Remise à niveau en C

I Syntaxes de base

Ce document est un inventaire succins des syntaxes de base du language C.

1 Variables

```c
/*
Une variable possède un type et doit toujours être déclaré avant de pouvoir
être utilisée.
Quelques type usuels:
*/

unsigned int a = 5; // entier positif stocké sur 4 octets
int b = -5; // entier relatif stocké sur 4 octets
float c = 1.32f; // flottant stocké sur 4 octets
double d = 1.32; // flottant stocké sur 8 octets
char e = 'a'; // caractère
bool f = true; // booléen
```

2 Opérateurs

- Opérateurs usuels $=, +, -, *, /$
- Incrémentation : `a++;` équivalent à `a= a+1;`
- Modulo : % renvois le reste de la division euclidienne de a par b
- Comparaisons : $>, >=, <, <=, ==, ! =$
- Opérations logiques :
  - && : "et" logique
  - || : "ou" logique
  - ! : négation

3 Branchement et boucles

Branchements :

```
int a = 5;
int b = 2;

if(a >= b){
  printf("a plus grand que b\n");
}
else{
  printf("a plus petit que b\n");
}
```

Boucle while :

```
int a = 5;
int b = 2;

// on décrémente tant que a>=b
while(a>=b){

  printf("a=%d\n",a);
  a--;
}
printf("a=%d\n",a);
```

Boucle for :

```
int a = 0.5f;

// on décrémente a 5 fois
for(int i=0;i<5;i++){

  printf("a=%d",a);
  a--;
}
printf("a=%d",a);
```

4 Tableaux - allocation statique

```c
#include <stdio.h>

int main(){

  int T[10] = {}; // initialisation d'un tableau à 10 élément:
  // allocation statique, donc la taille est fixée
  // à l'écriture du programme

  // affichage de chaques éléments
  for (int i = 0; i < 10; ++i) {

    printf("T[%d] = %d\n", i, T[i]);
  }

  return 0;
}
```

5 Fonctions

```c
#include <stdio.h>

// declaration d'une fonction:
int somme(int a, int b, int c) // entête
{
  a = a + b + c;
  return a;
}

int main(){
  int a=1, b=2, c=3;
  printf("valeurs initiale:\n");
  printf("a=%d b=%d c=%d\n", a, b, c);
  printf("\n");

  int r = somme(a,b,c);

  printf("somme=%d\n", r);
  printf("valeurs finales :\n");
  printf("a=%d b=%d c=%d\n", a, b, c);

  return 0;
}
```

6 Structures

```c
#include <stdio.h>

// definition d'une structure représentant un vecteur
struct Vector3{
    float x;
    float y;
    float z;
};

int main(){
    struct Vector3 v = {1.f, 2.f, 3.f}; // initialisation

    printf("v = {%f, %f, %f}", v.x, v.y, v.z); // on accède aux champs avec "."

    struct Vector3* p = &v;
    p->x = 3.f; // si on passe par un pointeur, on remplace "." par "->"

    printf("v = {%f, %f, %f}", v.x, v.y, v.z);

    return 0;
}
```