Aide-mémoire

Boucles:

for

```
for (initialisation; condition; incrémentation) {
    // instructions à exécuter à chaque itération
}
```

while

```
while (condition) {
    // instructions à répéter
}
```

do...while

```
do {
    // Instructions à exécuter
} while (condition);
```

Switch:

```
char choix;
...

switch (choix) {
case '1':
    option1();
    break;
case '2':
    option2();
    break;
case '3':
    cout << "Au revoir!" << endl;
    break;
default:
    cout << "Choix invalide..." << endl;
    break;
}
```

Fonctions

- une fonction sans retour utilise le mot réservé `void` et n'a pas de `return`

Fonctions sans paramètres:

```
[type de retour] maFonction1(){
    //instructions
    return ... ; //si nécessaire
}
```

Fonctions avec paramètres

(ajouter une & entre le type de param et le nom de celui-ci pour passer par référence)

```
[type de retour] maFonction2([type de param1] nomParam1, [type de param2]
nomParam2, ...){
    //instructions
    return ...; //si nécessaire
}
```

avec tableau classique:

```
[type de retour] maFonction6([type de tableau]  tableau[], int taille, [autre
param] nomParam, ... ){
    //instructions
    return ...; //si nécessaire
}
```

Struct

```
//définition
struct T_NomStruct {
    Type1 nomMembre1;
    Type2 nomMembre2;
    // ...
};

//déclaration
T_NomStruct maStruct = {champ1, champ2, ...};

//accession aux valeurs dans l'instance de la struct:
maStruct.nomMembre1
```

```
maStruct.nomMembre2
...
```

Valeurs aléatoires

```
#include <time.h>   ou   <ctime>
...
srand(time(0)); //Pour utiliser le temps actuel pour générer le seed
int n = 5; //le nombre max inclus dans le tirage au sort
int nombreAleatoire = rand() % (n+1);
```

Fichiers externes

```
#include <iostream>
#include <fstream>
#include <string>
```

Lecture d'un fichier texte

```
fstream fichier;
fichier.open("nomFichier.txt", ios::in);

if (!fichier) {
    cout << "Erreur : Impossible d'ouvrir le fichier." << endl;
    return false;  //ou exit(1) selon le contexte
}
//instructions
fichier.close();
return true; // ne rien mettre si void
```

- Lecture ligne par ligne:

```
while (getline(fichier, ligne)) { // Lit le fichier ligne par ligne
    cout << ligne << endl; //afficher les lignes en console
}
```

- lecture mot par mot:

```
string mot;
while (fichier >> mot) { // Lit le fichier mot par mot
    cout << mot << endl; //afficher les mots en console
}
```

- lecture type par type:

```
int nombre;
string mot;
float valeur;
while (fichier >> nombre >> mot >> valeur) {
    cout << "nombre: " << nombre << endl;
    cout << "mot: " << mot << endl;
    cout << "valeur: "<< valeur << endl;
}

if (fichier.fail() && !fichier.eof()) {
    cout << "Erreur de lecture dans le fichier."<< endl;
    return false;
}
```

Écriture dans un fichier texte

```
fstream fichier;
fichier.open("nom_du_fichier.txt", ios::out || ios::app);

    if (!fichier) {
        cout << "Erreur lors de l'ouverture du fichier." << endl;
        return false; //ou bien exit(1) selon le type de retour
    }

    fichier << "Bonjour, ceci est une ligne de texte." << endl;

    fichier.close();
    return true;
```

Fichier passé à une fonction

- créer une fonction qui prend un paramètre de type fstream
- dans une autre fonction, ouvrir le fichier
- passer l'objet fstream en paramètre lors de l'appel de la fonction créée à l'étape 1.