Hochschule Trier, Umwelt-Campus Birkenfeld

PROGRAMMIERUNG 2
WS 2020/2021

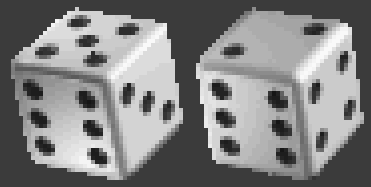
Prüfungsvorleistung PROGRA 2

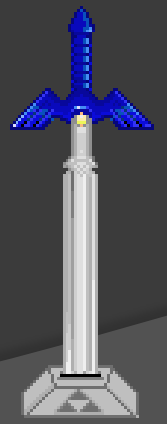
Dr. Markus Schwinn
Hochschule Trier, Umwelt-Campus Birkenfeld

Prüfungsvorleistung

- Befolgen Sie die Aufgabenstellungen auf den nächsten Folien
- Implementieren Sie ein lauffähiges C-Programm
- Sie können das Programm in einem 2er-Team erstellen
- Reichen Sie Ihre Lösung bis zum 18.11.2020 per Mail ein
- Geben Sie bei der Abgabe an, ob eine weitere Person an der Bearbeitung mitgewirkt hat
- Dieses Programm wird Teil der weiteren Übungen in Programmierung 2 sein.

Aufbau einer kleinen Spielewelt





Dungeon


LAIR OF THE BEAST
Entrance
Worship Chamber
N
Lair
Trolls
Throne Room
FANTASYMAPMAKER.COM

Aufgabe 1 – Der Würfel

Zufall in C

In C gibt es eine Funktion **rand()**, die eine Pseudo-Zufallszahl zurückliefert. Die Zahl liegt zwischen 0 und einer maximalen Zahl RAND_MAX.

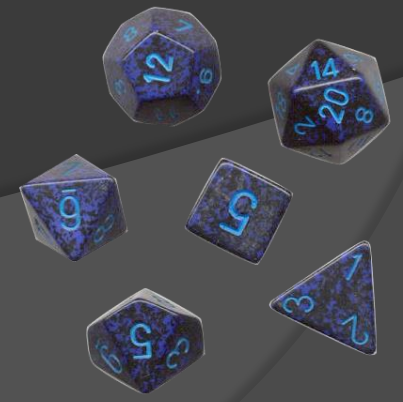
Das Eingrenzen der Zufallszahl kann mit Hilfe des Modulo-Operatores durchgeführt werden.

Bevor die Funktion **rand()** aufgerufen wird, sollte der Zufallsgenerator initialisiert werden, damit die Zahlen auch tatsächlich zufällig sind.

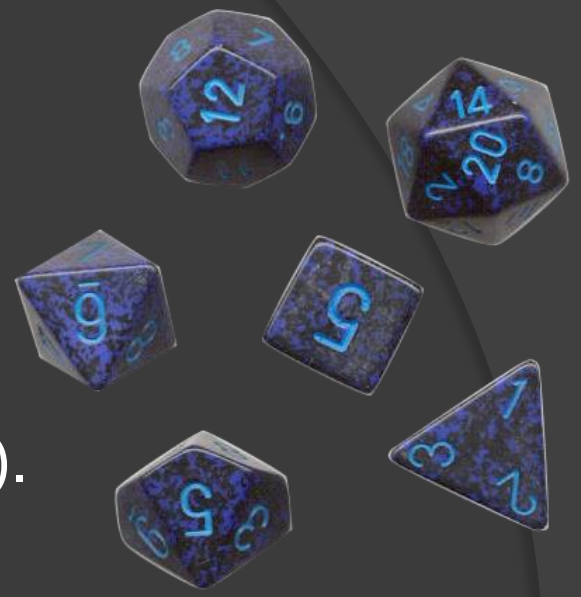
Dazu wird die Funktion **srand()** verwendet. Diese wird mit einem Zeitstempel aufgerufen.

srand(time(null));

Die Funktion **time()** ist in der Bibliothek **time.h** zu finden.



Würfel



Schreiben Sie eine Funktion **rollDice.**

Die Funktion bekommt als Argument übergeben um welche Art Würfel es sich handelt (z.B. 6 für einen 6-seitigen Würfel). Der Rückgabewert der Funktion ist das Ergebnis des Würfelwurfs.

Testen Sie anschließend ihre Funktion, indem Sie in der Main-Funktion jeweils einen Wurf mit einem W6, W3, W20 und W8 Würfel auf der Konsole ausgeben.

Aufgabe 2 – Gegenstände

Attribute

- name 6(max. 25 Zeichen)
- position:
  - Head
  - Neck
  - Armor
  - RightHand
  - LeftHand
  - Foot
- weight (z.B. 17.5)
- value (z.B. 75)
- lp_change (z.B. -3, 0, 8)

Strukturen

- Implementieren Sie zunächst eine Aufzählung E**ItemPosition** für die Tragepositionen von Gegenständen.

- Implementieren Sie dann eine Struktur **Item**, die die zuvor genannten Attribute eines Gegenstandes enthält.

Ausgabe von Gegenständen

- Schreiben Sie eine Funktion **printItem.** Diese bekommt einen Gegenstand übergeben und gibt auf der Konsole alle Informationen über den Gegenstand aus:

```
****** Gegenstands-Info ******
Name: Holzschwert
Position: 4
Gewicht: 2.50
Wert: 30
Veraenderung LP: 0
********************************
```

Erstellen von Gegenständen

- Erstellen Sie in der Main-Funktion, den unten angezeigten Gegenstand und geben Sie diesen mit Hilfe ihrer Funktion **printItem** auf der Konsole aus.

```
****** Gegenstands-Info ******
Name: Holzschwert
Position: 4
Gewicht: 2.50
Wert: 30
Veraenderung LP: 0
**********************************
```

Erstellen von Gegenständen

- Schreiben Sie abschließend eine Funktion **createItem.** Die Funktion liefert einen Gegenstand zurück. Übergeben bekommt die Funktion alle Attribute, aus denen der Gegenstand aufgebaut ist.

  z.B. **createItem(„Eisenschild“, hand, 5.0, 20, 0);**

Der besondere Gegenstand NICHTS

- Erstellen Sie außerhalb der Main-Funktion einen neuen Gegenstand und nennen Sie diesen NOTHING.

- Der Gegenstand hat den Namen „Leer“, die Position 0, wiegt nichts und hat keinen Wert

Aufgabe 3 – Der Held

Attribute

- name (max. 25 Zeichen)
- lifepoints
- itemSlots
  - headSlot
  - neckSlot
  - armorSlot
  - rightHandSlot
  - leftHandSlot
  - feetSlot
- inventory (max. 10 Gegenstände)

Strukturen und Funktionen

- Implementieren Sie eine Struktur **Hero**
- Implementieren Sie dann eine Funktion **printHero** und **createHero** (ohne Items und Inventar)
- Verwenden Sie NOTHING als Gegenstand in Slots und im Inventar
- Erzeugen Sie einen Helden in der Main-Funktion und geben Sie anschließend seine Informationen auf der Konsole aus.

Strukturen und Funktionen

- Die Ausgabe auf der Konsole sollte wie folgt aussehen:

```
****** Helden-Info ******
Name: Zelda
Lebenspunkte: 10
Slots:
          Kopf: Leer
          Hals: Leer
          Ruestung: Leer
          Linke Hand: Leer
          Rechte Hand: Leer
          Fuesse: Leer
Inventar:
          1. Item: Leer
          2. Item: Leer
          3. Item: Leer
          4. Item: Leer
          5. Item: Leer
          6. Item: Leer
          7. Item: Leer
          8. Item: Leer
          9. Item: Leer
          10. Item: Leer
**********************************
```

Gegenstände anlegen

- Schreiben Sie eine Funktion **wearItem**
- Die Funktion bekommt einen Hero-Pointer und das Item übergeben, das angezogen werden soll
- Der Gegenstand wird an der Position angelegt, für die er vorgesehen ist. Aber nur wenn diese noch frei ist.

Gegenstände anlegen

- Legen Sie jetzt ihrem Helden mit Hilfe der Funktion **wearItem** das Holzschwert aus Aufgabe 1 an.

- Erzeugen Sie ein zusätzliches Eisenschwert und versuchen Sie auch dieses anzulegen, was nicht funktionieren sollte.

Gegenstände ins Inventar legen

- Implementieren Sie eine Funktion **addToInventory**
- Die Funktion bekommt einen Hero-Pointer und ein Item übergeben
- Das Item wird an den nächsten freien Platz im Inventory abgelegt
- Sollte kein Platz frei sein, kommt als Ergebnis -1 zurück. Ansonsten 1.
- Legen Sie das Eisenschwert aus der vorherigen Aufgabe ins Inventar.

Gegenstände anlegen (verbessert)

- Ändern Sie die Funktion **wearItem** folgend ab:

- Wenn ein Gegenstand an einen bereits belegten Slot angezogen werden soll, wird der alte Gegenstand ins Inventar gelegt, wenn dort noch Platz ist.

- Sollte dort kein Platz sein. Ändert sich nichts.

Gegenstände anlegen (verbessert)

- Testen Sie, ob das Anlegen des Eisenschwertes nun funktioniert.

- Erzeugen Sie in der Main-Funktion einen Heiltrank, der die Lebenspunkte um +5 erhöht und legen Sie diesen in das Inventar.

Gegenstände anlegen (verbessert)

- Wenn bisher alles bei ihnen funktioniert, sollte die Ausgabe des Helden so aussehen:

```
***** Helden-Info *****
Name: Zelda
Lebenspunkte: 10
Slots:
        Kopf: Leer
        Hals: Leer
        Ruestung: Leer
        Linke Hand: Eisenschwert
        Rechte Hand: Leer
        Fuesse: Leer
Inventar:
        1. Item: Holzschwert
        2. Item: Heiltrank
        3. Item: Leer
        4. Item: Leer
        5. Item: Leer
        6. Item: Leer
        7. Item: Leer
        8. Item: Leer
        9. Item: Leer
        10. Item: Leer
*****************************
```

Aufgabe 4 – Die Gegner

Attribute

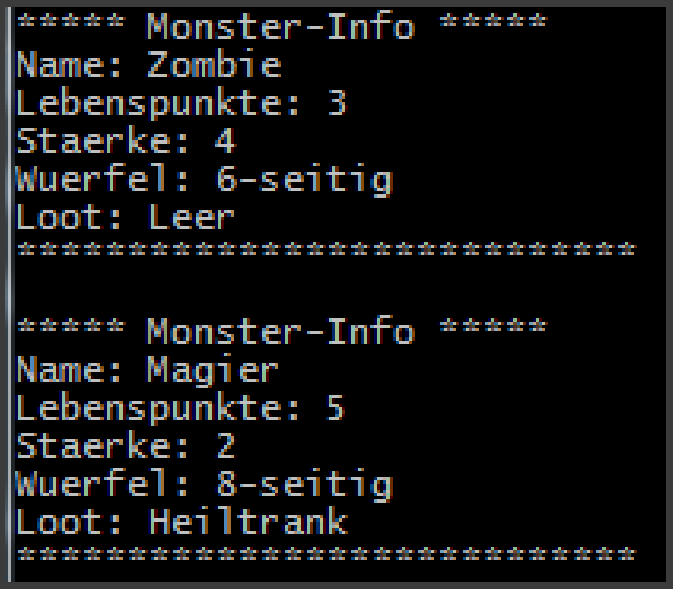
- Name (max. 20 Zeichen)
- Lebenspunkte (z.B. 10)
- Stärke (z.B. 5)
- Seitenanzahl des Würfels (z.B. 8), der beim Angriff geworfen wird
- Item (welches es fallen lässt)

Strukturen und Funktionen

- Implementieren Sie eine Struktur **Monster**
- Schreiben Sie eine Funktion **createMonster.**
- Schreibe Sie eine Funktion **printMonster**
- Schreiben Sie eine Funktion **monsterAddLoot**, die einem Monster einen Gegenstand zuweist

Strukturen und Funktionen

- Erzeugen Sie einen Zombie, der keinen Gegenstand fallen lässt
- Erzeugen Sie einen Magier, der einen Heiltrank fallen lässt. Weisen Sie den Heiltrank mit der Funktion **monsterAddLoot** zu.
- Testen Sie die **printMonster-Funktion** mit beiden Monstern.



Aufgabe 5 – Die Räume

Attribute

- Name (max. 25 Zeichen)
- Nachbarräume
  (Nächste Folie erst lesen)
- Auswirkung auf Lebenspunkt
- Monster (Monster in dem Raum oder NULL wenn keines vorhanden)
- Schatz (ein Item, falls ein Schatz in dem Raum vorhanden ist, ansonsten LEER)

Nachbarräume

- Erstellen Sie eine ENumeration mit den Werten NORTH, SOUTH, EAST und WEST
- Erstellen Sie eine globale Variable vom Typ **Room** und nennen diese WALL
- Ein Raum kann maximal 4 Nachbarräume enthalten (einen in jede Richtung oder eine WALL)

Struktur und Funktionen

- Implementieren Sie eine Struktur **Room**.
- Erweitern Sie die Struktur des Helden um eine weitere Komponente, in der der Raum hinterlegt wird, in dem sich der Held befindet
- Schreiben Sie eine Funktion **createRoom** und **printRoom**

Funktionen

- Schreiben Sie eine Funktion **setRoomInDirection**. Diese Funktion bekommt zwei Räume und eine Richtung übergeben. Der zweite Raum wird bei dem ersten Raum in dieser Richtung als Nachbar gesetzt. Beim zweiten Raum wird der andere in der gegengesetzten Richtung als Nachbar gesetzt.

Funktionen

- Implementieren Sie eine Funktion **getRoomInDirection,** die überprüft, ob es einen Raum in der angegeben Richtung gibt und einen Pointer darauf liefert, falls vorhanden

- Implementieren Sie eine Funktion **walkInDirection**. Die Funktion bewegt den Helden in den Raum der angegebenen Richtung, falls dort einer ist.

Funktionen

- Erstellen Sie in der Main-Funktion zwei Räume „Eingang“ und „Thronsaal“.
- Setzen Sie den Thronsaal in den Westen vom Eingang als Nachbarraum.
- Geben Sie anschließend beide Räume mit der **printRoom-Funktion** aus.

```
***** Raum-Info *****
Name: Eingang
Nachbarraeume:
          Nord: Wand
          Ost: Wand
          Sued: Wand
          West: Thronsaal
Monster: Zombie
Schatz: Heiltrank
*******************************

***** Raum-Info *****
Name: Thronsaal
Nachbarraeume:
          Nord: Wand
          Ost: Eingang
          Sued: Wand
          West: Wand
Monster: Magier
Schatz: Leer
*******************************
```

Funktionen

- Schreiben Sie eine Funktion **setRoom**, mit der der Held direkt in einen Raum gesetzt werden kann.
- Setzen Sie ihren Helden in den Eingang und lassen Sie ihn anschließend mit der **walkInDirection-Funktion** in den Thronsaal gehen

```
***** Helden-Info *****
Name: Zelda
Lebenspunkte: 10
Raum: Thronsaal
Slots:
          Kopf: Leer
          Hals: Leer
          Ruestung: Leer
          Linke Hand: Eisenschwert
          Rechte Hand: Leer
          Fuesse: Leer
Inventar:
          1. Item: Holzschwert
          2. Item: Heiltrank
          3. Item: Leer
          4. Item: Leer
          5. Item: Leer
          6. Item: Leer
          7. Item: Leer
          8. Item: Leer
          9. Item: Leer
          10. Item: Leer
*****************************
```

Zusammenfassung

Ihr fertiges Programm enthält am Ende folgende Elemente:

- 2 Aufzählungen (EItemPosition und EDirection)
- 4 Strukturen (Item, Hero, Monster, Room)
- 2 globale Variablen (Item NOTHING und Room WALL)
- 16 Funktionen
  - rollDice
  - printItem
  - createItem
  - createHero
  - printHero
  - wearItem
  - addToInventory
  - createMonster
  - printMoster
  - monsterAddLoot
  - createRoom
  - printRoom
  - setRoomInDirection
  - getRoomInDirection
  - walkInDirection
  - setRoom