# Projet LO21

## Modalités

Le projet doit être effectué **seul ou en binôme**.
Une soutenance (sans transparents) sera réalisée pendant les séances de TP.
**Les éléments à fournir :**
Un rapport, le code source des programmes, les scripts d'exécution, le makefile ou script de compilation.

## Objectif

Le projet a pour objectif la définition et la manipulation de types abstraits de données Individu et Population. Un Individu est représenté par une suite de bits et une Population est une suite d'Individus. **Il s'agit d'une version simplifiée d'algorithmes génétiques, constituant une des approches de résolution de problèmes d'optimisation.**

## 1. Type abstrait « Individu »

Définir un type abstrait de données appelé Individu, représenté par une liste de bits, de longueur donnée (longIndiv).
Écrire l'algorithme et le sous-programme C correspondant aux opérations suivantes :
• Initialiser aléatoirement la liste de bits (donner une version itérative et une version récursive de cette opération).
• Décoder la liste de bits et donner la valeur entière correspondante.
• Croiser deux listes de bits, c'est à dire intervertir les éléments des deux listes selon une probabilité donnée (pCroise) pour chaque position dans la liste (tirage aléatoire et comparaison avec la probabilité).
• Calculer à partir de la valeur d'un individu sa qualité, en utilisant la fonction réelle f1 donnée ci-dessous:

$f_1(x) = -X^2$

avec : $X = (x / 2^{longIndiv}) * (B-A) + A$, $A = -1$, $B = 1$, $longIndiv = 8$

Au niveau de l'implémentation en C, on utilisera pour représenter un bit la définition de type suivante : « **`typedef unsigned char Bit;`** ». Ainsi, une variable de type Bit prendra les valeurs **`0`** ou **`1`**. De plus, une suite de bits sera représentée sous forme d'une liste chaînée. Pour la génération de nombres aléatoires, on utilisera les fonctions **`rand`**, **`srand`** et **`time`** des bibliothèques **`stdlib`** et **`time`**.

## 2. Type abstrait « Population »

Définir un type abstrait de données appelé Population, constitué d'une liste d'Individus. Écrire l'algorithme et le sous-programme C correspondant aux opérations suivantes :
• Initialiser de manière aléatoire la liste d'Individus.
• Trier la liste par Qualité décroissante des Individus au moyen de Quicksort (voir ci-dessous). On utilisera pour cela la fonction Qualité définie sur le type abstrait Individu.
• Sélectionner les meilleurs Individus de la Population en tronquant la liste et en la complétant par recopie des tSelect premiers éléments.
Par exemple : tselect = 4
Qualité des Individus :

| | | | | | | |
|---|---|---|---|---|---|---|
| 7 | 6 | 5 | 4 | 3 | 2 | 1 |

Liste après sélection :

| | | | | | | |
|---|---|---|---|---|---|---|
| 7 | 6 | 5 | 4 | 7 | 6 | 5 |

• Croiser la Population, c'est à dire à partir d'une Population P1, créer une seconde Population P2, constituée d'Individus sélectionnés aléatoirement deux à deux dans P1 et croisés entre eux.
Au niveau de l'implémentation en C, une suite d'Individus sera représentée sous forme de liste chaînée.

## 3. Quicksort

Le tri rapide, ou Quicksort, est fondé sur le principe « diviser pour mieux régner ». Il est donc composé des deux étapes suivantes :

• Diviser : la liste des éléments à trier est partitionnée en deux sous listes S1 et S2, telles que chaque élément de S1 soit inférieur à tout élément de S2. On procède en inversant lorsque c'est nécessaire les extrémités de la liste.

• Régner : les deux sous-listes sont triées par appels récursifs à Quicksort. Les deux sous listes étant individuellement triées et les éléments de la première inférieurs à ceux de la seconde, la liste est donc globalement triée.

## 4. Programme

En utilisant les précédentes définitions, réaliser le programme correspondant à l'algorithme ci-dessous:

```
Initialiser la Population
Répéter nGen fois
Début
Croiser la Population
Trier la Population
Sélectionner la Population
Fin
Afficher le meilleur Individu de la Population
```

Les paramètres à utiliser sont les suivants :
-Longueur d'un individu : longIndiv = 8
-Probabilité de croisement : pCroise = 0.5
-Taille de la Population : 20 ≤ TaillePop ≤ 200
-Taux de sélection : 10% ≤ tSelect ≤ 90%
-Nombre de générations : 20 ≤ nGen ≤ 200

## 5. Manipulation

Une fois les types de données et leurs opérations définis et le programme réalisé, tester celui-ci en remplaçant la fonction Qualité f1 par la fonction suivante :

• $f_2(x) = -\ln(X)$

avec : $X = (x / 2^{longIndiv}) * (B - A) + A,\ A = 0.1,\ B = 5,\ longIndiv = 16$

On précise que l'on réalise préalablement au calcul de la fonction en lui même, une mise à l'échelle, au moyen de X.

## 5. Rapport

Le rapport doit contenir les éléments suivants :

• Description des choix de conception et d'implémentation relatifs aux structures de données utilisées et à la démarche adoptée.
• Algorithmes des sous-programmes (en utilisant des opérations abstraites) et leurs explications.
• Jeux d'essais.
• Commentaires sur les résultats.

Le rapport ne devra pas contenir de listing du programme C.

## 6. Remarques

- Un programme C et du pseudo-code ne sont pas des algorithmes.
- Un algorithme nécessite de donner son profil (données, résultat, lexique).
- Un programme doit être commenté (de façon utile).
- La qualité d'implémentation est prise en compte.
- Faites simple.