

## Mini-Projet : Challenge Unesco

Ce mini-projet s'étale sur deux TPs (séances 7 et 8). Il est noté et vous devrez présenter vos résultats et votre code lors d'une soutenance organisée le vendredi 8 janvier 2021. Vous devrez également soumettre votre code **commenté** sur Moodle avant le mercredi 6 janvier 23h59 (heure de Paris).

Le travail est à réaliser en binôme. Vous devez être capable d'expliquer l'ensemble du code, même pour des parties du code sur lesquelles vous n'avez pas travaillé.

L'ensemble des informations et fichiers nécessaires à ce mini-projet peuvent être récupérés sur Moodle.

### 1 Présentation

Vous êtes un fan de voyage, c'est la fin de l'année scolaire et vous avez trois semaines de vacances devant vous ! Vous décidez de saisir cette opportunité pour réaliser votre rêve : visiter les endroits les plus incroyables de la terre. Votre lieu de départ est connu sous forme LAT/LONG et vous disposez d'un hélicoptère parfait (y compris le pilote) qui voyage à la vitesse constante de 85km/h et possède un réservoir illimité. Vous décidez de vous limiter à la visite de sites UNESCO inscrits au patrimoine de l'humanité, qui sont de trois types : endroits culturels (tels que les centres historiques), les endroits naturels (tels que les parcs nationaux) et les endroits mixtes qui représentent les deux. Vous décidez que vous voulez visiter autant de sites que possible, avec un nombre égal (à un près) de sites culturels et naturels (les sites mixtes comptent aussi bien pour naturel que pour culturel).

Le challenge est de trouver un itinéraire, partant de n'importe quel endroit LAT/LONG, qui va vous permettre de visiter autant de sites que possible, et revenir à votre point de départ. Le temps maximum de votre voyage est de trois semaines, et vous supposez que vous allez passer 7 heures par site, comprenant la visite, les repas/boissons et le repos (que vous pourrez aussi réaliser dans l'hélicoptère). Vous évaluez la qualité de votre itinéraire de la façon suivante :

- Chaque site UNESCO visité compte pour 10 points.
- Chaque pays différent visité compte pour 20 points
- Chaque site qui est listé comme "en danger" rapporte 30 points en plus.

Votre but est de trouver un itinéraire permettant d'obtenir le score le plus élevé.

Lors de la soutenance, un point de départ sera aléatoirement choisi et les trois binômes ayant trouvés les meilleurs scores seront récompensés.

#### Contraintes :

Le programme doit accepter en argument les coordonnées LAT et LONG du point départ (par exemple `./main 48.8464111 2.3548468`, si vous partez de la place Jussieu).

En sortie du programme, vous devez afficher l'itinéraire de voyage, y compris les noms des sites, leurs types (culturel, naturel ou mixte) et le pays, ainsi que l'évaluation finale de l'itinéraire. Vous devez donc obtenir un résultat semblable à celui-ci :

```
thibaut@ThibautPortable:~/Documents/Cours/2020-2021/InfoGen/Projet2020/ProjetR0B3/Code2020$ ./main 48.8464111 2.3548468
Challenge Unesco
Temps total = 501.654144
Nombre de sites naturels = 14
Nombre de sites culturels = 13
Nombre de sites mixtes = 1
ITINERAIRE DE VOYAGE :
0) Depart 48.85 2.35 Unknown 0
1) Paris Banks of the Seine 48.86 2.29 Cultural France 0
2) Dorset and East Devon Coast 50.71 -2.99 Natural United Kingdom of Great Britain and Northern Ireland 0
3) City of Bath 51.38 -2.36 Cultural United Kingdom of Great Britain and Northern Ireland 0
4) Giant's Causeway and Causeway Coast 55.25 -6.49 Natural United Kingdom of Great Britain and Northern Ireland 0
5) Brú na Bóinne - Archaeological Ensemble of the Bend of the Boyne 53.69 -6.45 Cultural Ireland 0
6) St Kilda 57.82 -8.58 Mixed United Kingdom of Great Britain and Northern Ireland 0
7) Surtsey 63.30 -20.60 Natural Iceland 0
8) Pingvellir National Park 64.25 -21.04 Cultural Iceland 0
9) Ilulissat Icefjord 69.13 -49.50 Natural Denmark 0
10) Red Bay Basque Whaling Station 51.73 -56.43 Cultural Canada 0
11) Gros Morne National Park 49.61 -57.53 Natural Canada 0
12) L'Anse aux Meadows National Historic Site 51.47 -55.62 Cultural Canada 0
13) Mistaken Point 46.63 -53.21 Natural Canada 0
14) Landscape of Grand Pré 45.12 -64.31 Cultural Canada 0
15) Joggins Fossil Cliffs 45.71 -64.44 Natural Canada 0
16) Old Town Lunenburg 44.38 -64.31 Cultural Canada 0
17) Miguasha National Park 48.10 -66.35 Natural Canada 0
18) Historic District of Old Québec 46.81 -71.21 Cultural Canada 0
19) Great Smoky Mountains National Park 35.59 -83.44 Natural United States of America 0
20) Monticello and the University of Virginia in Charlottesville 38.03 -78.50 Cultural United States of America 0
21) Mammoth Cave National Park 37.19 -86.10 Natural United States of America 0
22) Cahokia Mounds State Historic Site 38.66 -90.06 Cultural United States of America 0
23) Carlsbad Caverns National Park 32.17 -104.38 Natural United States of America 0
24) Archaeological Zone of Paquimé Casas Grandes 30.38 -107.96 Cultural Mexico 0
25) Islands and Protected Areas of the Gulf of California 27.63 -112.55 Natural Mexico 0
26) Rock Paintings of the Sierra de San Francisco 27.66 -112.92 Cultural Mexico 0
27) Whale Sanctuary of El Vizcaino 27.79 -114.23 Natural Mexico 0
28) Chaco Culture 36.06 -107.97 Cultural United States of America 0
29) Grand Canyon National Park 36.10 -112.09 Natural United States of America 0
30) Depart 48.85 2.35 Unknown 0
Evaluation :
    29 sites visités * 10 points = 290 points
    8 pays visités * 20 points = 160 points
    0 sites en danger visités * 30 points = 0 points
Score total = 450
```

## 2 Lecture des données et génération d'un premier itinéraire

Il y a plus de 1000 sites Unesco dans le monde, qui sont représentés sur l'image ci-dessous.



Les données relatives aux différents sites sont données dans un fichier joint au projet (fichier "sites.csv"). Dans ce fichier, vous trouverez pour chaque site :

- Nom
- Latitude
- Longitude
- Catégorie (Cultural, Natural ou Mixed)
- Pays
- Continent
- En danger ou pas sous forme booléenne (1 si le site est en danger, 0 sinon)

---

### Exercice 1 – Lecture des données

---

#### Q 1.1 Structure de données associée à un site Unesco

Pour sauvegarder les informations liées à un site Unesco, vous pourrez utiliser la structure suivante, que vous créerez dans un fichier `site.h`.

```
1 typedef struct site{
2     char* nom;
3     float LAT;
4     float LONG;
5     char* categorie; //cultural,natural,mixed
6     char* pays;
7     int enDanger; //0,1
8 } Site;
```

Dans un fichier `site.c` ajouter une méthode de signature `Site construireSite(char* nom,float`

LAT,float LONG,char\* categorie,char\* pays,int enDanger); permettant de construire une variable de type `Site` à partir des informations données en arguments.

Ajouter également une méthode `void affichageSite(Site s);` permettant d'afficher les caractéristiques d'un site.

Tester et valider vos méthodes sur de petits exemples.

### Q 1.2 Lecture des données du fichier "sites.csv"

Nous vous conseillons ensuite de construire un tableau de `Site` de taille égale au nombre de sites présents dans le fichier.

Chaque case du tableau contiendra les sites présents dans le fichier "sites.csv". Pour lire ce fichier, vous pouvez vous aider des différentes fonctions présentes dans les fichiers "lectureFichiers.c" et "lectureFichiers.h" à télécharger sur Moodle. La fonction `GetChaine()` (lire la prochaine chaîne de caractères avant une virgule ou une fin de ligne) est particulièrement utile. Par lire des entiers et réels, vous pouvez utiliser la fonction `fscanf()`.

---

## Exercice 2 – Distance entre sites

---

Pour calculer la distance entre deux points du globe terrestre, vous utiliserez la formule de Haversine ([https://fr.wikipedia.org/wiki/Formule\\_de\\_haversine](https://fr.wikipedia.org/wiki/Formule_de_haversine)). L'implémentation de cette méthode vous est donnée ci-dessous et dans les fichiers "harversine.c" et "haversine.h" téléchargeables sur Moodle.

```
1  #include <math.h>
2  #include "haversine.h"
3
4  double toRad(double angleD){
5      return (angleD * M_PI / 180.0);
6  }
7
8  double haversin(double val){
9      return pow(sin(val / 2), 2);
10 }
11
12 double calculDistance(double nLat1, double nLon1, double nLat2, double nLon2){
13     double dLat = toRad(nLat2 - nLat1);
14     double dLong = toRad(nLon2 - nLon1);
15     double startLat = toRad(nLat1);
16     double endLat = toRad(nLat2);
17     double a = haversin(dLat) + cos(startLat) * cos(endLat) * haversin(dLong);
18     double c = 2 * atan2(sqrt(a), sqrt(1 - a));
19     return RAYON_TERRE * c;
20 }
```

### Q 2.1 Validation du calcul de distance

Vérifier que vous obtenez bien une distance de 182.27 km entre les points (50,3) et (49,5) et une distance de 18806.70 km entre les points (-85.7,-170.6) et (83.2,165.3).

### Q 2.2 Sauvegarde des distances

Créer une matrice de distances que vous remplirez avec les distances entre les différents sites. Notez qu'il est également utile de sauvegarder les distances entre le point de départ et les différents sites.

---

**Exercice 3 – Génération d'un premier itinéraire**

---

Un itinéraire est composé d'une suite de sites à visiter. Le nombre de sites à visiter n'étant pas fixé, il vous est demandé d'utiliser une liste chaînée pour sauvegarder les informations d'un itinéraire.

**Q 3.1** Gestion de la liste chaînée

Ajouter les fonctions permettant de créer une cellule (ou "maillon") de votre liste chaînée, d'insérer un élément en tête, de désallouer l'espace mémoire occupé par la liste, d'afficher la liste, etc.

Tester et valider vos méthodes sur de petits exemples.

**Q 3.2** Méthode du plus proche voisin

Implémenter une première méthode permettant d'obtenir un itinéraire rapidement, basée sur l'algorithme du "plus proche voisin". Cette méthode fonctionne de la manière suivante : on construit un itinéraire en choisissant à chaque fois le site le plus proche du site que l'on est train de visiter. Attention, différentes conditions seront à vérifier :

- On doit toujours être capable de revenir à notre point de départ (c'est-à-dire avoir assez de temps pour retourner à notre point de départ après la visite d'un site ; le temps total de l'itinéraire étant limité à 3 semaines)
- On doit visiter autant de sites culturels que naturels (à un près). Une façon simple de gérer cette contrainte est d'alterner les sites naturels/culturels (ou mixtes car les sites mixtes comptent pour naturels et culturels)
- Il faut bien sûr éviter de visiter un site deux fois

**Q 3.3** Affichage de la solution obtenue

Afficher l'itinéraire obtenu avec cette méthode, et vérifier qu'il respecte bien les contraintes du problème.

**Q 3.4** Calcul du score

Calculer et afficher le score obtenu avec cette méthode.

---

**Exercice 4 – Visualisation d'un itinéraire**

---

Il peut être pratique de visualiser (sur une carte) un itinéraire.

Pour cela, une petite application écrite en Java est disponible sur Moodle. Cette application a besoin en entrée d'un fichier nommé "Tour.txt" contenant les coordonnées des différents sites de l'itinéraire, du point de départ et du point d'arrivée (égal au point de départ). Par exemple :

*Tour.txt* :

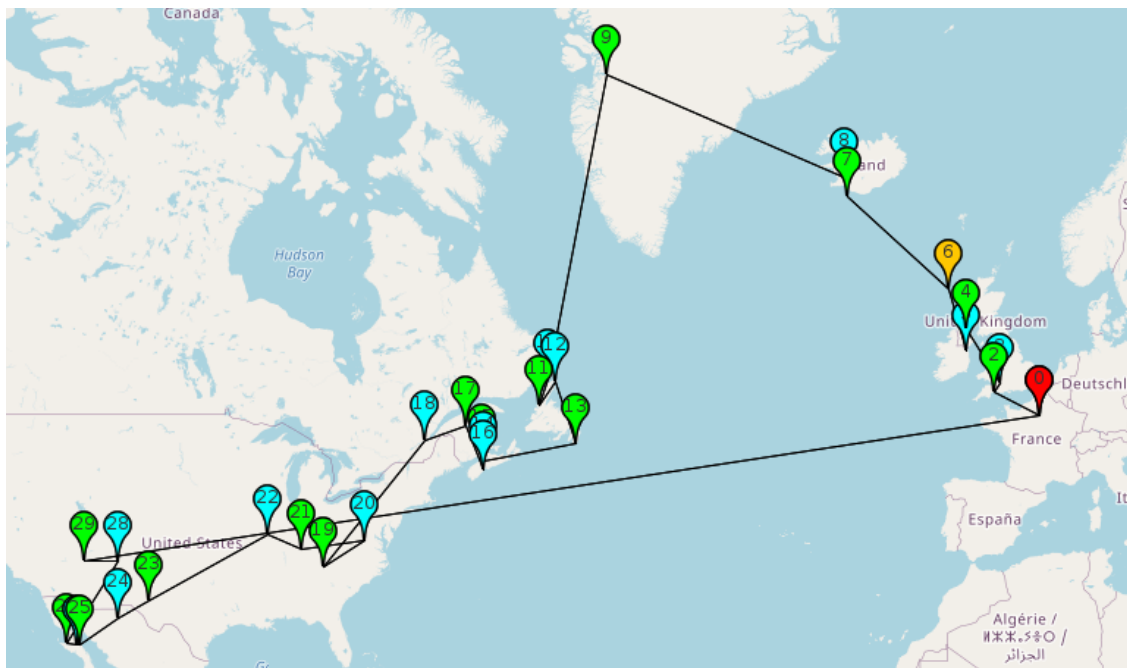
```
1 48.846413, 2.354847
2 48.858334, 2.294167, Cultural
3 50.705555, -2.989889, Natural
4 51.381390, -2.358611, Cultural
5 55.250000, -6.485278, Natural
6 48.846413, 2.354847
```

Pour lancer cette application à partir d'un code C, il suffit d'ajouter les instructions suivantes :

```
1 const char* commandLine="java -jar _UnescoMap.jar";
2 system(commandLine);
```

Vous pourrez ainsi créer une fonction d'affichage d'itinéraire où vous créerez dans un premier temps le fichier "Tour.txt", et ensuite lancerez l'application de visualisation.

Pour la méthode du plus proche voisin, en partant de la place Jussieu, vous pourrez obtenir un itinéraire ressemblant à cela :






Les points verts correspondent à des sites naturels, bleus à des sites culturels et oranges à des sites mixtes.

Pour info, le score obtenu par cette solution est de 450 (29 sites visités, 8 pays visités, 0 site en danger visité), mais il est possible de faire bien mieux !

### Remarque :

Pour pouvoir lancer l'application de visualisation, il est nécessaire que les dossiers "images" et "lib" téléchargeables sur Moodle soient dans le même dossier que l'application `UnescoMap.jar`, comme illustré sur la figure ci-dessous :

| Nom   |
|---|
|  images        |
|  lib           |
|  UnescoMap.jar |

Il est aussi nécessaire d'être connecté à Internet pour que l'affichage de la carte fonctionne correctement.

## 3 Optimisation : A vous de jouer !

Vous remarquerez que dans l'algorithme du plus proche voisin, on ne tient pas compte du fait que les sites en danger rapportent 30 points en plus et que les pays visités rapportent 20 points.

La séquence est également loin d'être optimale, puisque des croisements entre routes reliant les sites sont présents.

De plus, avec cette méthode, on a tendance à s'éloigner du point de départ, sans tenir compte du fait qu'il faudra y revenir à la fin du tour.

Des adaptations sont donc nécessaires pour augmenter le score de l'itinéraire.

Attention : votre méthode doit générer rapidement un itinéraire (c'est-à-dire en moins de 30 secondes).

## 4 Conseils

- Décomposer votre code en différents fichiers, et utilisez un `makefile` pour compiler l'ensemble des fichiers
- Tester et valider chaque nouvel élément de votre code avant d'aller plus loin
- Commencer par implémenter des méthodes simples pour trouver un itinéraire, que vous pourrez améliorer par la suite
- Avant de sortir de la fonction principale, n'oubliez pas de désallouer la mémoire qui a été allouée dynamiquement (pour chaque `malloc`, un `free`!)
- En cas de bogue (ce qui risque d'arriver!), n'hésitez pas à utiliser les outils de débogage comme `gdb`, `ddd`, ou `valgrind` vus au cours.
- N'oubliez pas de commenter votre code