

☒ Gruppe 1 (J. Heinzelreiter)☐ Gruppe 2 (M. Hava)Name: PAPESH KonstantinAufwand [h]: 4☐ Gruppe 3 (P. Kulczycki)

Übungsleiter/Tutor: \_\_\_\_\_

Punkte: \_\_\_\_\_

Beispiel	Lösungsidee (max. 100%)	Implement. (max. 100%)	Testen (max. 100%)
1 (40 P)	100	100	90
2 (60 P)	100	100	100

Machen Sie sich – soweit für diese Aufgaben notwendig – mit der Programmiersprache C, dem GNU-Compiler gcc und mit der Behandlung von Kommandozeilenparametern (mittels argc und argv) in der Funktion main vertraut.

**Beispiel 1: NOVA-Rechner (nova/)**

Schreiben Sie ein C-Programm, das auf der Kommandozeile folgende Daten liest:

- durchschnittlichen Verbrauch
- Kraftstoffart (z.B. 1 für Diesel, 2 für Benzin)

und die Normverbrauchsabgabe (NOVA) berechnet und auf der Kommandozeile formatiert ausgibt. Laut Bundesministerium für Finanzen berechnet sich die NOVA in Prozent wie folgt:

- bei Benzin:  $(\text{Verbrauch in l/100 km} - 3) \cdot 2 \cdot 100$
- bei Diesel:  $(\text{Verbrauch in l/100 km} - 2) \cdot 2 \cdot 100$

Berechnen Sie die NOVA für mindestens fünf verschiedene Automodelle. Entnehmen Sie die durchschnittlichen Verbrauchswerte aus den Datenblättern der Hersteller. Dann berechnen Sie die NOVA für die gewählten Automodelle, indem Sie aus dem [www.spritmonitor.de](http://www.spritmonitor.de) den durchschnittlichen Verbrauch entnehmen, der von der autofahrenden Bevölkerung selbst ermittelt wurde. Stellen Sie die NOVA-Werte gegenüber.

**Beispiel 2: Umfang eines Polygons (poly/)**

Entwickeln Sie ein C-Programm, dem als Kommandozeilenparameter die Koordinaten der Eckpunkte eines Polygons übergeben werden und welches den Umfang dieses Polygons berechnet und ausgibt. Die Eckpunkte werden immer durch zwei ganzzahlige Werte (x- und y-Wert) angegeben und können auch im negativen Teil des Koordinatensystems liegen. Achten Sie darauf, dass

- die Ausgabe sowohl die Eingabe (die Koordinaten) als auch das Ergebnis (den Umfang) in übersichtlicher Form darstellt,
- bei falscher Anzahl von Kommandozeilenparameter (ungerade Anzahl oder weniger als drei Punkte) eine sinnvolle Fehlermeldung ausgegeben wird und
- bei Aufruf des Programms ohne Parameter eine Erklärung ausgegeben wird. Beachten Sie dabei, dass der Benutzer das Programm beliebig umbenennen kann und dass in dieser Erklärung der aktuelle Programmname verwendet wird.

**Hinweise (diese gelten auch für alle weiteren Übungen)**

(1) Lesen Sie die organisatorischen Hinweise. (2) Geben Sie für alle Ihre Lösungen immer eine Lösungsidee an. (3) Kommentieren und testen Sie Ihre Programme ausführlich.

# SWO31 & SWB31 Softwareentwicklung mit klassischen Sprachen und Bibliotheken – WS 2018/19

## Übungsabgabe 1

Konstantin Papesh

15. Oktober 2018

### 1.1 NOVA-Rechner

#### 1.1.1 Lösungsidee

Gegeben werden zwei Werte:

- Durchschnittlicher Verbrauch *avgConsumption*
- Kraftstoffart *fuelType*

Diese werden in der Konsole als Argumente mitgegeben. Zuerst analysiert das Programm, ob zu viele oder zu wenige Argumente mitgegeben werden. Passt die Anzahl der Argumente, werden diese konvertiert und im Falle der Kraftstoffart geparkt (Ob Diesel oder Benzin). Danach wird mithilfe der gegebenen Formel die NOVA berechnet und ausgegeben.

#### 1.1.2 Implementierung

**Listing 1.1:** main.c

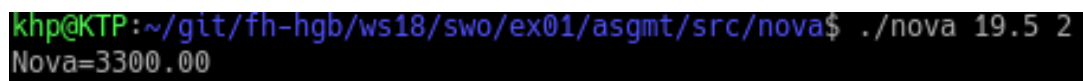
```
1 #include <stdio.h>
2 #include <stdlib.h>
3 #define DIESEL 1
4 #define BENZIN 2
5
6 int main(int argc, char * argv[]){
7     if(argc != 3) {
8         printf("Invalid number of arguments!\n");
9         return 2;
10    }
11
12    float avgConsumption = atof(argv[1]);
13    int fuelType = atoi(argv[2]);
14    float nova = 0;
15    int subtractor;
16
17    if(fuelType == DIESEL)
18        subtractor = 2;
```

```

19  else if(fuelType == BENZIN)
20      subtractor = 3;
21  else {
22      printf("Invalid fuel type!\n");
23      return 1;
24  }
25  nova = (avgConsumption - subtractor) * 200;
26  printf("Nova=%.2f\n",nova);
27  return 0;
28 }

```

### 1.1.3 Testen



```

khp@KTP:~/git/fh-hgb/ws18/swo/ex01/asgmt/src/nova$ ./nova 19.5 2
Nova=3300.00

```

Abbildung 1.1: Beispielhafte Eingabe für eine NOVA-Berechnung

### Gegenüberstellung

Automodell	KS	Verbrauch H	Verbrauch SM	NOVA H	NOVA SM
VW Polo	D	3.9l/100km	3.7l/100km	380	340
Opel Corsa	D	3.5l/100km	4.14l/100km	300	428
Alfa Romeo 145	B	7.9l/100km	7.34l/100km	980	868
Mazda MX-5	B	6.6l/100km	5.65l/100km	720	530
Lamborghini Gallardo	B	19.5l/100km	17.82l/100km	3300	2964

Wobei KS für Kraftstoff, H für Hersteller und SM für Spritmonitor.de steht. B und D stehen respektiverweise für Benzin und Diesel.

## 1.2 Umfang eines Polygons

### 1.2.1 Lösungsidee

Wieder werden vom Programm Argumente übernommen und verarbeitet. In diesem Fall ist die Anzahl der Argumente jedoch variable, bzw. treten in 2er Paaren auf. Dadurch, dass auch der Programmname als Argument gezählt wird, muss somit im Programm überprüft werden, ob die Anzahl der Argumente ungerade ist. Weiters müssen mindestens 3 Punkte spezifiziert werden, damit eine Fläche berechnet werden kann. Daraus ergibt sich, dass mindestens 7 Argumente benötigt werden, um das gewünschte Ergebnis, der Umfang des Polygons, zu berechnen. Da die Punkte als ganzzahlige Werte angegeben sind, reicht eine Konvertierung von string auf integer. Dann wird mithilfe einer for-Schleife die Distanz zwischen den gegebenen Punkten ausgerechnet. Als letztes wird dann noch die Distanz zwischen letztem und erstem Punkt berechnet, zur Summe dazugezählt und diese ausgegeben.

### 1.2.2 Implementierung

**Listing 1.2:** main.c

```

1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <math.h>
4
5 int main(int argc, char * argv[]){
6     double polyPerimeter = 0;
7     if(argc == 1) {
8         printf("Usage: %s point1x point1y point2x point2y point3x point3y [pointNx
          pointNy]\n", argv[0]);
9         return 1;
10    } else if(argc < 7 || argc%2 == 0) {
11        printf("Wrong number of arguments. At least 3 points have to be specified!\n");
12        return 2;
13    }
14    int i;
15    int firstX = atoi(argv[1]);
16    int prevX = firstX;
17    int firstY = atoi(argv[2]);
18    int prevY = firstY;
19
20    for(i = 3; i < argc-1; i+=2){ //start with 3 --> distance 2nd point to 1st point
21        int px = atoi(argv[i]);
22        int py = atoi(argv[i+1]);
23
24        printf("P%d[%d,%d]\n", i/2, prevX, prevY);
25        polyPerimeter += sqrt(((px-prevX)*(px-prevX)) + ((py-prevY)*(py-prevY)));
26        prevX = px;
27        prevY = py;
28    }
29    polyPerimeter += sqrt(((firstX-prevX)*(firstX-prevX)) + ((firstY-prevY)*(firstY-
          prevY)));
30    printf("P%d[%d,%d]\nUmfang=%f\n", i/2, prevX, prevY, polyPerimeter);
31    return 0;

```

```
32 }
```

### 1.2.3 Testen

```
khp@KTP:~/git/fh-hgb/ws18/swo/ex01/asgmt/src/poly$ ./a.out
Usage: ./a.out point1x point1y point2x point2y point3x point3y [pointNx pointNy]
```

**Abbildung 1.2:** Fehlermeldung, sollte nur der Programmname angegeben werden.

```
khp@KTP:~/git/fh-hgb/ws18/swo/ex01/asgmt/src/poly$ ./poly
Usage: ./poly point1x point1y point2x point2y point3x point3y [pointNx pointNy]
```

**Abbildung 1.3:** Sollte das Programm umbenannt werden, wird dies in der Hilfe berücksichtigt

```
khp@KTP:~/git/fh-hgb/ws18/swo/ex01/asgmt/src/poly$ ./a.out 0 0 0 1 1 1 1 0 1
Wrong number of arguments. At least 3 points have to be specified!
```

**Abbildung 1.4:** Fehlermeldung, wenn zu wenige oder eine falsche Anzahl an Koordinaten angegeben wird.

```
khp@KTP:~/git/fh-hgb/ws18/swo/ex01/asgmt/src/poly$ ./a.out 0 0 0 1 1 1 1 0 1 2
P1[0,0]
P2[0,1]
P3[1,1]
P4[1,0]
P5[1,2]
Umfang=7.236068
```

**Abbildung 1.5:** Beispielhafte Eingabe für eine Umfangs-Berechnung