Übung 11

a) In einem C-Programm werden in *main()* drei Variablen wie folgt angelegt. Jedes Speicherwort habe 4 Byte. Geben Sie in nachfolgender Tabelle die fehlenden Werte an.

```
int x, n = 10, *y = &n;
```

Name der Variablen	Speicheradresse	Wert im Speicher	Typ der Variablen
x	4096		

b) Geben Sie an, welche der folgenden Aussagen zu Pointern in C richtig sind. Es können mehrere Antworten richtig sein.

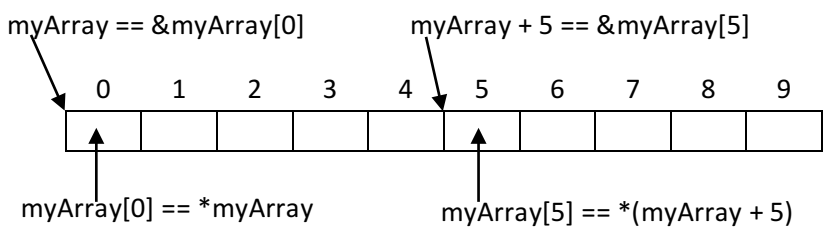
- O Pointer sind Zeiger auf Speicherplätze, deren Wert nicht auslesbar ist.
- O Pointer sind Zeiger auf Speicherplätze, deren Wert eine Speicherplatzadresse darstellt.
- O Die Adresse eines Speicherplatzes ist immer eine Hexadezimalzahl, die für deklarierte Variablen nicht bestimmt werden kann.
- O Die Adresse der Integer-Variable `x` kann mit `&x` bestimmt werden.
- O Wenn die Pointer wie folgt deklariert wurden: `int *p1, *p2;` dann zeigen nach der Ausführung der Zeile `p1=p2;` beide Zeiger auf die gleiche Speicherstelle.
- O Wenn die Pointer wie folgt deklariert wurden: `int *p1, *p2;` dann zeigt `p1` nach der Ausführung von `p1=p2+1;` auf die nächste Speicherstelle direkt hinter `p2`.

c) Ein Array ist in C auch über Pointer zugreifbar. Der vereinbarte Name selbst stellt die (konstante) Adresse des Arrays dar. Dazu ein Beispiel:

```
int myArray[10];    /* Feld, das 10 Integer aufnehmen kann */
```

Der Compiler reserviert den Speicherplatz auf dem Stack. Der Index des Arrays beginnt immer mit 0. Da hier `myArray` auf das erste Element zeigt, können die einzelnen Feldelemente auf verschiedene Weise adressiert werden (mit *&a* wird die Adresse der Speicherzelle geholt, die den Wert von *a* enthält) und auch auf verschiedene Weise auf die Inhalte zugegriffen werden.

Das folgende Bild zeigt Beispiele für die Adressierung und den Zugriff auf die Daten im Array.



i. Sei nun die Variable `ip` ein Zeiger auf einen Integer. Auf welches Element im Array zeigt `ip` nach Ausführung der folgenden beiden Zeilen:

```
int *ip = myArray;
ip += 2;
```

ii. Wie weisen Sie der Speicherzelle, auf die `ip` aktuell zeigt, einen Wert zu (z.B. 255)? Benutzen Sie dazu die Variable `ip`!

d) Geben Sie die Bedeutung der folgenden Ausdrücke an:

`float zahl[10];` _____

`*(zahl + 7)` _____

`int a[10][20];` _____

`a[6]` _____

`int *werte[10];` _____

e) Es sei ein Array in C wie folgt definiert:
`int array[] = {1, 1, 2, 3, 5, 8, 13, 21, 34, 55};`

i. Geben Sie für die folgenden Codezeilen an, welche Werte die Variablen je haben:

```
int a, b, c, d, *ptr = NULL;
a = *array;
b = *(array+3);
ptr = &array[3];
c = *ptr;
ptr = array + 2;
d = *ptr;
```

ii. Wie kann ein Pointer auf den Beginn eines Arrays gesetzt werden?

f) Ordnen Sie die folgenden Funktionsdeklarationen einer dieser Möglichkeiten zu.

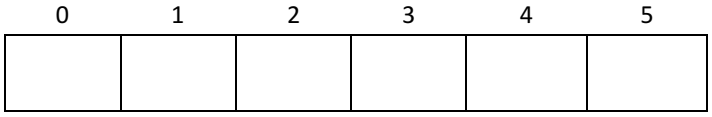
C-Deklarationen	Call-by-Value	Call-by-‚Reference'
void swap(float *a, float *b);		
void yoLetsGo(char name[]);		
int getMaximum(int a, int b);		
float getLength(float vec3[]);		

Vervollständigen Sie anschließend noch die folgenden Codezeilen entsprechend:

```
char str[] = "Jeff";
float p = 3.14f, q = 2.71f;
int x = 47, y = 11;

swap(_______,_______);

printf("Das Maximum ist ______.\n", getMaximum(______,______));

yoLetsGo(_______________);
```

g) Es sei ein Feld vom Typ `float` mit dem Namen `arr` gegeben. Wie sehen die Inhalte von `arr` nach der Abarbeitung folgender Anweisungen aus? Tragen Sie die Werte unten ein!

```
float arr[] = {2.0, 3.1, 1.7, 2.5, 1.0, 0.4};
arr[1] = arr[2];
*(arr + 3) = 4.2;
arr[5] = *(arr + 4) * *(arr + 5);
arr[4] = *arr;
```



h) Gegeben ist folgender Prototyp zur Berechnung des Kreuzproduktes zwischen zwei dreidimensionalen Vektoren im $\mathbb{R}^3$: `void cross(Vec3 c, Vec3 a, Vec3 b);`

Implementieren Sie vom vorangegangenen Foliensatz das Beispiel von Folie Nr. 6 fertig und schreiben Sie noch obige Funktion *cross()* so, dass folgende Rechenregel umgesetzt wird:

$$\vec{c} = \vec{a} \times \vec{b} = \begin{pmatrix} a_0 \\ a_1 \\ a_2 \end{pmatrix} \times \begin{pmatrix} b_0 \\ b_1 \\ b_2 \end{pmatrix} = \begin{pmatrix} a_1 b_2 - b_1 a_2 \\ b_0 a_2 - a_0 b_2 \\ a_0 b_1 - b_0 a_1 \end{pmatrix}$$

Legen Sie in *main()* die Variablen **u**, **v** und **w** vom Typ `Vec3` an, wobei **u** den Wert $(1, 0, 0)^T$ habe und **v** den Wert $(0, 1, 0)^T$. Rufen Sie nun mit den Eingabewerten **u**, **v** die Funktion auf, wobei das Ergebnis in **w** stehen soll. Geben Sie den Wert von **w** nach der Ausführung aus.

Führen Sie mit den gleichen Ausgangswerten jetzt noch, wie oben angegeben, den Aufruf `cross(u, u, v)` aus und geben Sie den Wert von **u** nach der Ausführung aus. Erklären Sie den Unterschied!

i) Legen Sie dynamisch ein Feld mit 20 Integerwerten auf dem Heap an und initialisieren Sie alle Feldelemente mit 0, vorausgesetzt die Speicherplatzanforderung war erfolgreich. Geben Sie in dem Fall anschließend nach Benutzung den Speicher wieder frei.

j) In Übung 9.b) sollten Sie ein kleines Programm zur Begrüßung des Users schreiben. Erweitern Sie dieses Programm so, dass die gesamte Begrüßung in einem String dynamischer Länge abgelegt wird. Hierzu muss erst die Länge des Gesamtbegrüßungsstrings bestimmt werden, um entsprechend Speicher auf dem Heap allokieren zu können. Mit *sprintf()* sollen dann beide Strings in das dynamisch angelegte „Array“ geschrieben werden. Vergessen Sie zuletzt nicht, vor Programmende den Speicher auch wieder freizugeben.