FSST3_Weitere Themen in C – Cheat Sheet (Code-Schemata)

1. Funktionen in C

3 Schritte

- Deklaration (Funktionsprototyp)
- Definition (Funktionskörper)
- Aufruf (Call)

Deklaration (Schema)

```c
// Funktionsprototyp
rueckgabetyp function_name(parameter_typ1 parameter1,
                           parameter_typ2 parameter2,
                           /* ... */);
```

Definition (Schema)

```c
// Funktionsdefinition
rueckgabetyp function_name(parameter_typ1 parameter1,
                           parameter_typ2 parameter2,
                           /* ... */)
{
    /* Anweisungen */
    /* optional: Rueckgabewert */
    return rueckgabewert;  /* entf llt bei rueckgabetyp void */
}
```

Aufruf (Schema)

```c
// Funktionsaufruf
variable_mit_rueckgabewert = function_name(argument1, argument2, /* ... */);

/* oder bei rueckgabetyp void */
function_name(argument1, argument2, /* ... */);
```

2. Pointer (Zeiger)

Definition

- Pointer speichern **Speicheradressen** von Objekten (Variablen, Arrays, Funktionen, ...).
- Zugriff auf den Inhalt erfolgt über **Dereferenzierung**.

Deklaration und Initialisierung (Schema)

```
// Deklaration einer normalen Variable
wert_typ variable_name;

/* Deklaration eines Pointers auf diesen Typ */
wert_typ *pointer_name;

/* Initialisierung: Pointer zeigt auf die Adresse der Variable */
pointer_name = &variable_name;

/* Dereferenzierung: Zugriff auf den Wert  ber  Pointer */
wert_typ kopie_des_werts = *pointer_name;
```

Arten von Pointern (Schema)

```
/* Pointer auf einfachen Datentyp */
wert_typ *ptr_wert;

/* Pointer auf erstes Element eines Arrays */
element_typ *ptr_array = array_name;

/* Funktionszeiger (auf Funktion mit bestimmter Signatur) */
rueckgabetyp (*ptr_funktion)(parameter_typ1, parameter_typ2);

/* Doppelpointer (zeigt auf einen Pointer) */
wert_typ **ptr_ptr;

/* NULL-Pointer (zeigt auf nichts G ltiges) */
wert_typ *ptr_null = NULL;
```

Pointer-Arithmetik (Schema)

```
/* Zeiger auf Elementtyp */
element_typ *ptr;

/* Inkrement: zeigt auf das naechste Element dieses Typs */
ptr++;

/* Dekrement: zeigt auf das vorherige Element */
ptr--;

/* Offset: zeigt auf das Element mit Abstand n */
element_typ *ptr_offset = ptr + n;
```

3. Call by Value vs. Call by Reference

Call by Value (Schema)

```
/* Deklaration/Definition: Parameter als Wert (Kopie) */
void function_value(wert_typ parameter)
{
    /* Aenderungen an 'parameter' betreffen nur die lokale Kopie */
}

```

```c
/* Aufruf */
wert_typ variable;
function_value(variable);  /* Kopie von 'variable' wird uebergeben */
```

Call by Reference (Schema über Pointer)

```c
/* Deklaration/Definition: Parameter als Pointer (Referenz) */
void function_reference(wert_typ *parameter)
{
    /* Aenderungen ueber *parameter wirken auf die urspruengliche Variable */
}

/* Aufruf */
wert_typ variable;
function_reference(&variable);  /* Adresse von 'variable' wird uebergeben */
```

4. File I/O in C

Grundprinzip

- Arbeiten mit Dateien ueber einen **Dateizeiger** vom Typ `FILE *`.
- Typische Schritte: Datei oeffnen → lesen/schreiben → Datei schliessen.

Dateizeiger und Datei oeffnen (Schema)

```c
/* Deklaration eines Dateizeigers */
FILE *file_pointer;

/* Datei zum Lesen oeffnen ("r") */
file_pointer = fopen("dateiname.txt", "r");

/* Datei zum Schreiben oeffnen ("w") - bestehende Inhalte werden ggf.
   ueberschrieben */
file_pointer = fopen("dateiname.txt", "w");

/* Datei zum Anhaengen oeffnen ("a") */
file_pointer = fopen("dateiname.txt", "a");

/* Fehlerpruefung */
if (file_pointer == NULL) {
    /* Fehlerbehandlung, z.B. Meldung und Abbruch */
}
```

Zeilenweises Lesen/Schreiben (Schema)

```c
/* Zeilenpuffer */
char zeile[PUFFERGROESSE];

/* Zeile lesen (bis Zeilenende oder Puffer voll) */
if (fgets(zeile, PUFFERGROESSE, file_pointer) != NULL) {
    /* gelesene Zeile in 'zeile' verarbeiten */
}

/* Zeile schreiben */
fputs(zeile, file_pointer);
```

Zeichenweises Lesen/Schreiben (Schema)

```
/* Zeichen lesen */
int ch = fgetc(file_pointer);
if (ch != EOF) {
    /* gelesenes Zeichen in ch verarbeiten */
}

/* Zeichen schreiben */
int zeichen = 'X';   /* Beispiel-Char-Konstante */
fputc(zeichen, file_pointer);
```

Datei schliessen (Schema)

```
/* Datei schliessen, wenn alle Operationen abgeschlossen sind */
fclose(file_pointer);
```