Cheat-Sheet C++

1. Includes (Bibliotheken/Header-Dateien)

|  |  |
| --- | --- |
| Code | Kurzbeschreibung |
| #include <iostream> | Ein- und Ausgabe |
| #include <math.h> | Mathematik Sonderfunktionen |
| #include <string> | Zeichenketten Sonderfunktionen |
| #include <cstdlib> | Speicher, Bearbeitung und Zufallszahl Generator |
| #include <iomanip> | Manipulatoren hinzufügen |
| #include <ctime> | Zeit Funktionen einbinden |
| #include <fstream> | Zur Datei Erstellung |

1. Datentypen

|  |  |  |
| --- | --- | --- |
| Art | Code | Kurzbeschreibung (32 Bit System) |
| ganzzahlig | word | 2 Byte Zahlenwert |
|  | short | 2 Byte Zahlenwert |
|  | int | 4 Byte Zahlenwert |
|  | long | 4 Byte Zahlenwert |
| fließzahl | float | 4 Byte Zahlenwert mit Komma |
|  | double | 8 Byte Zahlenwert mit Komma |
| zeichen | char | 1 Byte |
|  | string | Pro Zeichen 1 Byte |
| vorzeichen | unsigned | Bei Zahlen Datentypen ohne Vorzeichen (nur positive Werte) |
|  | signed | Bei Zahlen Datentypen mit Vorzeichen (positive und negative Werte) |
| konstante | const | Bestimmung von Konstanten Werten z.B.: const int |
|  | #define | Bestimmung von Konstanten Werten z.B.: #define PI 3.14; |
| sonder | Enum | Eigener Datentyp erstellen z.B.: enum Ampel {rot, gelb, gruen}; |
|  | bool | 1 Byte Wert: True oder False Bestimmung |
|  | void | Leere Angabe |

1. Variablen und co.

|  |  |  |
| --- | --- | --- |
| Art | Code | Kurzbeschreibung |
| Variablen | Int variable; | Variable |
|  | Int array[ ]; | Feld Variable |
|  |  |  |
| Ein-/Ausgabe | cout<<; | Commando Ausgabe |
|  | cin>>; | Commando Eingabe (Nach Space und Enter ist die Eingabe bestätigt) |
|  | fflush(stdin); | Tastatur-Puffer Speicher löschen |
|  | gets( ); | String einlesen |
|  |  |  |

1. Verzweigungen

|  |  |
| --- | --- |
| Code | Struktogramm |
| If (Bedienung == true) {  …  } | C:\Users\Weazly\AppData\Local\Microsoft\Windows\INetCache\Content.MSO\EA511C4C.tmp |
| If (Bedienung == true) {  …  }  else  {  …  } | C:\Users\Weazly\AppData\Local\Microsoft\Windows\INetCache\Content.MSO\DA70643A.tmp |
| If (Bedienung == true) {  …  }  else If (Bedienung2 == true)  {  …  }  else  {  …  } | C:\Users\Weazly\AppData\Local\Microsoft\Windows\INetCache\Content.MSO\BB006858.tmp |
| Switch (Variable)  {  case Wert1:  …  Break;  case Wert2:  …  Break;  case Wert …:  …  Break;  default:  …  Break;  } | C:\Users\Weazly\AppData\Local\Microsoft\Windows\INetCache\Content.MSO\541DE826.tmp |

1. Schleifen

|  |  |
| --- | --- |
| Code | Struktogramm |
| while (Bedienung == true)  {  …  } |  |
| for (variable; bedienung == true; manipulation)  {  …  } |  |
| do  {  …  }  while (Bedienung == true) |  |

1. Manipulatoren

|  |  |  |
| --- | --- | --- |
| Art | Code | Kurzbeschreibung |
| Feld Definition | Setw(Wert); | Feldbreite festlegen |
|  | setfill('.'); | Feld mit '.' füllen und Manipulator einschalten |
|  | setfill(' '); | Manipulator setfill ausschalten |
|  |  |  |
| Zeichen | <<left<<; | Linksbündig schreiben |
| Position | <<right<<; | Rechtsbündig schreiben |
|  | <<internal<<; | Vorzeichen linksbündig und Zahl rechtsbündig schreiben |
|  | <<showpos<<; | positives Vorzeichen wird explizit ausgegeben |
|  | <<noshowpos<<; | wieder das showpos aufgehoben |
|  | <<nouppercase<<; | Folgende Ausgabe wird klein geschrieben |
|  | <<uppercase<<; | Folgende Ausgabe wird groß geschrieben |
|  |  |  |
| Wert- | <<dec<<; | Folgender Wert als Dezimal darstellen |
| Darstellung | <<hex<<; | Folgender Wert als Hexadezimal darstellen |
|  | <<oct<<; | Folgender Wert als Octal darstellen |
|  |  |  |
| Fließzahlen | <<setprecision(x); | Folgender Wert hat x, Nachkomma Stellen |
|  | <<scientfific; | Folgender Wert wird exponentiell dargestellt |

1. Funktionen von Bibliotheken

|  |  |  |
| --- | --- | --- |
| Art | Code | Kurzbeschreibung |
|  |  |  |
| Zeit | srand( ); | Zeitliche Zufallszahl |
|  | time(NULL); | Wiedergabe von aktueller Zeit |
|  | rand( ); | Einfache Zufallszahl generieren |
|  |  |  |
| Math | pow(a, b); | a Basis, b Exponent | pow liefert das Ergebnis |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

1. Sprunganweisungen

|  |  |
| --- | --- |
| Code | Kurzbeschreibung |
| Break | Unterbricht die Schleife |
| Continue | Aktuellen Schleifendurchlauf beenden. Wenn die Schleife erneut aufgerufen wird, wird die Schleife fortgesetzt |
| Return | Rücksprung von Funktion |

1. Funktionen

|  |  |
| --- | --- |
| Code | Kurzbeschreibung |
| void Funktionsname\_Prototyp(void); | Funktionsprototypen werden über main () geschrieben.  Funktion: ohne Rückgabe | ohne Übergabe |
| int Funktionsname\_Prototyp(void); | Funktionsprototypen werden über main () geschrieben.  Funktion: mit Rückgabe „int“ | ohne Übergabe |
| void Funktionsname\_Prototyp(int var); | Funktionsprototypen werden über main () geschrieben.  Funktion: ohne Rückgabe | mit Übergabe „int var“ |
|  |  |
| void Funktionsname (void)  {  …  } | Unterfunktionen werden außerhalb von main () geschrieben. Funktion: ohne Rückgabe | ohne Übergabe |
| int Funktionsname (void)  {  …  } | Unterfunktionen werden außerhalb von main () geschrieben. Funktion: mit Rückgabe „int“ | ohne Übergabe |
| void Funktionsname (int var)  {  …  } | Unterfunktionen werden außerhalb von main () geschrieben. Funktion: ohne Rückgabe | mit Übergabe „int var“ |
|  |  |
| #define TITEL „Titel“ | Makro: der String „Titel“ |
|  |  |
|  |  |
|  |  |

1. Strukturen

|  |  |
| --- | --- |
| Code | Kurzbeschreibung |
| struct Strukturname  {  int var;  double value;  char charakter;  }; | Eine Struktur wir über main( ) erstellt. Dieser beinhaltet verschiedene definierte Variablen unter einem zusammenhängenden Verhältnis. |
| Strukturname Variablenname; | Neue Variablen erstellen, welche mehrere Variablen beinhaltet von der Struktur.  Kann global und lokal erstellt werden |
| Variablenname.var = 5; | Struktur-Variable mit dem Wert 5 zuweisen. |

1. Operatoren

|  |  |  |
| --- | --- | --- |
| Art | Code | Kurzbeschreibung |
|  |  |  |
| Arithmetisch | + | Addieren |
|  | - | Subtrahieren |
|  | \* | Multiplizieren |
|  | / | Dividieren |
|  | % | Modulo = Restangabe von Dividieren |
|  | += | Addieren und gleich zuweisen |
|  | -= | Subtrahieren und gleich zuweisen |
|  | \*= | Multiplizieren und gleich zuweisen |
|  | /= | Dividieren und gleich zuweisen |
|  | %= | Modulo = Restangabe von Dividieren und gleich zuweisen |
|  | b = a ++ | Erst Inkrementieren (+1 rechnen), dann zuweisen |
|  | b = a -- | Erst Dekrementieren (-1 rechnen), dann zuweisen |
|  | b = ++ a | Erst zuweisen, dann Inkrementieren (+1 rechnen) |
|  | b = -- a | Erst zuweisen, dann Dekrementieren (-1 rechnen) |
|  |  |  |
| Logisch | && | Prüfe ob, Wert 1 und Wert 2 gleich sind |
|  | || | Prüfe ob, Wert 1 und Wert 2 oder sind |
|  | ! | Aussage Nicht-igen |
|  |  |  |
| Rationale | == | Ist gleich |
|  | != | Ist nicht gleich |
|  | > | Größer |
|  | < | Kleiner |
|  | >= | Größer gleich |
|  | <= | Kleiner gleich |
|  |  |  |
| Bitgesteuert | & | Ver-Und-e Wert 1 mit Wert 2: 0001 & 1001 -> 0001 |
|  | | | Ver-Oder-e Wert 1 mit Wert 2: 0001 | 1001 -> 1001 |
|  | ^ | Ver-Exklusiv-Oder-e Wert 1 mit Wert 2: 0001 ^ 1001 -> 1000 |
|  | ~ | Kippe jedes Bit um: 0001 -> 1110 |
|  | << | Schiebe die Bits um x-Stellen nach links z.B.: 0011 << 2 |
|  | >> | Schiebe die Bits um x-Stellen nach rechts z.B.: 0011 >> 2 |