Pointer und Arrays and Malloc (3)

- malloc, calloc, free and realloc
- String Handling mit Pointer
- Handling Array of Pointers
- Pointers und Array Notation

Dynamische Speicher Allokation

- Erlaubt zur Laufzeit zusätzliches Memory anzufordern
- Dafür sind Pointer notwendig
- Die meisten Programme benutzen dynamische Memory
  z.B. Email Client
- Dynamisches Memory wird auf auf dem Heap gespeichert
  während lokale Variablen und Parameter auf dem Stack gespeichert werden
- Beim Beenden wird der Speicher auf dem Heap gelöscht

malloc, free, calloc and realloc

100 Byte allozieren:

`stdlib.h`

```
int *pNumber = (int*)malloc(100);
```

oder besser

```
int *pNumber = (int*)malloc(25*sizeof(int));
```

der cast ist nicht zwingend, wird von Compiler hinzugefügt

Speicher prüfen

```
int *pNumber = (int*)malloc(25*sizeof(int));
if (!pNumber){
  // memory allocation failure…
}
```

malloc, free, calloc and realloc

Speicher freigeben:

```
free(pNumber);
pNumber = NULL;
```

Schreibe ein Programm, welches eine unbestimmte Anzahl von Primzahlen berechnet:

1. 2, 3 und 5 sind die ersten drei Primzahlen
2. Primzahlen sind ungerade
3. Berechne die nächste Primzahl indem du die letzte Primzahl mit 2 addierst und mit allen bereits gefundenen Primzahlen testest .
4. Implementiere das Programm mit Pointers und dynamischen Memory
    1. Speichere die ersten drei bekannten Primzahlen gerade zu Beginn ab.
    2. Finde die weiteren Primzahlen
    3. Gib die gefunden Primzahlen aus.

malloc, free, calloc and realloc

```
int *pNumber = (int*)calloc(75, sizeof(int));
```

Die `calloc` Funktion benötigt zwei Parameter:

1. Anzahl
2. Grösse einer Speichereinheit `size_t`

Die `calloc` Funktion initialisiert alle Bytes mit 0

```
int *pPrimes =
     calloc(25, sizeof(unsigned long long));


if (pPrimes == NULL){
  printf("Not enough memory.");
}
```

malloc, free, calloc and realloc

Speicher erweitern (verkleinern oder vergrössern)

- Bei einer Vergrösserung hat der Inhalt des neuen Speicherinhalt zufällige Werte

```
int *pPrimes = calloc(25, sizeof(unsigned long long));
if (pPrimes == NULL) {
  printf(“Not enough memory.”);
}

...
...
pTemp = realloc(pPrimes, 100*sizeof(unsigned long long));
if (pTemp == NULL) {
  printf(“No memory reallocation.”);
  free(pPrimes);
  pPrimes = NULL;
}
pPrimes = pTemp;
...
```

malloc, free, calloc and realloc

Probiere es aus:

Ändere das Primzahlen Programm so ab, dass die max. Anzahl Primzahlen, welche zu berechnen sind, nicht mehr angegeben werden muss.

String Handling mit Pointer

```
char *pString = NULL;
```

Ein `char *Pointer` alloziert noch kein Memory für den Inhalt des Strings.

```
const size_t BUF_SIZE = 100;          // Input buffer size
char buffer[BUF_SIZE];                // A 100 byte input buffer
scanf_s("%s", buffer, BUF_SIZE);      // Read a string
// Allocate space for the string
size_t length = strnlen_s(buffer, BUF_SIZE) + 1;
char *pString = malloc(length);
if(!pString)
{
  printf("Memory allocation failed.\n");
return 1; }
strcpy_s(pString, length, buffer);  // Copy string to new memory
printf("%s", pString);
free(pString);
pString = NULL;
```

Handling Array of Pointers

```
char *pS[10] = NULL;
```

```
#define STR_COUNT  10                          // Number of string pointers
const size_t BUF_SIZE = 100;                   // Input buffer size
char buffer[BUF_SIZE];                         // A 100 byte input buffer
char *pS[STR_COUNT] = {NULL};                  // Array of pointers
size_t str_size = 0;
for(size_t i = 0 ; i < STR_COUNT ; ++i)
{
  scanf_s("%s", buffer, BUF_SIZE);             // Read a string
  str_size = strnlen_s(buffer, BUF_SIZE) + 1;  // Bytes required
  pS[i] = malloc(str_size);                    // Allocate space for the string
  if(!pS[i]) return 1;                         // Allocation failed so end
  strcpy_s(pS[i], str_size, buffer);           // Copy string to new memory
}
// Do things with the strings...
// Release the heap memory
for(size_t i = 0 ; i < STR_COUNT ; ++i)
{
  free(pS[i]);
  pS[i] = NULL;
}
```

Handling Array of Pointers

Probiere es aus:

- Gegeben ist das Programm “TextAnalyser”. Es durchsucht ein eingegebener Text nach unterschiedlichen Wörtern und gibt die verschiedenen Wörter zusammen mit der Häufigkeit aus.
- Ändere das Programm so ab, dass der untersuchte Text und die Wörter auf dem Heap gespeichert werden. Die Länge des eingegebenen Textes ist unbestimmt. Verwende ein Array of Pointer anstelle eines mehrdimensionalen Arrays.

Pointers und Array Notation

Für einen Pointer kann eine Array-Notation verwendet werden

```
int count = 100;
double* data = calloc(count, sizeof(double));
...
for(int i = 0 ; i < count ; ++i)
  data[i] = (double)(i + 1)*(i + 1);
```

Die Array Notation ist besser lesbar!

Probiere es aus:

- Schreibe ein Programm, das mehrere eingegebene Sätze alphabetisch sortiert.