result = 0;
    result = add_a_b(a,b);
    printf("%d",a[0]);
    return 0;
}
int add_a_b(int a[], int b){ // möglich auch int a[8]
    a[0] = 8;
    return a + b;
}
```

- Ausgabe von a[0]: 8

- ÜBUNG!!!!

Grundlagen C

- Einleitung
- Variablen
 - Einfache Datentypen
 - Erweiterte Datentypen
 - Arrays
 - Pointer
- Ausgabe
- Eingabe
- Bedingungen
- Schleifen
- Funktionen
- **String-Operationen**
- Dateien behandeln
- Dynamischer Speicher
- H-Dateien
- Übungsspiel Spiel

String-Operationen

- Um den Umgang von Text in Char-Arrays zu erleichtern bietet die `string.h` hilfreiche Funktionen. Hier ein paar Beispiele der wichtigsten Funktionen.

Beispiel Prototyp

```
char source[]="Sample string";
char dest[40];
int laenge = 5;

strcpy (dest,source); //String kopieren

strncpy (dest,source,laenge); //Ein bestimmten Teil kopieren

int r = strcmp (source,"Hi"); /* String vergleichen
    Wenn Rückgabe 0, dann String gleich.
    Wenn ungleich Null wird die erste ungleiche
    position zurückgegeben*/

laenge = strlen(source); //Länge des Strings
```


- ÜBUNG!!!!

Grundlagen C

- Einleitung
- Variablen
 - Einfache Datentypen
 - Erweiterte Datentypen
 - Arrays
 - Pointer
- Ausgabe
- Eingabe
- Bedingungen
- Schleifen
- Funktionen
- String-Operationen
- **Dateien behandeln**
- Dynamischer Speicher
- H-Dateien
- Übungsspiel Spiel

Dateien behandeln

- Wenn man Dateien lesen oder schreiben möchte, benutzt man eine Funktion aus der `stdio.h`
- Damit öffnet man einen Stream zu einer Datei

Aufbau `fopen`

```
FILE *<Filepoint> = fopen(char *<filename>, char *<modus>);
```

- Schließt den Stream zu einer Datei

Aufbau `fclose`

```
fclose(FILE *<filepointer>);
```

Dateien behandeln II

- Die verschiedenen Zugriffsmodus

Zugriffsmodus

r	Datei lesen, Datei muss existieren
w	Erzeugt eine Datei zum beschreiben, wenn diese Existiert, wird der Inhalt überschrieben
a	Daten am Ende einer Datei anfügen. Datei wird erzeugt wenn sie nicht existiert
r+	Lesen und beschreiben einer Datei. Datei muss existieren
w+	Erzeugen und Lesen/schreiben einer Datei. wenn diese Existiert, wird der Inhalt überschrieben
a+	Lesen und schreiben, Daten am Ende einer Datei anfügen. Datei wird erzeugt wenn sie nicht existiert

- Um die Datei im Binerymode zu öffnen fügt man ein b dem modus zu
z.B. rb oder a+b

Dateien behandeln III

- Öffnen einer Datei zum lesen (in Bytemode)

Beispiel Datei öffnen

```
void main(){  
    FILE *pFile;  
    pFile = fopen("meine-datei.txt","rb");  
    fclose(pFile);  
}
```

- Einlesen eines Datensatzes:

Einlesen

```
fgets(char *<str>,int <länge>,FILE *filepointer)
```

- Damit liest man entweder bis zum aktuellen Zeilenende oder bis die angegebene Länge erreicht ist

Datei Funktionen

- Öffnen einer Datei zum lesen (in Bytemode) und auslesen der ersten Zeile oder die erste Zeile bis Position 128

Beispiel Einlesen

```
void main(){
    FILE *pFile;
    char str[128];
    pfile = fopen("meine-datei.txt","rb");
    //gegebenfalls Prüfung auf erfolg einfügen
    fgets(str,128,pfile);
    printf("%s\n",str);
    fclose(pFile);
}
```

Dateien behandeln V

- Öffnen einer Datei zum lesen (in Bytemode) einer Zeile bis Position 128/Zeilenende , bis zum Dateiende

Beispiel Einlesen

```
void main(){
    FILE *pFile;
    Char str[128];
    pFile = fopen("meine-datei.txt","rb");
    while(!feof(pFile)){
        fgets(str,128,pfile);
        printf("%s\n",str);
    }
    fclose(pFile);
}
```

- TIPP: Um den ausgelesenen String zu bearbeiten nutzt die

Dateien behandeln VI

- Öffnen eine Datei zum schreiben (in Bytemode)

Beispiel Schreiben

```
void main(){
    FILE *pFile;
    Char str[] = {"Hallo du!"};
    pFile = fopen("meine-datei.txt","wb");
    //gegebenfalls Prüfung auf erfolg einfügen
    fputs(str,pfile);
    printf("%s\n",str);
    fclose(pFile);
}
```

- ÜBUNG!!!!

Grundlagen C

- Einleitung
- Variablen
 - Einfache Datentypen
 - Erweiterte Datentypen
 - Arrays
 - Pointer
- Ausgabe
- Eingabe
- Bedingungen
- Schleifen
- Funktionen
- String-Operationen
- Dateien behandeln
- **Dynamischer Speicher**
- H-Dateien
- Übungsspiel Spiel

Beispiel

```
void main(){
    char name[3];
    printf("Leg los: ");
    scanf("%s",name);
    printf("%s",name);
}
```

- Die Eingabe von mehr als 3 Zeichen führt zu Fehlern!

Dynamischer Speicher II

Beispiel

```
void main(){
    char *name = NULL;
    int l;
    printf("Wie lang soll deine Eingabe sein: ");
    scanf("%d", &l);
    name = (char*) malloc (l * sizeof(char));
    if (name==NULL) exit (1);
    printf("Leg los: ");
    scanf("%s",name);
    printf("%s",name);
    free(name);
}
```

- Speicher wird zur Laufzeit reserviert.

Dynamischer Speicher III

- Speicher wird zur Laufzeit neu reserviert.

Beispiel

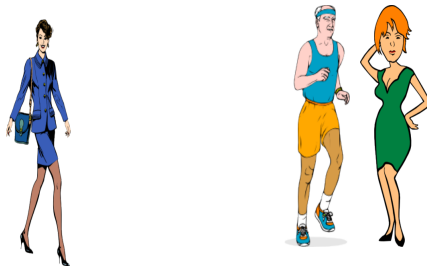
```
void main(){
    char *name,*tmp;
    int l;
    name = (char*) malloc (1);
    printf("Wie lang soll deine Eingabe sein: ");
    scanf("%d", &l);
    tmp = (char*) realloc (name,l * sizeof(char));
    name = tmp;
    if (name==NULL) exit (1);
    printf("Leg los: ");
    scanf("%s",name);
    printf("%s",name);
    free(name);
}
```

- ÜBUNG!!!!

Grundlagen C

- Einleitung
- Variablen
 - Einfache Datentypen
 - Erweiterte Datentypen
 - Arrays
 - Pointer
- Ausgabe
- Eingabe
- Bedingungen
- Schleifen
- Funktionen
- String-Operationen
- Dateien behandeln
- Dynamischer Speicher
- **H-Dateien**
- Übungsspiel Spiel

- Stellt euch vor ihr habt eine Schwester und die bringt ihren neuen Freund ohne Vorwarnung mit nach Hause um ihn euren Eltern vorzustellen



- Ihr neuer Freund!!



- Wie wird dein Vater wohl reagieren?



- Dieses mal bereitet eure Schwester deine Eltern vor:
 - Er sieht Verrückt aus
 - Hat zwei Doktor Titel
 - Eine Firma
 - Geschätztes Vermögen 32Mio
 - Kennengelernt beim Singelkochen für schüchterne Männer
 - Kennst Ihn nun schon 3 Jahre und seit schon 1 Jahr zusammen
 - Liebste Person der Welt
 - WILL KEIN SEX VOR DER EHE

- Dieses mal bereitet eure Schwester deine Eltern vor:
 - Er sieht Verrückt aus
 - Hat zwei Doktor Titel
 - Eine Firma
 - Geschätztes Vermögen 32Mio
 - Kennengelernt beim Singelkochen für schüchterne Männer
 - Kennst Ihn nun schon 3 Jahre und seit schon 1 Jahr zusammen
 - Liebste Person der Welt
 - WILL KEIN SEX VOR DER EHE

- Dieses mal bereitet eure Schwester deine Eltern vor:
 - Er sieht Verrückt aus
 - Hat zwei Doktor Titel
 - Eine Firma
 - Geschätztes Vermögen 32Mio
 - Kennengelernt beim Singelkochen für schüchterne Männer
 - Kennst Ihn nun schon 3 Jahre und seit schon 1 Jahr zusammen
 - Liebste Person der Welt
 - WILL KEIN SEX VOR DER EHE

- Dieses mal bereitet eure Schwester deine Eltern vor:
 - Er sieht Verrückt aus
 - Hat zwei Doktor Titel
 - Eine Firma
 - Geschätztes Vermögen 32Mio
 - Kennengelernt beim Singelkochen für schüchterne Männer
 - Kennst Ihn nun schon 3 Jahre und seit schon 1 Jahr zusammen
 - Liebste Person der Welt
 - WILL KEIN SEX VOR DER EHE

- Dieses mal bereitet eure Schwester deine Eltern vor:
 - Er sieht Verrückt aus
 - Hat zwei Doktor Titel
 - Eine Firma
 - Geschätztes Vermögen 32Mio
 - Kennengelernt beim Singelkochen für schüchterne Männer
 - Kennst Ihn nun schon 3 Jahre und seit schon 1 Jahr zusammen
 - Liebste Person der Welt
 - WILL KEIN SEX VOR DER EHE

- Dieses mal bereitet eure Schwester deine Eltern vor:
 - Er sieht Verrückt aus
 - Hat zwei Doktor Titel
 - Eine Firma
 - Geschätztes Vermögen 32Mio
 - Kennengelernt beim Singelkochen für schüchterne Männer
 - Kennst Ihn nun schon 3 Jahre und seit schon 1 Jahr zusammen
 - Liebste Person der Welt
 - WILL KEIN SEX VOR DER EHE

- Dieses mal bereitet eure Schwester deine Eltern vor:
 - Er sieht Verrückt aus
 - Hat zwei Doktor Titel
 - Eine Firma
 - Geschätztes Vermögen 32Mio
 - Kennengelernt beim Singelkochen für schüchterne Männer
 - Kennst Ihn nun schon 3 Jahre und seit schon 1 Jahr zusammen
 - Liebste Person der Welt
 - WILL KEIN SEX VOR DER EHE

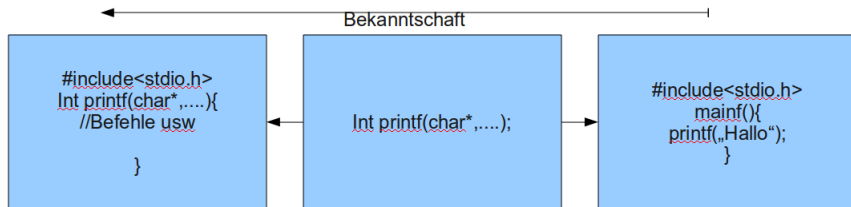
- Dieses mal bereitet eure Schwester deine Eltern vor:
 - Er sieht Verrückt aus
 - Hat zwei Doktor Titel
 - Eine Firma
 - Geschätztes Vermögen 32Mio
 - Kennengelernt beim Singelkochen für schüchterne Männer
 - Kennst Ihn nun schon 3 Jahre und seit schon 1 Jahr zusammen
 - Liebste Person der Welt
 - WILL KEIN SEX VOR DER EHE

Header

- Header Dateien macht Objekte (Typen Variablen Methoden) einem Programm bekannt und wie folgt eingebunden:

Header

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



- ① Einleitung
- ② Entwicklungsumgebung
- ③ Grundlagen C
- ④ **Codingstyle**
- ⑤ Große Übung 1
- ⑥ Grundlagen C++
- ⑦ Große Übung 2
- ⑧ Grundlagen SDL
- ⑨ Große Übung 3
- ⑩ Offene Fragen

- Vereinbarung und Empfehlungen wie man Code formatieren sollte
 - Einrücktiefe
 - Formatierung
 - Funktionen
 - Variablen

Codingstyle

- Die Einrücktiefe beträgt genau 8 Zeichen und nicht mehr als 5 Einrücktiefen.

```
main(){  
    for(int i = 0; i < 1701;i++){  
        if(i == 2)  
            printf(„Toll!“);  
    }  
}
```

```
main(){  
    for(int i = 0; i < 1701;i++){  
        if(i == 2)  
            printf(„Toll!“);  
    }  
}
```

```
main(){  
    for(int i = 0; i < 1701;i++){  
        if(i == 2)  
            printf(„Toll!“);  
    }  
}
```


Coding Style

- Die Klammerung von Funktionen, If und anderen Objekten die {} verwenden

Klammerung

```
main()
{
    if (i == j){
        //Mach was
    }
}
```

- Zu lange Zeilen (viel länger als Bildschirm breit):

Klammerung

```
machwas(7,5,&das,4,8,variable,variable2,
        Variable3,5,8,tmp);
```

Coding Style - Funktionen

- eine Funktion sollte nie länger als eine Bildschirmseite sein (Ausnahme switch)
- eine Funktion hat genau eine Aufgabe
- Um so mehr Funktion, um sehr mehr sinkt die Fehlerwahrscheinlichkeit

Coding Style - Variablen

- Variablen sollten kurz und sexy sein und nicht:
ThisVariablesATemporaryCounter
- Variablen sparsam einsetzen
- Namensgebung gleichmäßig durchziehen

- ① Einleitung
- ② Entwicklungsumgebung
- ③ Grundlagen C
- ④ Codingstyle
- ⑤ **Große Übung 1**
- ⑥ Grundlagen C++
- ⑦ Große Übung 2
- ⑧ Grundlagen SDL
- ⑨ Große Übung 3
- ⑩ Offene Fragen

Große Übung 1

- Ziel der Übung: Programmierung vom Spiel Zahlenraten
 - ① Spieler 1 gibt eine Zahl ein
 - ② Der Bildschirm wird gelöscht*
 - ③ Spieler 2 beginnt nun zu raten
 - ④ Ist die eingegebene Zahl zu klein oder zu groß, soll das als Meldung ausgegeben werden
 - ⑤ Ist die Zahl richtig erraten, bekommt der Spieler die Anzahl der Versuche angezeigt
 - ⑥ Als letztes Frag das Programm ob der Spieler nochmals spielen möchte
- * Tipp: Zum löschen des Bildschirm: `#define CLS printf("\033[2J")`

Übung 1 - Lösung

Lösungs-Beispiel

```
#include<stdio.h>
#define CLS printf("\033[2J")

int main()
{
    int zuraten;
    int versuch;
    int trys;
    char yesno = 'j';
    int okay = 1;
    printf("#####\n");
    printf("#####\n");
    printf("##### Zahlenraten #####\n");
    printf("#####\n");
    printf("#####\n\n\n\n");
```

Übung 1 - Lösung

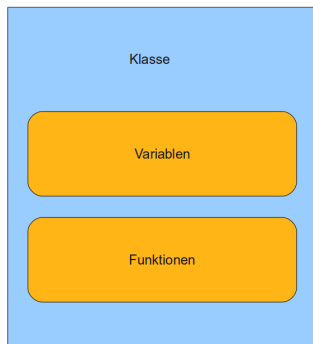
Lösungs-Beispiel

```
while (okay == 1) {
    zuraten = 0;
    versuch = 0;
    printf("Hallo Spieler 1 gib eine Zahl ein : ");
    scanf("%d",&zuraten);
    CLS;
    for (trys = 1; zuraten != versuch; trys++) {
        printf("\nSo Spieler 2, jetzt rate mal die gesuchte Zahl : ");
        scanf("%d",&versuch);

        if (zuraten > versuch) {
            printf("\n Deine Zahle %d ist zu klein!\n ", versuch);
        } else if (zuraten < versuch) {
            printf("\n Deine Zahle %d ist zu groß!\n ", versuch);
        } else if (zuraten == versuch) {
            printf("\n Deine Zahle %d ist richtig und du hast %d Versuche benötigt!!!! ",
                trys);
        }
    }
    getchar();
    printf("\n Noch ein Spiel j/n?\n");
    scanf("%c",&yesno);
    if (yesno != 'j') {
        okay = 0;
    }
}
return 0;
}
```

- ① Einleitung
- ② Entwicklungsumgebung
- ③ Grundlagen C
- ④ Codingstyle
- ⑤ Große Übung 1
- ⑥ **Grundlagen C++**
- ⑦ Große Übung 2
- ⑧ Grundlagen SDL
- ⑨ Große Übung 3
- ⑩ Offene Fragen

- Objekt orientierte Erweiterung
- Variablen und Funktionen können in Klassen verpackt werden
- Von einer Klasse können mehrere Instanzen generiert werden
- Substituiert C
- Daten Deklaration auch nach Programmanweisungen möglich



① Grundlagen C++

① **Klassen**

② Vererbung

③ Polymorphy

C++ - Klassen

Beispiel Klasse

```
class meine_klasse{
    private:
        int a,b;
    Public:
        int result;
        meine_klasse(int x, int y){
            A = x;
            B = y;
        };
        int add_function(){
            result = a + b;
            return result;
        };
        ~meine_klasse(){
            //Aufräumen
        };
};
```

Beispiel Klasse

```
int main(){
    meine_klasse meine(3,4);
    meine.add_function();
    printf(„%d\\,meine.reslut);
    return 0;
};
```

- Ausgabe von mein.result ist 7
- Ausgabe von mein.a oder mein.b ergibt einen Fehler

C++ - Klassen III

Beispiel Klasse

```
class meine_klasse{
    private:
        int a,b;
    Public:
        int result;
        meine_klasse(int x, int y){
            A = x;
            B = y;
        };
        int add_function(){
            result = a + b;
            return result;
        };
        ~meine_klasse(){
            //Aufräumen
        };
};
```

Beispiel Klasse

```
int main(){
    meine_klasse *instanz1;
    instanz1 = new meine_klasse(3,4);
    meine_klasse *instanz2 = new meine_klasse(5,5);
    instanz1->add_function();
    printf(„%d\\,instanz1->reslut);
    printf(„%d\\,instanz2->reslut);
    return 0;
};
```

- Ausgabe von instanz1->result ist 7
- Ausgabe von instanz2->result ist 10
- Ausgabe von mein.a oder mein.b ergibt einen Fehler

Beispiel Klasse

```
class meine_klasse{
    private:
        int a,b;
    Public:
        int result;
        meine_klasse(int x, int y){
            A = x;
            B = y;
        };
        int add_function();
        ~meine_klasse(){
            //Aufräumen
        };
};
```

Beispiel Klasse

```
int main(){
    meine_klasse meine(3,4);
    meine.add_function();
    printf(„%d\\,meine.reslut);
    return 0;
};

int meine_klasse::add_function(){
    result = a + b;
    return result;
};
```

- Ausgabe von mein.result ist 7
- Ausgabe von mein.a oder mein.b ergibt einen Fehler

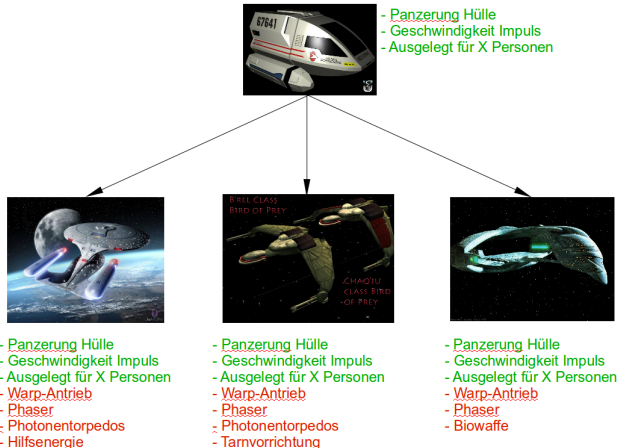
① Grundlagen C++

① Klassen

② **Vererbung**

③ Polymorphy

C++ - Vererbung



C++ - Vererbung

Beispiel Klasse

```
class erben
{
    private:
        int x;
    protected:
        int y;
    public:
        int k;
        erben()
        {
            k = 15;
            x = k;
            y = k;
        };
};
```

C++ - Vererbung II

Beispiel Klasse

```
class meine_klasse:public erben
{
private:
    int a;
protected:
    int b;
public:
    int result;
    meine_klasse(int x, int y):erben()
    {
        a = x;
        b = y;
    };
    int add_function(){
        result = a + b;
        return result;
    };
    ~meine_klasse()
    {
        //Aufräumen
    };
};
```

C++ - Vererbung III

Beispiel Klasse

```
int main()
{
    meine_klasse *meine = new meine_klasse(3,4);
    meine->add_function();
    printf("%d\n",meine->result);
    printf("%d",meine->k);
    return 0;
};
```

- Ausgabe von mein.result ist 7
- Ausgabe von mein.a oder mein.b ergibt einen Fehler

- ① Grundlagen C++
 - ① Klassen
 - ② Vererbung
 - ③ **Polymorphy**

Beispiel Klasse

```
class tier
{
public:
    virtual void gib_laut(){
        printf("Ich mach nix!!!!\n");
    };
};
```

C++ - Polymorphy II

Beispiel Klasse

```
class kuh:public tier
{
public:
    void gib_laut(){
        printf("Mmmmuuuuhhhhh!!!!\n");
    };
};

class hund:public tier
{
public:
    void gib_laut(){
        printf("Wufffff!!!!\n");
    };
};

void sprich(tier *t)
{
    t->gib_laut();
}
```


C++ - Polymorphy III

Beispiel Klasse

```
int main()
{
    tier *t[3];

    t[0] = new kuh();
    t[1] = new hund();
    t[2] = new tier();

    sprich(t[0]);
    sprich(t[1]);
    sprich(t[2]);

    return 0;
};
```

- Ausgabe von mein.result ist 7
- Ausgabe von mein.a oder mein.b ergibt einen Fehler

- ① Einleitung
- ② Entwicklungsumgebung
- ③ Grundlagen C
- ④ Codingstyle
- ⑤ Große Übung 1
- ⑥ Grundlagen C++
- ⑦ **Große Übung 2**
- ⑧ Grundlagen SDL
- ⑨ Große Übung 3
- ⑩ Offene Fragen

- Ziel der Übung: Programmierung vom Spiel Würfeln
 - ① Der Computer gibt eine Zahl zwischen 0 und 100 vor
 - ② Der Spieler1 würfelt und entscheidet danach ob er nochmal würfeln möchte
 - ③ Der Spieler1 würfelt solange bis er nicht mehr möchte, wenn er höher ist als die vorgegebene Zahl, dann hat er verloren
 - ④ Dann beginnt Spieler2 zu würfeln
 - ⑤ Gewonnen hat der der am nächsten an der vorgegeben Zahl dran ist
 - * Tipp: Zufalls-Funktion rand()

- ① Einleitung
- ② Entwicklungsumgebung
- ③ Grundlagen C
- ④ Codingstyle
- ⑤ Große Übung 1
- ⑥ Grundlagen C++
- ⑦ Große Übung 2
- ⑧ **Grundlagen SDL**
- ⑨ Große Übung 3
- ⑩ Offene Fragen

- **Einleitung**
- Entwicklungsumgebung vorbereiten
- SDL Initialisierung
- Bild anzeigen
- Weitere Bildformate
- Transparenter Hintergrund
- Bildteil laden
- Schrift
- Ereignisse
- Sound Ausgabe

- Standardisierte Schnittelle zur Grafik, Sound, Eingabe usw.
- In C geschrieben
- Kann durch diverse Bibliotheken erweitert werden
- Ist für 2D Spiele konzipiert kann aber durch z.B. OpenGL erweitert werden
- Es müssen die nötigen Bibliotheken vor der Anwendung installiert werden

- Einleitung
- **Entwicklungsumgebung vorbereiten**
- SDL Initialisierung
- Bild anzeigen
- Weitere Bildformate
- Transparenter Hintergrund
- Bildteil laden
- Schrift
- Ereignisse
- Sound Ausgabe

- Einleitung
- Entwicklungsumgebung vorbereiten
- **SDL Initialisierung**
- Bild anzeigen
- Weitere Bildformate
- Transparenter Hintergrund
- Bildteil laden
- Schrift
- Ereignisse
- Sound Ausgabe

SDL Initialisierung

- Einbinden der nötigen Header

Header einbinden

```
#include "SDL/SDL.h"
```

- Vor der Verwendung der SDL-Funktionen muss SDL initialisiert werden

Initialisierung

```
SDL_Init( SDL_INIT EVERYTHING );
```

- Was initialisiert wird, muss auch beendet werden

SDL Schließen

```
SDL_Quit();
```

SDL Initialisierung

- Damit wir was anzeigen können brauchen wir ein Fenster, das wir wie folgt erzeugen

Fenster erzeugen

```
SDL_Surface* fenster = SDL_SetVideoMode( 640, 480, 32, SDL_SWSURFACE );
```

- Die Funktion gibt eine Referenz (Pointer) auf ein SDL Oberfläche (In diesem Fall unser Fenster) zurück die wir in einer Variable vom Typ `SDL_Surface` speichern
- 640 und 480 gibt die Auflösung an
- 32 die Farbtiefe
- `SDL_SWSURFACE` setzt das Fenster im Speicher
- Die Hauptoberfläche "fenster" wird mit `SDL_Quit` wieder freigegeben

SDL Initialisierung

- Fenster eine Bezeichnung geben

Fenstername

```
SDL_WM_SetCaption( "Mein erstes SDL-Spiel", NULL );
```

SDL Initialisierung

Beispiel

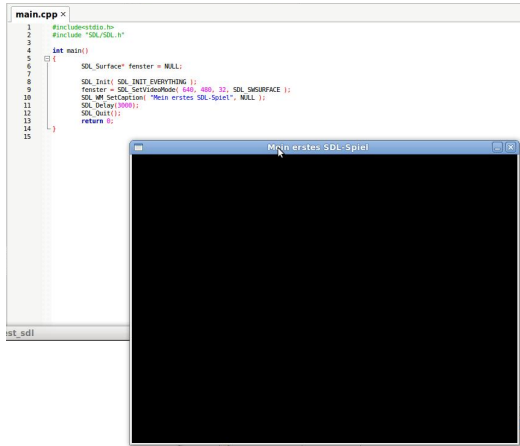
```
#include<stdio.h>
#include "SDL/SDL.h"

int main()
{
    SDL_Surface* fenster = NULL;

    SDL_Init( SDL_INIT_EVERYTHING );
    fenster = SDL_SetVideoMode( 640, 480, 32, SDL_SWSURFACE );
    SDL_WM_SetCaption( "Mein erstes SDL-Spiel", NULL );
    SDL_Delay(3000);
    SDL_Quit();
    return 0;
}
```

- `SDL_Delay(3000)` pausiert das Programm für 3 Sekunden

SDL Initialisierung



- Einleitung
- Entwicklungsumgebung vorbereiten
- SDL Initialisierung
- **Bild anzeigen**
- Weitere Bildformate
- Transparenter Hintergrund
- Bildteil laden
- Schrift
- Ereignisse
- Sound Ausgabe

SDL Bild laden

- Im ersten Schritt müssen wir ein Bild laden

Bild laden

```
SDL_Surface* bild = SDL_LoadBMP( "hallo.bmp" );
```

- Die Funktion gibt eine Referenz (Pointer) auf ein SDL Oberfläche zurück die wir in einer Variable vom Typ SDL_Surface speichern

SDL Bild laden

- Das geladene Bild optimieren
- Bild hat eine Farbtiefe von 24Bit und unsere Anwendung 32Bit -> Live-Umwandlung kostet viel Performance

Optimiertes Bild laden

```
SDL_Surface* opt_bild = SDL_DisplayFormat( bild );
```

- Die Funktion gibt eine Referenz (Pointer) auf die optimierte SDL Oberfläche zurück, die wir in einer Variable vom Typ SDL_Surface speichern

SDL Bild laden

- Damit wir das Bild anzeigen können müssen wir es in den Speicher laden und unserem Fenster zuordnen

Bild in Speicher laden

```
SDL_BlitSurface( opt_bild, NULL, fenster, NULL );
```

- Der erste Parameter ist die Quell und der dritte Parameter das Ziel-
Rest später

Refresh Fenster

```
SDL_Flip( fenster );
```

- Damit das Fenster mit dem gerade zugewiesenen Bild angezeigt wird müssen wir die Anzeige mit `SDL_Flip` erneuern

- Als letzten Schritt müssen wir den Speicher für unsere geladenen Bilder freigeben

Speicher freigeben

```
SDL_FreeSurface( bild );  
SDL_FreeSurface( opt_bild );
```

SDL Bild laden

Beispiel

```
#include<stdio.h>
#include "SDL/SDL.h"

int main()
{
    SDL_Surface* fenster = NULL;
    SDL_Surface* opt_bild = NULL;
    SDL_Surface* bild = NULL;

    SDL_Init( SDL_INIT_EVERYTHING );
    fenster = SDL_SetVideoMode( 640, 480, 32, SDL_SWSURFACE );
    SDL_WM_SetCaption( "Mein erstes SDL-Spiel", NULL );
    bild = SDL_LoadBMP( "hallo.bmp" );
    opt_bild = SDL_DisplayFormat( bild );
    SDL_BlitSurface( opt_bild, NULL, fenster, NULL );
    SDL_Flip( fenster );
    SDL_Delay(3000);
    SDL_FreeSurface( bild );
    SDL_FreeSurface( opt_bild );
    SDL_Quit();
    return 0;
}
```

Übung SDL 1

- ÜBUNG!!!!

- Einleitung
- Entwicklungsumgebung vorbereiten
- SDL Initialisierung
- Bild anzeigen
- **Weitere Bildformate**
- Transparenter Hintergrund
- Bildteil laden
- Schrift
- Ereignisse
- Sound Ausgabe

Weitere Bildformate

- Um andere Bildformate zu laden benötigen wir die Erweiterung SDL_Image

Image Header einbinden

```
#include "SDL/SDL_image.h"
```

- Dann müssen wir nur SDL_LoadBMP durch IMG_Load ersetzen

JPG Laden

```
SDL_Surface* bild = IMG_Load( "hallo.jpg" )
```

Weitere Bildformate

Beispiel

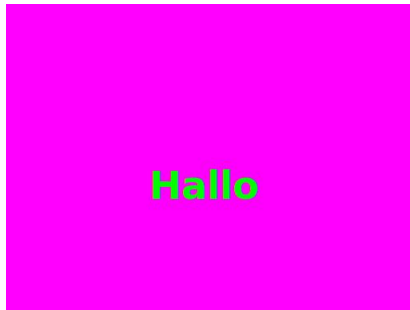
```
#include<stdio.h>
#include "SDL/SDL.h"
#include "SDL/SDL_image.h"

int main()
{
    SDL_Surface* fenster = NULL;
    SDL_Surface* opt_bild = NULL;
    SDL_Surface* bild = NULL;

    SDL_Init( SDL_INIT_EVERYTHING );
    fenster = SDL_SetVideoMode( 640, 480, 32, SDL_SWSURFACE );
    SDL_WM_SetCaption( "Mein erstes SDL-Spiel", NULL );
    bild = IMG_Load( "hallo.bmp" );
    opt_bild = SDL_DisplayFormat( bild );
    SDL_BlitSurface( opt_bild, NULL, fenster, NULL );
    SDL_Flip( fenster );
    SDL_Delay(3000);
    SDL_FreeSurface( bild );
    SDL_FreeSurface( opt_bild );
    SDL_Quit();
    return 0;
}
```

- Einleitung
- Entwicklungsumgebung vorbereiten
- SDL Initialisierung
- Bild anzeigen
- Weitere Bildformate
- **Transparenter Hintergrund**
- Bildteil laden
- Schrift
- Ereignisse
- Sound Ausgabe

- Aus dem folgenden Bild entfernen wir den Hintergrund



Transparenter Hintergrund

- An dieser Stelle erzeugen wir einen Farbschlüssel passend zur Farbtiefe des Fensters

Farbschlüssel erzeugen

```
Uint32 colorkey = SDL_MapRGB( opt_bild->format, 0xFF, 0, 0xFF );
```

- Den zurückgegebenen Farbschlüssel speichern wir in einer Variable vom Typ Uint32, 0xFF, 0, 0xFF ist unsere schöne Hintergrundfarbe
- Dann müssen wir nur noch den Farbschlüssel unserem Bild zuweisen

Farbe setzen

```
SDL_SetColorKey( opt_bild, SDL_SRCCOLORKEY, colorkey );
```

Transparenter Hintergrund

Beispiel

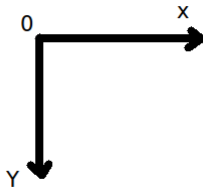
```
#include<stdio.h>
#include "SDL/SDL.h"
#include "SDL/SDL_image.h"

int main()
{
    SDL_Surface* fenster = NULL;
    SDL_Surface* opt_bild = NULL;
    SDL_Surface* bild = NULL;
    SDL_Init( SDL_INIT_EVERYTHING );
    fenster = SDL_SetVideoMode( 640, 480, 32, SDL_SWSURFACE );
    SDL_WM_SetCaption( "Mein erstes SDL-Spiel", NULL );
    bild = IMG_Load( "hallo.jpg" );
    opt_bild = SDL_DisplayFormat( bild );
    Uint32 colorkey = SDL_MapRGB( opt_bild->format, 0xFF, 0, 0xFF );
    SDL_SetColorKey( opt_bild, SDL_SRCCOLORKEY, colorkey );
    SDL_BlitSurface( opt_bild, NULL, fenster, NULL );
    SDL_Flip( fenster );

    SDL_FreeSurface( bild );
    SDL_FreeSurface( opt_bild );
    SDL_Delay(3000);
    SDL_Quit();
    return 0;
}
```

- Einleitung
- Entwicklungsumgebung vorbereiten
- SDL Initialisierung
- Bild anzeigen
- Weitere Bildformate
- Transparenter Hintergrund
- **Bildteil laden**
- Schrift
- Ereignisse
- Sound Ausgabe

- Bevor wir einen bestimmten Teil eines Bildes an einer vorgegebenen Stelle am Bildschirm anzeigen, schauen wir uns das Koordinatensystem in SDL an



Bildteil laden

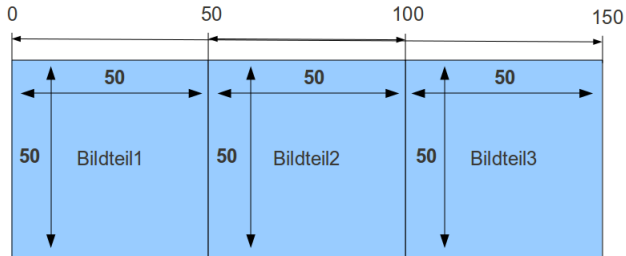
- Zum speichern der Koordinaten benutzen wir den Variablentyp `SDL_Rect`
- `SDL_Rect` besteht aus folgenden Elementen
 - Höhe = `h`
 - Breite = `w`
 - X-Koordinate = `x`
 - Y-Koordinate = `y`

Abmaße setzen

```
SDL_Rect bild_pos;  
SDL_Rect bild_teil[3];  
  
bild_pos.x = 10;  
bild_pos.y = 10;  
  
bild_teil[0].x = 0;  
bild_teil[0].y = 0;  
bild_teil[0].h = 50;  
bild_teil[0].w = 50;  
  
bild_teil[1].x = 50;  
bild_teil[1].y = 0;  
bild_teil[1].h = 50;  
bild_teil[1].w = 50;  
  
bild_teil[2].x = 100;  
bild_teil[2].y = 0;  
bild_teil[2].h = 50;  
bild_teil[2].w = 50;
```

Bildteil laden

- Wenn ihr jetzt sagt hääää, dann schaut euch das Bild an



- Unser schönes Testbild



- Um die Bildteile in den Speicher zu schieben benutzen wir wieder die `SDL_BlitSurface` und ersetzen die `NULLEN`

Bild in Speicher laden

```
SDL_BlitSurface( quelle, &bild_teil[0], ziel, &bild_pos );
```

- Der zweite Übergabewert gibt den Bildbereich an und der vierte Parameter die Position auf dem Bildschirm

Bildteil laden

Beispiel

```
int main()
{
    SDL_Surface* fenster = NULL;
    SDL_Surface* opt_bild = NULL;
    SDL_Surface* bild = NULL;

    SDL_Rect bild_pos;
    SDL_Rect bild_teil[3];

    bild_pos.x = 10;
    bild_pos.y = 10;

    bild_teil[0].x = 0;
    bild_teil[0].y = 0;
    bild_teil[0].h = 50;
    bild_teil[0].w = 50;

    bild_teil[1].x = 50;
    bild_teil[1].y = 0;
    bild_teil[1].h = 50;
    bild_teil[1].w = 50;

    bild_teil[2].x = 100;
    bild_teil[2].y = 0;
    bild_teil[2].h = 50;
    bild_teil[2].w = 50;
```

Bildteil laden

Beispiel

```
SDL_Init( SDL_INIT_EVERYTHING );
fenster = SDL_SetVideoMode( 640, 480, 32, SDL_SWSURFACE );
SDL_WM_SetCaption( "Mein erstes SDL-Spiel", NULL );
bild = IMG_Load( "bildteil.png" );
opt_bild = SDL_DisplayFormat( bild );
Uint32 colorkey = SDL_MapRGB( opt_bild->format, 0xFF, 0, 0xFF );
SDL_SetColorKey( opt_bild, SDL_SRCCOLORKEY, colorkey );
SDL_Blitsurface( opt_bild, &bild_teil[0], fenster, &bild_pos );
bild_pos.y = bild_pos.y + 55;
SDL_Blitsurface( opt_bild, &bild_teil[1], fenster, &bild_pos );
bild_pos.y = bild_pos.y + 55;
SDL_Blitsurface( opt_bild, &bild_teil[2], fenster, &bild_pos );
SDL_Flip( fenster );

SDL_FreeSurface( bild );
SDL_FreeSurface( opt_bild );
SDL_Delay(3000);
SDL_Quit();
return 0;
}
```

- In eine Schleife gepackt kann man schon eine Animation erzeugen

- Einleitung
- Entwicklungsumgebung vorbereiten
- SDL Initialisierung
- Bild anzeigen
- Weitere Bildformate
- Transparenter Hintergrund
- Bildteil laden
- **Schrift**
- Ereignisse
- Sound Ausgabe

Schrift

- Um Schrift anzuzeigen benötigen wir die TTF(True Type Font)-Lib

TTF Header

```
#include "SDL/SDL_ttf.h"
```

- Initialisiert wird die TTF mit:

TTF Initialisieren

```
TTF_Init();
```

- Beendet wird TTF mit

TTF Close

```
TTF_Quit();
```

Schrift

- Um Schrift anzeigen zu lassen brauchen wir drei neue Dinge
 - Erzeugen einer Schriftfarbe
 - Laden einer Schriftart
 - Funktion zum Text Rendern (Text umwandeln in Bild) und zum Erzeugen eines SDL_Surface

Erzeugen einer Schriftfarbe

```
SDL_Color textfarbe = { 128, 88, 77 };
```

- 128 = Rot-Farbwert, 88 = Grün-Farbwert, 77 = Blau-Farbwert - Rückgabe vom Typ SDL_Color

Laden einer Schriftart

```
TTF_Font *schriftart = TTF_OpenFont( "helvetidoodlebyedt.ttf", 50 );
```

- Erster Parameter ist die Schriftart und der zweite die Schriftgröße - Rückgabe ist ein Pointer auf die geladene Schriftart

Render Funktion

```
SDL_Surface *text = TTF_RenderText_Solid( schriftart, "Mein Text!", textfarbe );
```

- Erster Parameter Schriftart, zweiter der Ausgabe Text, dritte Textfarbe - Rückgabe ist ein Pointer auf vom bekannten Typ SDL_Surface

Schrift

Beispiel

```
#include<stdio.h>
#include "SDL/SDL.h"
#include "SDL/SDL_ttf.h"

int main()
{
    SDL_Surface* fenster = NULL;
    SDL_Surface* text = NULL;
    TTF_Font *schriftart = NULL;
    SDL_Color textfarbe = { 128, 88, 77 };

    SDL_Init( SDL_INIT_EVERYTHING );
    TTF_Init();

    fenster = SDL_SetVideoMode( 640, 480, 32, SDL_SWSURFACE );
    SDL_WM_SetCaption( "Mein erstes SDL-Spiel", NULL );
    schriftart = TTF_OpenFont( "helvetidoodlebyedt.ttf", 50 );
    text = TTF_RenderText_Solid( schriftart, "Mein Text!", textfarbe );

    SDL_BlitSurface( text, NULL, fenster, NULL );
    SDL_Flip( fenster );

    SDL_FreeSurface( text );
    TTF_CloseFont( schriftart );
    SDL_Delay(3000);
    TTF_Quit();
    SDL_Quit();
    return 0;
}
```



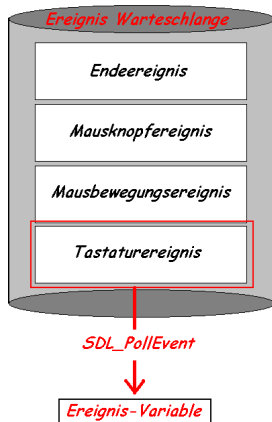
- Einleitung
- Entwicklungsumgebung vorbereiten
- SDL Initialisierung
- Bild anzeigen
- Weitere Bildformate
- Transparenter Hintergrund
- Bildteil laden
- Schrift
- **Ereignisse**
- Sound Ausgabe

- **Grundlagen**
- Programm beenden
- Tastaturereignis
- Mausereignis

- Alle Ereignisse landen in einem Art Stack, ähnlich einem Tellerstapel



- Die Funktion SDL_lädt das älteste Ereignis in eine Variable vom Typ SDL_Event



Grundlagen

- Variable zum Speichern eines Ereignis deklarieren

SDL Event deklarieren

```
SDL_Event ereignis;
```

- Älteste Ereignis laden

SDL Event laden

```
SDL_PollEvent( &ereignis);
```

- Wird in einer Schleife verwendet, um alle Ereignisse nacheinander zu bearbeiten

SDL Event in Schleife laden

```
while(SDL_PollEvent( &ereignis)){  
    //Ereignisbehandlung  
}
```

- Grundlagen
- **Programm beenden**
- Tastaturereignis
- Mausereignis

Beenden Ereignis

- Reagieren auf das Beendenereignis

Quit Ereignis

```
if(ereignis.type == SDL_QUIT){  
    ende = 1;  
}
```

- Fenster wird beim drücken des kleinen X geschlossen.

- Grundlagen
- Programm beenden
- **Tastaturereignis**
- Mausereignis

Tastaturereignis

- Reagieren auf Tastaturereignisse, Methode 1
- Alles ersten reagieren wir auf das Ereignis wenn die Taste gedrückt wird (es ist auch möglich erst beim Tasten los lassen zu reagieren)

Tastatur gedrückt?

```
if ( ereignis.type == SDL_KEYDOWN )  
{
```

- Als nächsten überprüfen wir welche Taste gedrückt wurden ist

Welche Taste?

```
switch ( ereignis.key.keysym.sym )  
{  
case SDLK_UP: // Pfeil nach oben  
    pos.y--;  
    break;  
case SDLK_DOWN: // Pfeil nach unten  
    pos.y++;  
    break;  
case SDLK_LEFT: // Pfeil nach links  
    pos.x--;  
    break;  
case SDLK_RIGHT: // Pfeil nach rechts  
    pos.x++;  
    break;  
case SDLK_1: //1  
    pos.x = 10;  
    pos.y = 10;
```

Tastaturereignis

Beispiel

```
#include<stdio.h>
#include "SDL/SDL.h"
#include "SDL/SDL_ttf.h"

int main()
{
    SDL_Surface* fenster = NULL;
    SDL_Surface* text = NULL;
    TTF_Font *schriftart = NULL;
    SDL_Color textfarbe = { 128, 88, 77 };
    SDL_Rect pos;
    int q = 0;
    SDL_Event ereignis;
    pos.x = 150;
    pos.y = 150;

    SDL_Init( SDL_INIT_EVERYTHING );
    TTF_Init();

    fenster = SDL_SetVideoMode( 640, 480, 32, SDL_SWSURFACE );
    SDL_WM_SetCaption( "Mein erstes SDL-Spiel", NULL );
    schriftart = TTF_OpenFont( "helvetidoodlebyedt.ttf", 50 );
    text = TTF_RenderText_Solid( schriftart, "Mein Text!", textfarbe );
```

Tastaturereignis

Beispiel

```
while (q != 1) {
    while (SDL_PollEvent( &ereignis)) {
        if ( ereignis.type == SDL_KEYDOWN ) {
            switch ( ereignis.key.keysym.sym ) {
                case SDLK_UP: // Pfeil nach oben
                    pos.y--;
                    break;
                case SDLK_DOWN: // Pfeil nach unten
                    pos.y++;
                    break;
                case SDLK_LEFT: // Pfeil nach links
                    pos.x--;
                    break;
                case SDLK_RIGHT: // Pfeil nach rechts
                    pos.x++;
                    break;
                case SDLK_1: //1
                    pos.x = 10;
                    pos.y= 10;
                    break;
            }
        }
        if (ereignis.type == SDL_QUIT)
            q = 1;
    }
    SDL_BlitterSurface( text, NULL, fenster, &pos );
    SDL_Flip( fenster );
}
```

Tastaturereignis

Beispiel

```
SDL_FreeSurface( text );  
TTF_CloseFont( schriftart );  
  
TTF_Quit();  
SDL_Quit();  
return 0;  
}
```

Tastaturereignis

- Reagieren auf Tastaturereignisse, Methode 2
- Bei Methode 2 benötigen wir ein extra Pointer, dieser Zeigt auf ein aktuellen Tastaturanschlag
- Diese Methode ist unabhängig von Events
- Diese Art von Tastenabfrage gibt es auch für Joysticks - `SDL_JoystickGetAxis()`

Tastenstatus laden

```
Uint8 *keystates = SDL_GetKeyState( NULL );
```

- Als nächsten überprüfen wir welche Taste gedrückt wurden ist

Welche Taste?

```
if( keystates[ SDLK_s ] )  
{  
    pos.x = 10;  
    pos.y= 10;  
}
```

Tastaturereignis

Beispiel

```
#include<stdio.h>
#include "SDL/SDL.h"
#include "SDL/SDL_ttf.h"

int main()
{
    SDL_Surface* fenster = NULL;
    SDL_Surface* text = NULL;
    TTF_Font *schriftart = NULL;
    SDL_Color textfarbe = { 128, 88, 77 };
    SDL_Rect pos;
    int q = 0;
    SDL_Event ereignis;
    pos.x = 150;
    pos.y = 150;

    SDL_Init( SDL_INIT_EVERYTHING );
    TTF_Init();

    fenster = SDL_SetVideoMode( 640, 480, 32, SDL_SWSURFACE );
    SDL_WM_SetCaption( "Mein erstes SDL-Spiel", NULL );
    schriftart = TTF_OpenFont( "helvetidoodlebyedt.ttf", 50 );
    text = TTF_RenderText_Solid( schriftart, "Mein Text!", textfarbe );
```

Tastaturereignis

Beispiel

```
while (q != 1) {
    while (SDL_PollEvent( &ereignis)) {
        if ( ereignis.type == SDL_KEYDOWN ) {
            switch ( ereignis.key.keysym.sym ) {
                case SDLK_UP: // Pfeil nach oben
                    pos.y--;
                    break;
                case SDLK_DOWN: // Pfeil nach unten
                    pos.y++;
                    break;
                case SDLK_LEFT: // Pfeil nach links
                    pos.x--;
                    break;
                case SDLK_RIGHT: // Pfeil nach rechts
                    pos.x++;
                    break;
            }
        }
        if (ereignis.type == SDL_QUIT)
            q = 1;
    }
    Uint8 *keystates = SDL_GetKeyState( NULL );
    if ( keystates[ SDLK_l ] ) {
        pos.x = 10;
        pos.y = 10;
    }
    SDL_BlitterSurface( text, NULL, fenster, &pos );
    SDL_Flip( fenster );
}
```

Tastaturereignis

Beispiel

```
SDL_FreeSurface( text );  
TTF_CloseFont( schriftart );  
  
TTF_Quit();  
SDL_Quit();  
return 0;  
}
```


- Grundlagen
- Programm beenden
- Tastaturereignis
- **Mausereignis**

Mausereignis

- Gleiches Vorgehen wie bei Tastaturereignissen
- Abfrage ob Maustaste gedrückt wurden ist

Maustaste gedrückt?

```
if ( ereignis.type == SDL_MOUSEBUTTONDOWN )  
{
```

- Welche Taste wurde gedrückt

Welche Maustaste?

```
    if ( ereignis.button.button == SDL_BUTTON_LEFT )  
    {  
        //Behandlung  
    }  
}
```

- Oder Reaktion auf Mausbewegung mit auslesen der aktuellen Position

Maus bewegt?

```
if( ereignis.type == SDL_MOUSEMOTION ) {  
    x = ereignis.button.x;  
    y = ereignis.button.y;  
}
```

Tastaturereignis

Beispiel

```
#include<stdio.h>
#include "SDL/SDL.h"
#include "SDL/SDL_ttf.h"

int main()
{
    SDL_Surface* fenster = NULL;
    SDL_Surface* text = NULL;
    TTF_Font *schriftart = NULL;
    SDL_Color textfarbe = { 128, 88, 77 };
    SDL_Rect pos;
    int q = 0;
    SDL_Event ereignis;
    pos.x = 150;
    pos.y = 150;

    SDL_Init( SDL_INIT_EVERYTHING );
    TTF_Init();

    fenster = SDL_SetVideoMode( 640, 480, 32, SDL_SWSURFACE );
    SDL_WM_SetCaption( "Mein erstes SDL-Spiel", NULL );
    schriftart = TTF_OpenFont( "helvetidoodlebyedt.ttf", 50 );
    text = TTF_RenderText_Solid( schriftart, "Mein Text!", textfarbe );
```

Tastaturereignis

Beispiel

```
while (q != 1) {
    while (SDL_PollEvent( &ereignis)) {
        if ( ereignis.type == SDL_MOUSEBUTTONDOWN ) {
            if ( ereignis.button.button == SDL_BUTTON_LEFT ) {
                pos.x = 0;
                pos.y = 0;
            }

        }

        if ( ereignis.type == SDL_MOUSEMOTION ) {
            pos.x = ereignis.button.x;
            pos.y = ereignis.button.y;
        }

    }

    SDL_BlitterSurface( text, NULL, fenster, &pos );
    SDL_Flip( fenster );
}

SDL_FreeSurface( text );
TTF_CloseFont( schriftart );

TTF_Quit();
SDL_Quit();
return 0;
}
```

- Einleitung
- Entwicklungsumgebung vorbereiten
- SDL Initialisierung
- Bild anzeigen
- Weitere Bildformate
- Transparenter Hintergrund
- Bildteil laden
- Schrift
- Ereignisse
- **Sound Ausgabe**

Sound Ausgabe

- Zum Sound laden nutzen wir die SDL Mixer Lib
- Um diese zu benutzen muss sie initialisiert werden

Musik laden

```
Mix_OpenAudio( 22050, MIX_DEFAULT_FORMAT, 2, 4096 );
```

- Parameter 1: Sound Frequenz, empfohlen 22050
- Parameter 2: Soundformat, empfohlen MIX_DEFAULT_FORMAT
- Parameter 3: Kanäle, für Stereo 2
- Parameter 4: Sample Rate, 4096
- SDL Mixer beenden

Musik laden

```
Mix_CloseAudio();
```

- Unterschieden wird zwischen zwei Musikarten
 - Musik - Läuft meist im Hintergrund als Hintergrundmusik
 - Soundeffekte werden nur bei bestimmten Ereignissen abgespielt

Musik laden

```
Mix_Music *musik = Mix_LoadMUS( "music.mp3" );
```

Effekt laden

```
Mix_Chunk *effekt = Mix_LoadWAV( "effect.wav" );
```

- Effekt abspielen

Effekt abspielen

```
Mix_PlayChannel( -1, effekt, 0 );
```

- Parameter 1: Mit -1 wird der nächste freie Kanal zum Abspielen genutzt
- Parameter 2: Pointer auf geladenen Effekt
- Parameter 3: Wie oft soll der Effekt wiederholt werden - 0 bedeutet einmal

Sound Ausgabe

- Musik abspielen

Musik abspielen

```
Mix_PlayMusic( music, -1 )
```

- Parameter 1: Pointer auf geladene Musik
- Parameter 2: Wie oft soll der Effekt wiederholt werden - -1 bedeutet solange bis gestoppt wird
- Musik pausieren

Musik pausieren

```
Mix_PauseMusic();
```

- Musik aus der Pause wieder abspielen

Musik abspielen

```
Mix_ResumeMusic();
```

Sound Ausgabe

- Prüfen ob Musik bereits abgespielt wird, das ist auch der Fall wenn die Musik schon einmal gelaufen ist und in Pause besetzt wurden ist

Musik abspielt?

```
int result = Mix_PlayingMusic()
```

- Gibt eine Integer zurück, wobei 0 = Spielt nicht und 1 = Spielt ist
- Prüfen ob Musik Pausiert wurden ist

Musik Pausieren

```
int result = Mix_PausedMusic()
```

- Pause = 1 und Nicht in der Pause = 0

- ① Einleitung
- ② Entwicklungsumgebung
- ③ Grundlagen C
- ④ Codingstyle
- ⑤ Große Übung 1
- ⑥ Grundlagen C++
- ⑦ Große Übung 2
- ⑧ Grundlagen SDL
- ⑨ **Große Übung 3**
- ⑩ Offene Fragen

Große Übung 3

- Wir bauen dieses Spiel!!!!
- Tipps zum setzen eines einzelnen Pixels
- Die Pixel einer Obefläche läd man wie folgt

Pixel

```
Uint32 *pixels = (Uint32 *)bild->pixels;  
//Unit 32 passt zur Farbtiefe
```

- Pixel liegen in einem eindimensionalen Array, das heist die Pixel sind nacheinander angeordnet, die Zeilenumbrüche müssen berechnet werden

- ① Einleitung
- ② Entwicklungsumgebung
- ③ Grundlagen C
- ④ Codingstyle
- ⑤ Große Übung 1
- ⑥ Grundlagen C++
- ⑦ Große Übung 2
- ⑧ Grundlagen SDL
- ⑨ Große Übung 3
- ⑩ **Offene Fragen**

Fragen

- FRAGEN?????