École de Technologie Supérieure **Automne 2022**

TCH009 – INFORMATIQUE DEVOIR

Enseignant : Florent Hernandez

DIRECTIVES

- Date limite de remise : **dimanche 11 décembre à 23h59**
- Travail par équipes de 3 ou 4.
- Date limite de constitution des équipes : **dimanche 13 novembre à 23h59**
- Une seule remise par équipe.
- Vous devez remettre vos fichiers **main.c** et **matrac_lib.c** (ou tout le dossier de votre solution *Visual Studio*. Dans ce cas, le dossier doit être compressé dans un seul fichier *zip*).
- La remise doit se faire sur ***Moodle***

Matrac – Mon Analyseur de Traces GPS

Table des matières

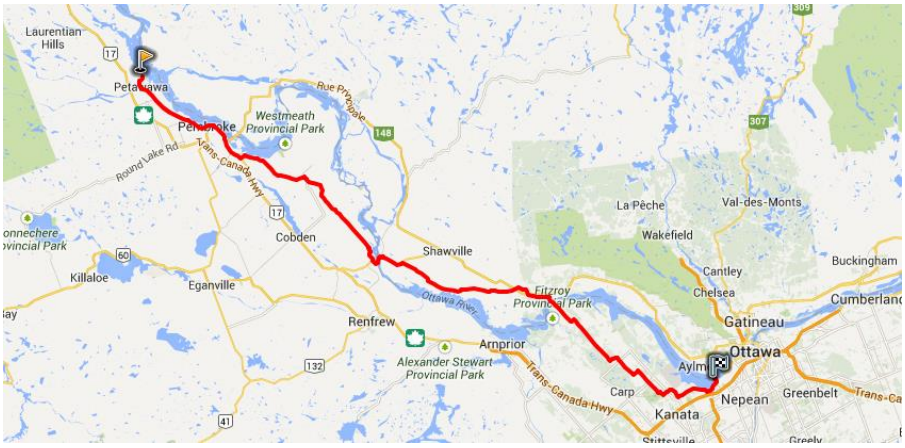
1. Objectifs

Ce travail a pour but de vous familiariser :

- Avec la résolution de problèmes à l'aide du langage C;
- À l'utilisation et à la création de fonctions et de librairies;
- À la déclaration et à l'utilisation de variables et de constantes;
- À la déclaration et à l'utilisation de variables et de tableaux;
- À l'utilisation de sélections (if, switch…case) et d'itérations (for, while);
- À l'accès à des fichiers de données;

2. Description générale:

Avec un récepteur GPS, il est aujourd'hui très facile d'obtenir les données décrivant une position sur la Terre. Si le GPS se déplace en sauvegardant à intervalles régulier ces données, il crée une *Trace* (*Track*) qui décrit géographiquement le parcours réalisé.



Exemple de trace enregistrée par un GPS lors d'un parcours en vélo entre Petawawa et Ottawa (ON). Image tirée d'une analyse de trace faite sur le site ibpindex.com

Le but de ce devoir est d'écrire un programme capable de lire un fichier de trace GPS et de réaliser les calculs d'analyse suivants sur les données récupérées :

- Distance totale parcourue
- Ascension totale réalisé
- Altitude maximum/minimum atteinte
- Pente maximale du parcours

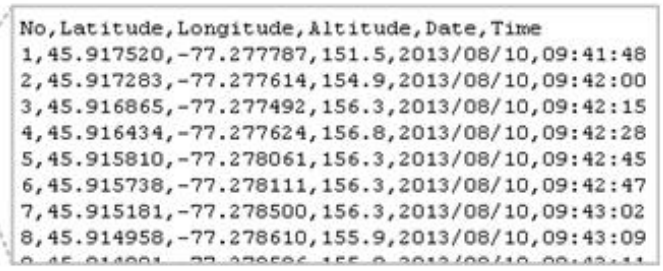
3. Processus général d'analyse de trace

Fichier GPX (fourni)

Programme de conversion de format GPS (fourni)

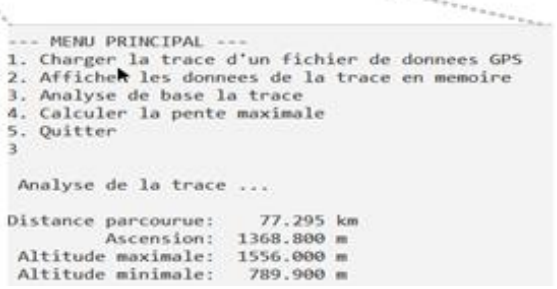
Exemple: GPSBabel

Fichier CSV (fourni)



Programme d'analyse (Matrac)

Exemple: ibp index



Fichier GPX

La majorité des GPS peuvent exporter leurs traces, routes et autres points dans le format standard GPX (GPS Exchange), et la majorité des programmes d'affichage ou d'analyse de données GPS sont capable de lire ce type de fichiers. À titre de référence, un ensemble de fichiers GPX est fourni, mais dans notre cas, l'extraction des données de tels fichiers est un défi que nous éviterons.

Convertisseur de Format GPS

Pour simplifier le problème d'accès aux données, certain fichiers GPX ont été convertis en fichier CSV. Ces derniers sont beaucoup plus simples à décoder et suffisent pour notre application. De nombreux logiciels peuvent être utilisés pour convertir les fichiers de données GPS; dans notre cas, c'est le logiciel gratuit GPSBabel qui a été utilisé (*XML GPX → UNIVERSAL CSV*). **Ce travail de conversion est déjà fait pour vous.**

Fichier CSV

Typiquement un fichier CSV (Comma-separated values) est un fichier texte contenant des valeurs séparées par des virgules. Les fichiers CSV de données GPS fournis pour ce devoir ont une structure très proche d'un tableau de données :

- Une ligne d'en-tête donne les noms des colonnes
- Chaque ligne supplémentaire décrit un point de la trace (latitude, longitude, etc.)
- Chaque valeur est séparée par une virgule ce qui en simplifie la lecture par un programme informatique

```
No,Latitude,Longitude,Altitude,Date,Time
1,45.917520,-77.277787,151.5,2013/08/10,09:41:48
2,45.917283,-77.277614,154.9,2013/08/10,09:42:00
3,45.916865,-77.277492,156.3,2013/08/10,09:42:15
4,45.916434,-77.277624,156.8,2013/08/10,09:42:28
5,45.915810,-77.278061,156.3,2013/08/10,09:42:45
6,45.915738,-77.278111,156.3,2013/08/10,09:42:47
7,45.915181,-77.278500,156.3,2013/08/10,09:43:02
```

Exemple de fichier CSV contenant les données GPS d'une trace

Votre programme devra pouvoir lire ce type de fichier pour mémoriser les données d'une trace et ensuite calculer certaines propriétés du parcours décrit.

4. Matrac : Analyseur de traces GPS

Le programme demandé doit offrir le menu principal suivant :

- Charger : chargement en mémoire des données d'un fichier GPS (format CSV)
- Afficher : affichage des point (coordonnées 3D) de la trace mise en mémoire
- Analyser : analyse des caractéristiques de base de la trace
- Calculer : calcul de la pente maximale de la trace
- Quitter : fin du programme

Charger

L'option *charger* permet de lire le contenu d'un fichier CSV pour mettre en mémoire toutes les données de position. Utilisez la fonction ***lire_fichier_gpx*** de la librairie ***matrac_lib*** pour charger les données du fichier vers des tableaux.
En complément, un résumé des principales fonctions de manipulation de fichiers en C est donné en *Annexe A*. La compréhension de cette annexe n'est pas nécessaire pour faire le devoir.

Les contraintes suivantes doivent être respectées :

- Le programme doit demander à l'utilisateur le nom du fichier à charger
- Le programme doit afficher un message d'erreur en cas de problème d'ouverture de fichier (ex : fichier absent)
- Le fichier de données doit se trouver dans le répertoire principal du projet
- Toutes les données pertinentes doivent être chargées dans un (des) tableau(x)
- Étant donnée la nature statique des tableaux, une constante MAXPOINT doit fixer le nombre maximal de point acceptés. Par exemple dix-mille (10000).
- Si une trace contient plus de points que le maximum permis, les points supplémentaires sont ignorés.

```
--- MENU PRINCIPAL ---
1. Charger la trace d'un fichier de donnees GPS
2. Afficher les donnees de la trace en memoire
3. Analyse de base la trace
4. Calculer la pente maximale
5. Quitter
1

Nom du fichier:3juillet.txt

Chargement de fichier 3juillet.txt...
...probleme de lecture du fichier


--- MENU PRINCIPAL ---
1. Charger la trace d'un fichier de donnees GPS
2. Afficher les donnees de la trace en memoire
3. Analyse de base la trace
4. Calculer la pente maximale
5. Quitter
1

Nom du fichier:4juillet.txt

Chargement de fichier 4juillet.txt...
...2482 points lus
```

Exemples d'exécutions : lecture d'un fichier CSV

Afficher

L'option *afficher* permet d'afficher les données qui ont été mises en mémoire lors de la lecture du fichier. Cette option permet de visualiser le contenu de la mémoire et de vérifier que la lecture à bien été réalisée.

Les contraintes suivantes doivent être respectées :

- Le format d'affichage doit être clair et compact pour vérifier facilement la validité des données en mémoire
- Au minimum, les valeurs de latitude, longitude et altitude doivent absolument apparaître dans cet affichage

```
--- MENU PRINCIPAL ---
1. Charger la trace d'un fichier de donnees GPS
2. Afficher les donnees de la trace en memoire
3. Analyse de base la trace
4. Calculer la pente maximale
5. Quitter
2

 Donnees en memoire:

 index      lattitude  longitude  altitude
         0  51.296203 -116.958720 794.700000
         1  51.296198 -116.959300 794.200000
         2  51.296195 -116.959988 794.200000
         3  51.296203 -116.960780 794.200000
         4  51.296206 -116.961556 793.200000
         5  51.296205 -116.962020 792.700000
```

Exemple d'exécution : affichage des données en mémoire

Analyser

L'option *analyser* permet de faire les calculs de bases suivants sur la trace en mémoire :

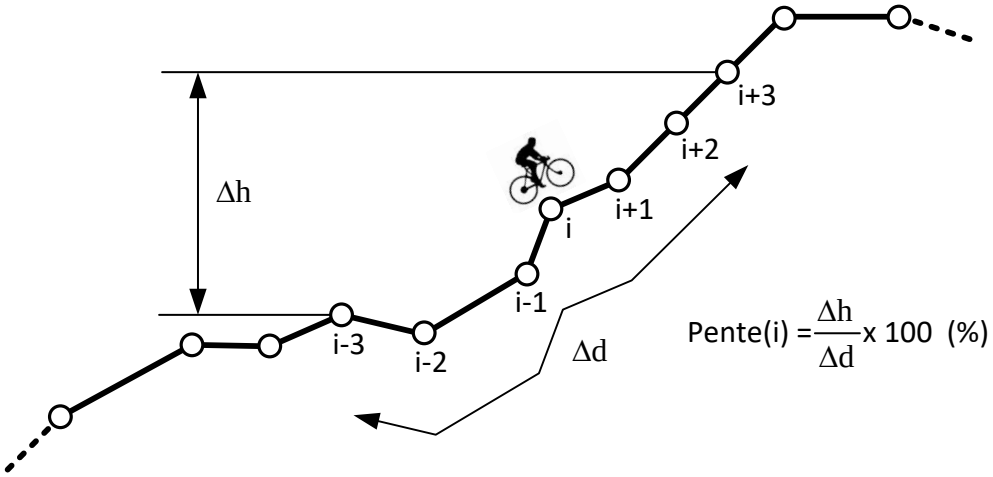
- Altitudes maximale et minimale atteintes
- Ascension totale réalisée : somme de toutes les montées le long du parcours
- Distance totale parcourue : somme de la longueur de chaque segment de la trace. La longueur d'un segment de trace peut être calculée en utilisant les coordonnées GPS des deux points extrémités. Le détail de ce calcul est donné en Annexe B et fait intervenir le rayon de la terre et les angles de latitude et longitude ainsi que les altitudes des points extrémités.

```
--- MENU PRINCIPAL ---
1. Charger la trace d'un fichier de donnees GPS
2. Afficher les donnees de la trace en memoire
3. Analyse de base la trace
4. Calculer la pente maximale
5. Quitter
3

 Analyse de la trace ...

Distance parcourue:    77.295 km
         Ascension:  1368.800 m
 Altitude maximale:  1556.000 m
 Altitude minimale:   789.900 m
```

Exemple d'exécution : analyse de base de la trace du 4 juillet

Calculer

L'option *calculer* permet de réaliser un calcul plus avancée. Dans ce cas, le calcul demandé est celui de la pente maximale de la trace. La pente sera ici définie comme le rapport entre la variation d'altitude et la distance parcourue (Δh/Δd). Ce résultat devrait indiquer la montée la plus raide du parcours. Notons que le simple fait de soulever le GPS sans se déplacer au sol crée une montée très courte et très localisée de pente infinie! Pour éviter ce genre d'aberration, le calcul de la pente en un point doit se faire par une moyenne mobile : en tenant compte d'un certain nombre de segments précédents et suivants ce point. Nous appellerons cet ensemble de segments la **fenêtre de calcul de pente** puisque c'est elle qui définit les données à prendre en compte pour déterminer la pente en un point.



Calcul de pente par moyenne mobile : exemple avec taille de fenêtre égal à 3

Les contraintes suivantes doivent être respectées :

- Le programme doit demander à l'utilisateur la taille de la fenêtre de calcul de pente. Le nombre entré définit le nombre de segments à considérer de chaque côté du point. Par exemple : une valeur de cinq (5) implique un calcul de pente sur un parcours de 10 segments de trace (cinq à droite du point et cinq à gauche du point)
- Le résultat final ne rend compte que de la pente maximale rencontrée

```
--- MENU PRINCIPAL ---
1. Charger la trace d'un fichier de donnees GPS
2. Afficher les donnees de la trace en memoire
3. Analyse de base la trace
4. Calculer la pente maximale
5. Quitter

4

 calcul de la pente maximale ...

Nombre de point pour calcul de pente moyenne: 50

    Pente maximale:     6.116 %
```

Exemple d'exécution : calcul de pente de la trace du 4 juillet

Note : La valeur de pente calculée dépend énormément de la taille de la fenêtre de calcul

Quitter

L'option *quitter* permet de mettre fin au programme.

5. Fichiers fournis

Une archive contenant une solution *Visual Studio* vous est fournie. Elle contient, en particulier, les répertoires et fichiers suivants,:

1. ***Répertoire CSV***
   Répertoire contenant des fichiers de traces en format CSV et devant être lus par votre programme. **Ces fichiers sont nécessaires à la réalisation de votre devoir et doivent se trouver dans le répertoire principal de votre projet pour que leur lecture par le programme *Matrac* soit possible**. Les fichiers fournis ont l'extension *.txt* (exemple : *4juillet.txt*).
2. ***Fichier main.c***
   Programme principal devant être complété
3. ***Fichier matrac_lib.h***
   Prototypes et constantes des fonctions d'analyse de traces GPS. Ce fichier est déjà complété et contient de la documentation importante sur les fonctions du devoir
4. ***Fichier matrac_lib.c***

   Implémentations des fonctions d'analyse de traces GPS. Ce fichier doit être complété en respectant la documentation présente dans le fichier *matrac_lib.h*
5. ***Fichier config.h***

   Contient des définitions de constantes utilisées par le programme.
6. ***Fichiers menu.h et menu.c***

   Prototype et définition de la fonction *afficher_menu* qui affiche le menu principal du programme et lit au clavier le choix de l'utilisateur.

Le dossier ***executable*** contient :

- Le fichier ***Matrac_DevoirTCH009.exe*** qui est la solution du devoir en format exécutable. Il vous permet de tester la solution pour voir le résultat attendu;
- Les fichiers avec l'extension *.txt* (*4juillet.txt*, *23juin.txt*,…) sont des fichiers de données avec lesquels vous pouvez tester le programme.

6. Environnement de programmation et travail demandé

- Vous êtes libres de travailler dans l'environnement de programmation de votre choix
- Vous devez compléter le programme principal (fichier ***main.c***) et définir dans ***matrac_lib.c*** les fonctions déclarées dans ***matrac_lib.h*** de manière à ce que le programme ***Matrac*** fonctionne correctement et tel que décrit dans ce document.
- Vous ne devez pas modifier les fichiers autres que ***main.c*** et ***matrac_lib.c***.

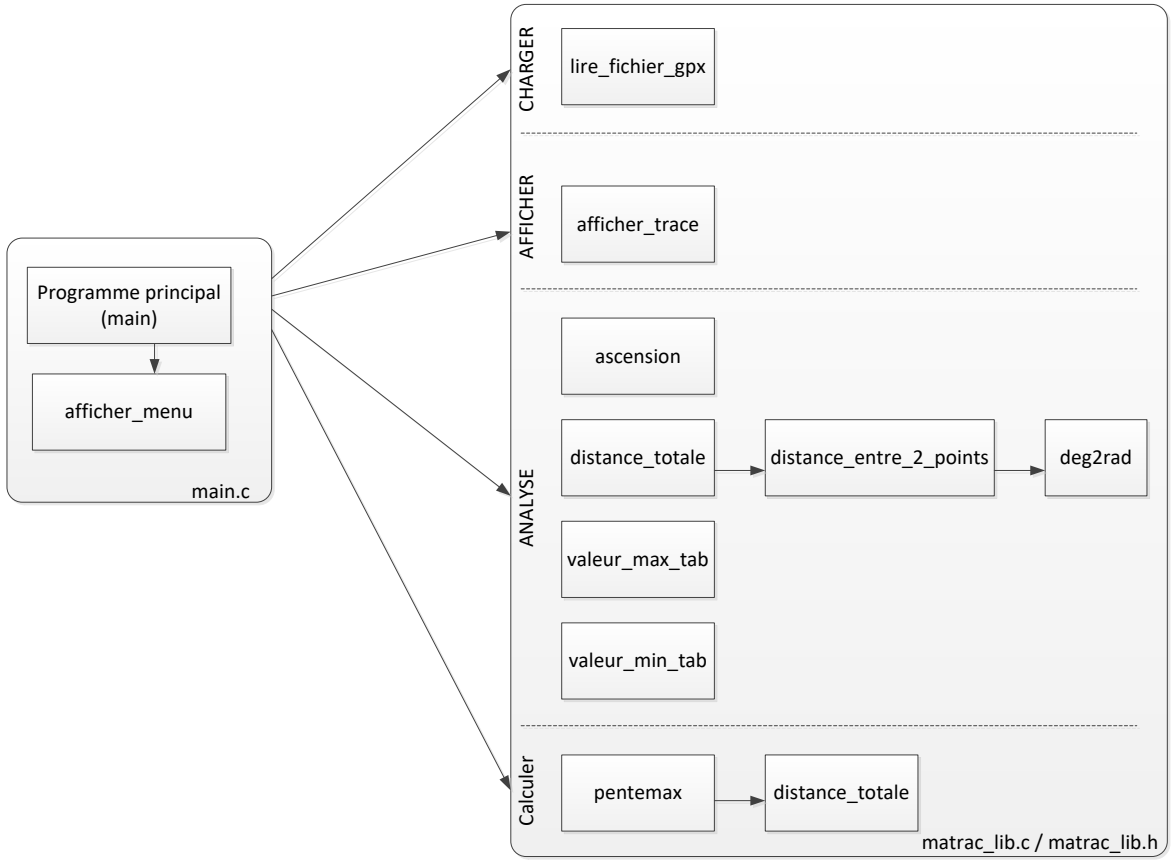
7. Quelques conseils

- Si votre programme principal dépasse 250 lignes de code (incluant aération et commentaires) c'est que votre solution est probablement trop compliquée.
- Mettez une seule instruction par ligne.
- Écrivez du code facile à lire.
- Commentez à mesure que vous écrivez le code.

8. Méthode de travail (fortement) conseillée

Dans le cadre de ce devoir, une approche de développement « bottom-up » est préconisée. Selon cette méthode, le travail se divise en deux parties principales :

1. Développement et test de la librairie de fonctions utiles à l'analyse de trace GPS (*matrac_lib.c* et *matrac_lib.h*)
2. Développement et test du programme principal (*main.c*).



Projet Matrac & librairie matrac_lib

9. Librairie matrac_lib.h / matrac_lib.c

Dans le fichier *matrac_lib.c* écrivez l'implémentation des fonctions de la librairie. Basez-vous sur les prototypes et la documentation spécifique à chacune des fonctions présents dans le fichier *matrac_lib.h*. **Le fichier *matrac_lib.h* ne doit pas être modifié.**

Pour chaque implémentation de fonction complétée, ajoutez dans la fonction *main* quelques appels à la fonction pour vérifier son fonctionnement. Le test d'une fonction devrait couvrir des situations typiquement rencontrées mais également les cas exceptionnels qui doivent être correctement gérés par la fonction. Des cas de test typiques sont donnés ci-dessous pour chaque fonction. Comparez ces résultats avec ceux de vos implémentations pour les valider et éventuellement les corriger.

Fonction *afficher_tab1D_double*

Paramètres ou conditions préalables		Résultat ou vérification à faire
Tableau: {1,2,3,4,5}	Taille: 5	affichage : 1.00 2.00 3.00 4.00 5.00 (sur 5 lignes)
Tableau: {1,2,3,4,5}	Taille: 3	affichage : 1.00 2.00 3.00 (sur 3 lignes)
Tableau: {1,2,3,4,5}	Taille: 0	affichage : aucun
Tableau: {1,2,3,4,5}	Taille: -1	affichage : aucun

Fonction *deg2rad* :

Paramètres ou conditions préalables	Résultat ou vérification à faire
180 degrés	3,14 radians
0 degré	0,00 radian
90 degrés	1,57 radian

Fonction *lire_fichier_gpx* :

Paramètres ou conditions préalables	Résultat ou vérification à faire
"test1.txt"	10
"test2.txt"	2482
"test99.txt" (ou autre nom de fichier inexistant)	-1
"test1.txt" avec *taillemax=4*	4 (seuls les 4 premiers points GPS doivent être chargés dans les tableaux de latitude, longitude et altitude)

Avec la fonction *afficher_tab1D_double*, vérifier que chaque colonne de valeurs du fichier *test1.txt* a bien été chargée en mémoire. Par exemple pour afficher les latitudes: *afficher_tab1D_double(tab_latt,nb_point)*;

Fonction *afficher_trace* :

Paramètres ou conditions préalables	Résultat ou vérification à faire
Charger la trace test1.txt (fonction *charger_trace*) Afficher les données de la trace (*afficher_trace*)	Vérifier que l'affichage correspond au contenu du fichier test1.txt et que la première ligne affichée est une en-tête expliquant le contenu de chaque colonne.
Même test avec taille=0	Affichage : uniquement la ligne d'en tête
Même test avec taille=-1	Affichage : uniquement la ligne d'en tête

Fonction *distance_entre_2_points* :

Paramètres ou conditions préalables	Résultat ou vérification à faire
45.37,75.80,45.38, 75.80,100,100	1.11 (effet de latitude)
45.37,75.80,45.37, 75.81,100,100	0.78 (effet de longitude)
45.37,75.80,45.37, 75.80,100,1000	0.90 (effet d'altitude)
45.37,75.80,45.38, 75.81,100,1000	1.63 (effet combinés)

Fonction *distance_totale* :

Paramètres ou conditions préalables	Résultat ou vérification à faire
Charger test1.txt, début=0, fin=9	0.45 km
Charger test1.txt, début=3, fin=6	0.2 km
Charger test1.txt, début=6, fin=3	0
Charger test1.txt, début=-1, fin=6	0
Charger test1.txt, début=20, fin=6	0
Charger test1.txt, début=3, fin=16	0
Charger test1.txt, début=3, fin=-1	0
Charger test2.txt, début=100, fin=1000	36.35

Fonction *ascension* :

Paramètres ou conditions préalables	Résultat ou vérification à faire
Charger test1.txt, taille=10	2.0 m
Charger test1.txt, taille=0	0
Charger test1.txt, taille=-2	0
Charger test2.txt, taille=2482	1368.8 m

Fonction *valeurmin_tab*:

Paramètres ou conditions préalables	Résultat ou vérification à faire
Charger test1.txt, altitudes, taille=10	Retour 1 et valmin=65
Charger test1.txt, altitudes, taille=0	Retour 0 et valmin=0
Charger test1.txt, altitudes, taille=-2	Retour 0 et valmin=0
Charger test2.txt, altitudes, taille=2482	Retour 1 et valmin=789.9 m

Fonction *valeurmax_tab*:

Paramètres ou conditions préalables	Résultat ou vérification à faire
Charger test1.txt, altitudes, taille=10	Retour 1 et valmin=67
Charger test1.txt, altitudes, taille=0	Retour 0 et valmin=0
Charger test1.txt, altitudes, taille=-2	Retour 0 et valmin =0
Charger test2.txt, altitudes, taille=2482	Retour 1 et valmin=1556.0 m

Fonction *pentemax*:

Paramètres ou conditions préalables	Résultat ou vérification à faire
Charger test1.txt, taille=10, taille_fenetre=3	0.599%
Charger test1.txt, taille=10, taille_fenetre=1	1.194%
Charger test1.txt, taille=10, taille_fenetre=10	0
Charger test1.txt, taille=10, taille_fenetre=-2	0
Charger test1.txt, taille=10, taille_fenetre=0	0
Charger test2.txt, taille=2482, taille_fenetre=50	6.116%

Quand toutes les fonctions ci-dessous sont testées, elles forment une librairie fiable qui peut être utilisée dans un programme d'analyse de traces GPS. Il reste alors simplement à écrire le programme principal faisant appel à ces fonctions.

10. Fichier du programme principal *main.c*

Dans le fichier *main.c* écrivez l'implémentation du programme principal et de toutes autres fonctions que vous jugez nécessaires pour afficher le menu et réaliser les actions reliées à chacun des choix possibles de l'utilisateur en vous basant sur les descriptions de la section 0 de ce document.

11. Contraintes particulières

- Il est interdit d'utiliser l'instruction GOTO, de sortir d'une boucle avec un RETURN, un CONTINUE ou un BREAK;
- Il est interdit d'utiliser des variables globales.

Attention !

Il s'agit d'un travail en équipe. Le plagiat attribuera automatiquement la note 0 à tous les membres de l'équipe, peu importe leur degré d'implication.

Bon travail !

13. Annexe A: lecture de fichiers texte en C

Les types et fonctions de manipulation de fichier décrit ici font partie de la librairie standard `stdio`. L'information ci-dessous est très incomplète. Un tutoriel plus complet peut être trouvé à l'adresse suivante : https://www.tutorialspoint.com/cprogramming/c_file_io.htm

- **Variable faisant référence à un fichier**

  Le type *FILE* est défini par la librairie *stdio* pour faire référence à un fichier.

  Exemple :

```
FILE* fp;      // la variable fp est un pointeur à un fichier
```

- **Ouverture de fichier**

  La fonction `fopen` permet d'ouvrir un fichier pour y lire des informations ou encore y écrire des informations:

```
FILE* fopen(const char* nomDuFichier, const char* modeOuverture);
```

  Exemples :

```
fp=fopen(nom_fichier,"r"); //ouverture pour lecture
```

  Si l'ouverture échoue, `fopen` renvoie la valeur *NULL*.
  Attention l'ouverture du fichier en mode écriture efface tout son contenu !

- **Fermeture de fichier**

  La fonction `fclose` permet de fermer un fichier :

```
FILE* fopen(const char* nomDuFichier, const char* modeOuverture);
```

  Exemples :

```
fp=fopen(nom_fichier,"r");       //ouverture pour lecture
fp=fopen("test.txt","w");  //ouverture pour écriture
```

  Si l'ouverture échoue, `fopen` renvoie la valeur *NULL*.
  Attention l'ouverture du fichier en mode écriture (`"w"`) efface tout son contenu !

- **Lecture de données (plusieurs autres méthodes possibles)**

  La fonction `fgets` permet de lire une ligne du fichier et la mettre en mémoire dans une chaine de caractère :

```
char* fgets(char* chaine, int MaxCar, FILE* pointeurFichier);
```

  Exemples :

```
//copier une ligne du fichier dans chaine (maximum de 100 caractères)
fgets(chaine, 100, fp);
```

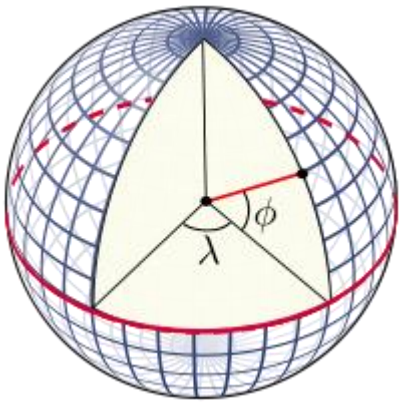
  La fonction `sscanf` permet ensuite d'extraire de la chaîne lue les données voulues.
  Exemple :

```
//extraire deux réels séparés par une virgule
sscanf(chaine,"%f,%f",&val1,&val2);
```

14. Annexe B: Coordonnées géographiques et distances

La position d'un point sur la terre est généralement donnée par deux coordonnées géographiques:

- La latitude (ϕ) est la mesure angulaire par rapport au plan de l'équateur. Elle indique la position nord-sud du point.
- La longitude (λ) est la mesure angulaire par rapport au plan du méridien de Greenwich. Elle indique la position est-ouest du point.



Latitude (ϕ) et Longitude (λ)

Les formules de trigonométrie sphérique permettent de calculer la distance entre 2 points (A et B) sur une sphère de rayon R:

$$d = R \cdot 2 \cdot \operatorname{asin}\left(\sqrt{\sin^2\left(\frac{\phi_A - \phi_B}{2}\right) + \cos(\phi_A) \cdot \cos(\phi_B) \cdot \sin^2\left(\frac{\lambda_A - \lambda_B}{2}\right)}\right)$$

Attention: les fonctions trigonométriques de la librairie *math.h* utilisent le radian comme unité d'angle.

Pour un meilleur positionnement géographique, en plus des informations angulaires de latitude et longitude, on précise souvent l'altitude. Pour les calculs de distance, on peut ajouter la contribution de la variation d'altitude entre les deux points (A et B) en appliquant simplement le théorème de Pythagore:

$$d_{Totale} = \sqrt{d^2 + (h_B - h_A)^2}$$

Le lecteur curieux d'en savoir plus sur ces calculs pourra consulter les références suivantes :

- Orthodromie:
  http://fr.wikipedia.org/wiki/Orthodromie
- Trigonométrie sphérique :
  http://fr.wikipedia.org/wiki/Trigonométrie_sphérique