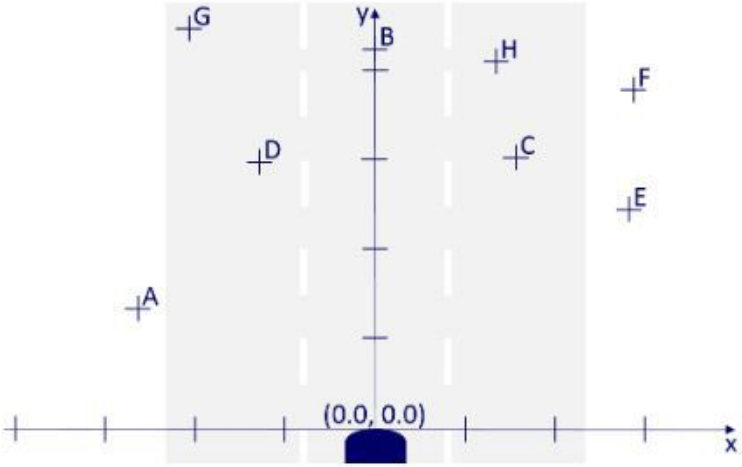
Abstands- und Nachbarpunkte/Distance and Neighbouring Points



Moderne Fahrzeuge (Autos, Flugzeuge, Flurförderzeuge, Roboter, ...) können mit Infrarot-, Kamera-, Laser-, Radar-, ... Sensoren ausgestattet werden, die fortlaufend zyklisch Hindernisse in Bewegungsrichtung detektieren. Diese Daten werden für Abstands-, Ausweich-, Brems-, ... Assistenten benötigt. Als vereinfachte Version hierzu sollen in dieser Aufgabe Hindernisse als Punkte modelliert werden, die von einem gedachten Sensorsystem vorne in Fahrzeugmitte als Ursprung eines relativen lokalen zweidimensionalen kartesischen Koordinatensystems plaziert detektiert werden. Es sollen beliebig viele solcher Hindernisse/Punkte vor dem Fahrzeug in einer Liste gespeichert werden. Diese Liste soll nach deren euklidischem Abstand vom Ursprung sortiert aufgebaut werden soll (siehe Abbildung oben und Beispiele unten)./
Modern vehicles (cars, industrial trucks, planes, robots, ...) have camera, infrared, laser, radar, ... sensors to continuously scan obstacles in the direction of movement. Their data are needed for distance, collision avoidance, breaking assistance, ... systems. In this task a simplified version of obstacles shall be modeled by points detected by an imaginary sensor system placed in the front and middle of the vehicle as origin of a relative local two dimensional cartesian coordinate system. Arbitrary many of such obstacles/points in front of a vehicle shall be stored in a list. This list shall be sorted by the Euclidian distance from the origin (see picure above and examples below).

Teilaufgabe 1/Subtask 1

Zur einfachen Identifizierung sollen für jedes Hindernis/jeden Punkt eine Zeichenkette ("A", "B", ... im Beispiel oben), dessen Koordinaten (x, y) im lokalen Koordinatensystem und der Abstand zum Ursprung gespeichert werden. Schreiben Sie die Typdefinition für eine Struktur mit diesen vier Daten sowie einem Zeiger auf das darauf folgende Listenelement als Komponenten/Variablen der Struktur.)/
To simply identify them each obstacle/point shall store a string ("A", "B", ... in example above), its coordinate (x, y) in the local coordinate system and its distance to the origin. Write a type definition for a structure with these four data as well as a pointer to the next list element as components/variables of the structure.

Teilaufgabe 2/Subtask 2

Schreiben Sie eine Funktion zur Berechnung des euklidischen Abstands d zweier Hindernisse/Punkte P1 und P2 mit den Koordinaten (x_1, y_1) und (x_2, y_2) nach der Formel: $d = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$ und geben d als Funktionswert zurück./
Write a function to calculate the Euclidian distance d of two obstacles/points P1 and P2 with coordinates (x_1, y_1) and (x_2, y_2) by the formula:
$d = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$ and return d as function value.

Teilaufgabe 3/Subtask 3

Schreiben Sie eine Funktion, die für eine Liste von Hindernissen/Punkten als ersten und ein Hindernis/einen Punkt als zweiten Parameter einen Zeiger auf das nach Abstand nächstliegende Hindernis/den nach Abstand nächstliegenden Punkt zurück liefert. Beachten Sie, dass ein Hindernis/Punkt nicht zu sich selbst (Abstand 0) zurück gegeben wird, sowie dass es mindestens zwei Hindernisse/Punkte geben muss, ansonsten soll der Nullzeiger `nullptr` zurück gegeben werden./
Write a function with a list of obstacles/points as first and one obstacle/point as second parameter calculating the neareast other obstacle/point to it and returning it as

Write a function with a list of obstacles/points as first and one obstacle/point as second parameter calculating the neareast other obstacle/point to it and returning it as pointer. Take care that an obstacle/point does not return itself (distance 0) as well as that at least two obstacles/points need to exist, otherwise a null pointer `nullptr` shall be returned.

Teilaufgabe 4/Subtask 4

Schreiben Sie eine Funktion, die in eine nach Abstand vom Ursprung des Koordinatensystems aufsteigend sortierte Liste von Hindernissen/Punkten als ersten Parameter hat und als zweiten Parameter einen Zeiger auf ein neues Hindernis/einen neuen Punkt. Sortieren Sie im Rumpf dieses neue Hindernis/diesen neuen Punkt in die Liste ein und geben einen Zeiger auf den (neuen) Kopf der Liste von Hindernissen/Punkten zurück./
Write a function with a list of obstacles/points sorted by distance from the origin of the coordinate system as first parameter and a pointer to a new obstacle/point as second parameter. In its body insert sorted this new obstacle/point into the list and return a pointer to the (new) head of the list of obstacles/points.

Teilaufgabe 5/Subtask 5

Schreiben Sie eine Funktion, die die Liste aller Hindernisse/Punkte formatiert ausgibt (alle Zahlen mit sechs Zeichen und zwei Nachkommastellen). Bei dieser Ausgabe sollen neben der identifizierenden Zeichenkette des Hindernisses/Punkts, dessen Koordinaten und dessen Abstand zusätzlich das jeweils nach Abstand nächste Hindernis/der nach Abstand nächste Nachbarpunkt zu diesem bestimmt und ausgegeben werden (rufen Sie dazu jeweils die Funktion aus Teilaufgabe 3 auf; siehe Beispiel unten)./
Write a function to output formatted the list of obstacles/points (all numbers with six characters and two digits after decimal point). Beside the identifying string of the obstacle/point, its coordinates and its distance additionally the nearest neighbour obstacle/point to it shall be found and outputted (for each call the function from subtask 3; see example below).

Teilaufgabe/Subtask 6

Schreiben Sie eine Funktion, die die gesamte Liste aller Hindernisse/Punkte auf dem Heap löscht./
Write a function to delete the whole list of all obstacles/points completely on the heap.

Teilaufgabe 7/Subtask 7

In einer Schleife in der Funktion `main` sollen beliebig viele Hindernisse/Punkte eingegeben und aufsteigend nach Abstand einsortiert in einer Liste gespeichert werden - rufen Sie hierzu für jedes eingegebene Hindernis/jeden eingegebenen Punkt die Funktion aus Teilaufgabe 4 auf. Nach dem Eingabeende soll die nach Abstand sortierte Liste über einen Aufruf der in Teilaufgabe 5 definierten Funktion ausgegeben werden. Zuletzt soll die gesamte Liste wieder gelöscht werden über einen Aufruf der in Teilaufgabe 6 definierten Funktion./
In a loop in function `main` arbitrary many of such obstacles/points shall be inputted and stored in a list sorted by distance in ascending order - call for each inputted obstacle/point the function from subtask 4. After the end of input output the list by calling the function from subtask 5. Last delete the whole list by calling the function from subtask 6.

Wichtig zu beachten/Important to Regard

Verwenden Sie ausschließlich Ein- und Ausgaben über C++, **keine Aufrufe** von `scanf` oder `printf`, nur `new` und `delete` (nicht `malloc`, `calloc`, `realloc` oder `free`. Ihre Datei muss `h2_IhreMatrikelnummer.cpp` heissen, die Endung muss für den Plagiatschecker sowie für unsere Prüfprogramme `.cpp` sein, darf also **keine Textdatei** mit Endung `.txt` sein, **keine Projektdatei** `.cbp` oder ähnlich und auch **keine** `.rar`, `.zip`-, **... Datei** (diese kann der Plagiatschecker nicht verarbeiten, Sie würden also 0 Punkte bekommen), Ihr Programm darf auch **nur ASCII-Zeichen** enthalten (also keine Sonderzeichen aus nicht lateinischen Alphabeten o.ä. enthalten), **nur die Standard-C++-Bibliotheken** einbinden (also keine mit Endung `.h` wie `conio.h`, `stdio.h`, `windows.h`, ...) und soll dem Standard **C++11** (oder neuer) folgen. Beginnen Sie Ihre Datei mit einem Kommentar `/* ... */` mit Ihren Daten, nicht mit `/ * ... * /` oder ... oder komplett ohne Kommentartags.
Only use C++ input and output, **no calls** of `scanf` or `printf` function, only `new` and `delete` (not `malloc`, `calloc`, `realloc` or `free`. Your file has to have name `h2_yourMatriculationNumber.cpp`, the ending `.cpp` is essential for the plagiarism checker as well as for our check programs, `no text file` with ending `.txt`, **no project file** `.cbp` or similar and **no** `.rar`, `.zip`, **... file** (the plagiarism checker does not understand all these formats, therefore you would get 0 points for it), also your program is **only allowed containing ASCII characters** (no characters or signs from non Latin ones), **only includes of standard C++ libraries** (i.e. no libraries with ending `.h` like `conio.h`, `stdio.h`, `windows.h`, ...) and shall follow standard **C++11** (or newer). Start your file with a comment `/* ... */` giving your

libraries with ending `.h` like `conio.h`, `stdio.h`, `windows.h`, ...) and shall follow standard **C++11** (or newer). Start your file with a comment `/* ... */` giving your personal data, not with `/ * ... * /` or ... or completely without C++ comment tags.

Beispiel 1 Programmlauf/Example 1 Program Run:

```
string describing obstacle ("end" for end of input): A
x and y coordinate: 0 1
string describing obstacle ("end" for end of input): X
x and y coordinate: 1 1
string describing obstacle ("end" for end of input): E
x and y coordinate: 0 3
string describing obstacle ("end" for end of input): K
x and y coordinate: -1 4
string describing obstacle ("end" for end of input): W
x and y coordinate: 0 10
string describing obstacle ("end" for end of input): end
obstacle A: (  0.00,   1.00), distance:  1.00m, nearest to this: X
obstacle X: (  1.00,   1.00), distance:  1.41m, nearest to this: A
obstacle E: (  0.00,   3.00), distance:  3.00m, nearest to this: K
obstacle K: ( -1.00,   4.00), distance:  4.12m, nearest to this: E
obstacle W: (  0.00,  10.00), distance: 10.00m, nearest to this: K
delete: A X E K W
```

Beispiel 2 Programmlauf/Example 2 Program Run:

```
string describing obstacle ("end" for end of input): A
x and y coordinate: -27.2 12.8
string describing obstacle ("end" for end of input): B
x and y coordinate: 0.1 41.4
string describing obstacle ("end" for end of input): C
x and y coordinate: 15.9 30.25
string describing obstacle ("end" for end of input): D
x and y coordinate: -12.74 29.13
string describing obstacle ("end" for end of input): E
x and y coordinate: 27.68 23.45
string describing obstacle ("end" for end of input): F
x and y coordinate: 29.41 37.92
string describing obstacle ("end" for end of input): G
x and y coordinate: -21.03 45.19
string describing obstacle ("end" for end of input): H
x and y coordinate: 13.47 42.1
string describing obstacle ("end" for end of input): end
obstacle A: (-27.20,  12.80), distance: 30.06m, nearest to this: D
obstacle D: (-12.74,  29.13), distance: 31.79m, nearest to this: B
obstacle C: ( 15.90,  30.25), distance: 34.17m, nearest to this: H
obstacle E: ( 27.68,  23.45), distance: 36.28m, nearest to this: C
obstacle B: (  0.10,  41.40), distance: 41.40m, nearest to this: H
```

Beispiel 2 Programmlauf/Example 2 Program Run:

```
string describing obstacle ("end" for end of input): A
x and y coordinate: -27.2 12.8
string describing obstacle ("end" for end of input): B
x and y coordinate: 0.1 41.4
string describing obstacle ("end" for end of input): C
x and y coordinate: 15.9 30.25
string describing obstacle ("end" for end of input): D
x and y coordinate: -12.74 29.13
string describing obstacle ("end" for end of input): E
x and y coordinate: 27.68 23.45
string describing obstacle ("end" for end of input): F
x and y coordinate: 29.41 37.92
string describing obstacle ("end" for end of input): G
x and y coordinate: -21.03 45.19
string describing obstacle ("end" for end of input): H
x and y coordinate: 13.47 42.1
string describing obstacle ("end" for end of input): end
obstacle A: (-27.20,  12.80), distance: 30.06m, nearest to this: D
obstacle D: (-12.74,  29.13), distance: 31.79m, nearest to this: B
obstacle C: ( 15.90,  30.25), distance: 34.17m, nearest to this: H
obstacle E: ( 27.68,  23.45), distance: 36.28m, nearest to this: C
obstacle B: (  0.10,  41.40), distance: 41.40m, nearest to this: H
obstacle H: ( 13.47,  42.10), distance: 44.20m, nearest to this: C
obstacle F: ( 29.41,  37.92), distance: 47.99m, nearest to this: E
obstacle G: (-21.03,  45.19), distance: 49.84m, nearest to this: D
delete: A D C E B H F G
```