

Zeitbudget: 30 Minuten (maximal)

Aufgabe: Arbeiten Sie alleine diese Lernaufgabe durch. Gehen Sie Zeile für Zeile durch und befolgen Sie alle Anweisungen, d.h. Sie sollen erst weiterlesen, wenn Sie die Aufgaben gelöst haben und ihre Lösungen aufs Blatt geschrieben haben.

Inhalt:

1. Die Zeichenkette
2. Initialisierung
3. Zusatzmaterial (*string.h*)

1. Die Zeichenkette

Eine Zeichenkette ist eine Aneinanderreihung von einzelnen Zeichen (chars). Das Ende dieser Zeichenkette wird mit dem Null-Zeichen '\0' markiert. Die Zeichenkette "HSR Rapperswil" würde im Speicher wie folgt dargestellt werden:

H	S	R		R	a	p	p	e	r	s	w	i	l	\0
---	---	---	--	---	---	---	---	---	---	---	---	---	---	----

Im Kapitel 7.4 haben Sie gelernt, dass jedem Zeichen im Speicher eine Speicheradresse zugeordnet ist. Dies liesse sich wie folgt darstellen:

Adressen: 812 813 814 815 816 817 818 819 820 821 822 823 824 825 826

H	S	R		R	a	p	p	e	r	s	w	i	l	\0
---	---	---	--	---	---	---	---	---	---	---	---	---	---	----

Wenn wir die Speicheradresse kennen ist es nicht mehr schwer das zugehörige Zeichen zu ermitteln. Vervollständigen Sie folgende Tabelle:

Adresse	Zeichen
817	a
822	
813	
825	

Wir möchten uns nicht die Speicheradresse von jedem einzelnen Zeichen merken. Dies wäre viel zu aufwendig. Überlegen Sie sich:

1. Was müssen Sie sich merken, damit Sie die ganze Zeichenkette wieder aus dem Speicher lesen können?

.....

2. Was für eine Rolle spielt dabei das '\0' Terminierungssymbol?

.....

Blättern Sie erst um, wenn Sie die Fragen oben beantwortet haben!

Wir müssen uns nur die Adresse des ersten Zeichens merken. In unserem Beispiel wäre dies die 812. Um das zweite Zeichen zu erhalten, rechnen wir einfach $(812+1) = 813$. Für das dritte $(812+2) = 814$, usw. Bis wir an der Stelle $(812+14) = 826$ das Terminierungssymbol '\0' finden. Wenn wir auf dieses Symbol stossen, wissen wir, dass die Zeichenkette zu ende ist.

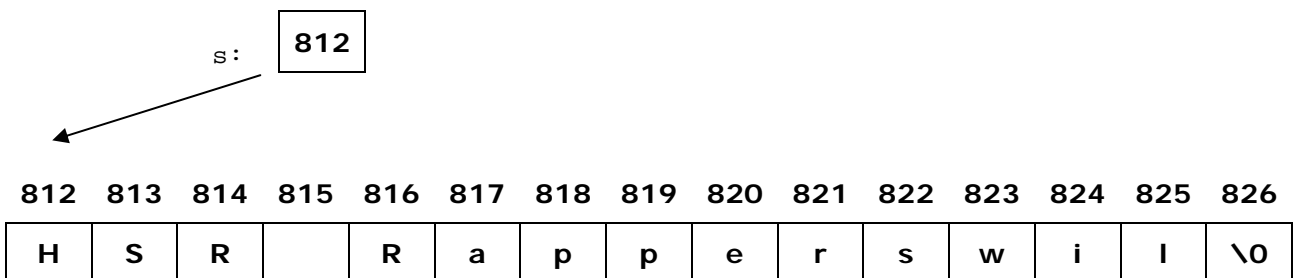
Die Adresse eines Zeichens speichern wir in einem Zeiger (Pointer):

```
char *s;
```

Der * ist das Zeiger(Pointer) Symbol. s wäre also ein Zeiger auf ein char.

Überblick:

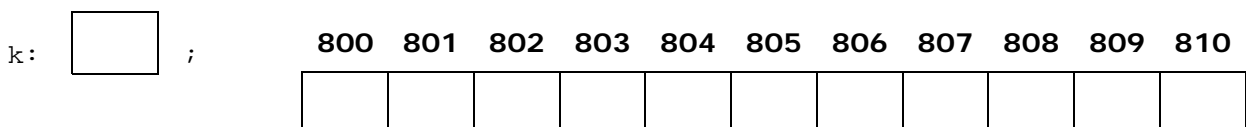
```
char *s = "HSR Rapperswil";
```



Aufgabe:

Ergänzen Sie die fehlenden Elemente.

```
char *k = "Jona";
```



Studieren Sie folgende Programmzeilen und ergänzen Sie die Ausgabe wo nötig.

```
cout << k;           // Ausgabe: Jona
cout << *k;          // Ausgabe: J
cout << *(k+1);      // Ausgabe: o
cout << s;           // Ausgabe: ...
cout << *(s+2);      // Ausgabe: ...
```

In der ersten Zeile wird dem cout ein Zeiger übergeben. Dies merkt das Programm und gibt die ganze Zeichenkette aus.

In der zweiten Zeile benutzen wir wieder den * Zeiger-Operator. Er weist den Computer an dem Zeiger zu folgen. Der Computer liest die Speicherstelle aus k, also 800, und übergibt diese dem cout. In der dritten Zeile übergeben wir dann die Speicheradresse 801.

Mit dem sogenannten Index-Operator [] können wir etwas einfacher auf einzelne Zeichen in der Zeichenkette zugreifen. Sie merken schnell was ich meine, wenn Sie sich in das nächste Beispiel vertiefen.

```
cout << *(k+0);      // Ausgabe: J
cout << k[0];        // Ausgabe: J

cout << *(k+1);      // Ausgabe: o
cout << k[1];        // Ausgabe: o

cout << *(k+3);      // Ausgabe: a
cout << k[3];        // Ausgabe: a
```

2. Initialisierung

Bis jetzt haben Sie eine Methode kennen gelernt, wie Sie eine Zeichenkette definieren können.

```
char *s = "HSR Rapperswil";
```

Hier zählt der Compiler selber wie viele Speicherplätze er für die Zeichenkette inklusive Terminierungssymbol benötigt. In unserem Beispiel 15 Speicherplätze.

Wir können aber dem Compiler auch direkt sagen, wie viel Platz er für unsere Zeichenkette reservieren muss. Dies benötigen wir zum Beispiel, wenn wir beabsichtigen verschieden lange Zeichenketten an der gleichen Stelle im Programm zu speichern;

```
char r[6]; //reserviere 6 Speicherplätze

for (int i=0; i<5; i++) {
    r[i] = 'X';
    r[i+1] = '\0';
    cout << r << endl;
}
```

// Ausgabe:
// X
// XX
// XXX
// XXXX
// XXXXX

Beachte, dass wir jedes Mal das Terminierungssymbol '\0' wieder am ende der Zeichenkette platzieren müssen damit cout weiss, wann die Zeichnkette zu ende ist.

Ergänzen Sie die folgende Tabelle für die einzelnen Iterationen von i.

r:	802	800	801	802	803	804	805	806	807	808	809	810
i = 0:												
i = 1:												
i = 2:												
i = 3:												
i = 4:												

Haben Sie darauf geachtet, dass die Zeichenkette erst an der Speicherstelle 802 und nicht an der Speicherstelle 800 beginnt?

Folgende Initialisierung wäre theoretisch auch möglich:

```
char p[5] = "hallo";
```

Sie ist aber gefährlich. So ist mir bei der obigen Initialisierung ein Fehler unterlaufen. Zeichnen Sie kurz für sich die Speicherbelegung dieser Initialisierung auf. Wo liegt der Fehler oder die Gefahr?

.....

Die Zeichenkette "hallo" besteht aus 5 Zeichen und dem Terminierungssymbol, insgesamt benötigen wir also Platz für 6 Zeichen. Ich habe aber nur Platz für 5 Zeichen reserviert. Somit hat es kein Platz für das Terminierungssymbol.

Sie können Zeichenketten auch zeichenweise initialisieren.

```
char a[6] = {'M', 'e', 'n', 's', 'a', '\0'}; // Ich muss genau zählen!
char b[] = {'M', 'e', 'n', 's', 'a', '\0'}; // Der Compiler zählt für mich.
char c[] = {77,101,110,115,97,0}; // Sie können auch die ASCII
// Nummer des Zeichens angeben.
```

Im ersten Teil haben Sie den * Zeiger (Pointer) Operator kennen gelernt. Der & Adressoperator bewirkt genau das Gegenteil.

```
char *s = "HSR Rapperswil";
if (s == &*s)
    cout << ";-)";
```

Erinnern wir uns nochmals an unser erstes Beispiel.

812	813	814	815	816	817	818	819	820	821	822	823	824	825	826
H	S	R		R	a	p	p	e	r	s	w	i	l	\0

Die Adresse des fünften Zeichens erhalten Sie mit &s[4].

```
char *s = "HSR Rapperswil";
cout << &s[4]; // Ausgabe: Rapperswil
```

Wenn wir aber &s[4] auf das cout hinausschreiben erscheint in der Ausgabe nicht nur das "R" sondern "Rapperswil". Was geschieht hier genau?

.....

.....

.....

Tipp: Falls es ihnen schwer fällt nachzuvollziehen was hier passiert, zeichnen Sie sich nochmals ganz genau die Speicherbelegung auf und lesen Sie erneut den unteren Teil der zweiten Seite.

Die Frage liegt nun nahe was passiert, wenn Sie &s Ausgeben? Haben Sie eine Idee?

.....

.....

Der Zeiger s befindet sich auch an einem bestimmten Ort im Speicher. Dieser Ort hat selbstverständlich auch eine Adresse. Eine Speicherbelegung könnte zum Beispiel wie folgt aussehen:

s:

797	798	799	800	801	802	803	804	805	806	807	808	809	810	811
								812						

812	813	814	815	816	817	818	819	820	821	822	823	824	825	826
H	S	R		R	a	p	p	e	r	s	w	i	l	\0

Was wird wo ausgegeben. Vervollständigen Sie.

```
cout << &s;           // Ausgabe: 805
cout << s;           // Ausgabe: ...
cout << *s;          // Ausgabe: ...
cout << s[0];        // Ausgabe: ...
cout << s[5];        // Ausgabe: ...
cout << *(s+5);      // Ausgabe: ...
cout << &s[0];       // Ausgabe: ...
```

Gratulation Sie haben es geschafft. Sie sind nun eine junior Expertin / ein junior Experte in Zeichenketten!

3. Zusatzmaterial (*string.h*)

Die Standardlibrary `string.h` enthält viele nützliche Funktionen für das Arbeiten mit Zeichenketten (Strings). In der Übung 9 werden Sie diese einsetzen. Ich hab für Sie eine Übersicht zusammengestellt.

Verschaffen Sie sich einen groben Überblick über die Library und beantworten Sie folgende Fragen:

1. Wie viele Funktionen enthält die Library `string.h` ?
2. Welche 3 Funktionen scheinen ihnen besonders nützlich?
 - i.
 - ii.
 - iii.

Lesen Sie nun die Beschreibung der Library Funktion für Funktion durch und studieren Sie die Beispiele. Lassen Sie sich nicht verwirren, dass für die Ein- und Ausgabe C-Funktionen verwendet werden. Sie können alle `string.h` Funktionen in C++ ganz genau gleich einsetzen wie in C.