Syntaxe du C
Aide-mémoire

MP2I Lycée Pierre de Fermat

Instructions basiques

Déclaration de variable

```c
float x;
int n, m, i, j;
/* Le type bool est un renommage du type des entiers 8 bit.
   Il faut la librairie stdbool.h pour l'utiliser. */
bool t;
```

Définition de variable

```c
x = 25.3;
n = x + 2;
t = true;
// on peut mélanger déclaration et définition:
bool b = (x > 20);
float y = 0, z = 8.3;
```

Opérations arithmétiques `+, x, *, -, %`

```c
n = ((8+7)*((65-4)/3))%2;
```

Opérateurs de comparaison `<, >, <=, >=, ==, !=`

```c
int x = 5;
int y = 11;
bool b1 = x < y;
bool b2 = x + 8 != y;
```

Opérateurs booléens `!, &&, ||`

```c
int x = 5;
int y = 11;
bool b1 = (x < y) && (y < 2 * x); // ET
bool b2 = (x == y) || (x > y+5); //  OU
bool b3 = !b2; // NEGATION

// On peut tout combiner comme on veut:
bool b4 = (b1 && !b2) || (x == 1) || !b3 ;
```

Affichage et saisie

Les fonctions d'affichage/saisie nécessitent l'usage de la librairie ***stdio.h***. Vous devez l'importer au début des programmes avec `#include <stdio.h>`.

Affichage

```c
printf("Bonjour\n"); // \n est un retour à la ligne

int x = 85;
float z = -6.978;
printf("%f est un flottant, %d un entier\n", z, x);
printf("%d + 1 vaut %d\n", x, x+1);

// Une valeur sera affichée différemment selon le type qui est indiqué.
printf("%c est le %d-eme caractère.\n", x, x);
```

Saisie

```c
int x, y;
float z;
scanf("%d %d", &x, &y);
scanf("%f", &z);
```

Temps et aléatoire

La fonction time : Elle renvoie le nombre de secondes écoulées depuis le 01/01/1970 à minuit.

```c
#include <time.h>
int main(){
  long int t = time(NULL);
  printf("%ld\n", t);
}
```

Génération d'aléatoire : Initialisation avec `srand` **une fois au début du main**.

```c
#include <time.h>   // pour la fonction time
#include <stdlib.h> // pour les fonctions srand et rand
#include <stdio.h>

// renvoie un nombre aléatoire entre 10 et 20 inclus
int note(){
  return 10 + rand()%11;
}

int main(){
  srand(time(NULL)); // initialisation du générateur alétatoire
  int x = note();
  int y = note();
  printf("%d, %d\n", x, y);
}
```

Conditions et boucles

If-else

```c
int x = 3;
float y = 5.75;
if (x > y){
  y = 2*y;
} else {
  y = y/2;
}
// On peut faire un if sans else:
bool b = (x == 3);
if (b){
  printf("Truc\n");
}
```

Boucles while

```c
int i = 3, j = 985;
while (i != j && j > 0){
  i++;
  j--;
}
```

Boucles for

```c
for (int i=7; i<98; i = i + 3){
  printf("%d\n", i);
}

/* Souvent, l'indice de boucle augmente de 1 en 1. L'instruction i++
   permet d'incrémenter la variable i de 1, on écrira souvent les
   boucles comme suit:  */
for (int i = 0; i < n; i++){
  ...
}

/* Les boucles for peuvent être utilisées de très nombreuses manières.
   La boucle suivante affiche toutes les lettres minuscules. */
for (char c = 'a'; c <= 'z'; c++){
  printf("%c", c);
}

/* En C, les boucles for sont aussi puissantes
   que les boucles while (ce qui n'est pas le cas
   en algorithmique). N'utilisez pas les boucles
   for du C pour faire des boucles while !
   Exemple à ne pas reproduire chez soi: */
for ( ; ; ){
   // cette boucle tourne à l'infini sans rien faire
}
```

Fonctions

Déclaration de fonction

```
bool est_pair(int x);
float somme(float x, float y);
```

Définition de fonction

```
bool est_pair(int x);
bool est_pair(int x){
  return (x%2 == 0);
}

// Comme pour les variables on peut fusionner déclaration et définition

float somme(float x, float y){
  return x+y;
}
```

Appel de fonction

```
float somme(float x, float y){
  return x+y;
}

int main(){
  float x = -5;
  float y = 6.22;
  float z = somme(x, y);
}
```

Assertions

```
#include <assert.h>

float inverse(float x){
  assert (x != 0);
  return 1/x;
}

int main(){
  float x = 5, y = 0;
  x = inverse(x); // pas de souci
  y = inverse(y); // Le programme affiche une erreur d'assertion et s'arrête

  return 0;
}
```