# Liste d'exercices 3 : les boucles et le parcours de vecteurs (1 séance)

## Notions à considérer pour pouvoir faire les exercices

### Construction des boucles for et while :

Les boucles for et while sont similaires en assembleur.

Exemple :

```
int i;
```

| *La boucle for en C* | *La boucle while en C* | *Assembleur* |
|---|---|---|
| `for (i = 0; i < 5; i++)`<br>`{`<br>`  // contenu de la boucle`<br>`}` | `i = 0;`<br><br>`while (i < 5)`<br>`{`<br>`  // contenu de la boucle`<br><br>`  i++;`<br>`}` | `  mov i, 0`<br><br>`debutboucle:`<br>`  cmp i, 5`<br>`  jnl finboucle`<br><br>`  // contenu de la boucle`<br><br>`  inc i`<br><br>`  jmp debutboucle`<br><br>`finboucle:` |

Exemple : afficher les lettres minuscules depuis la lettre z jusque la lettre a.

```
char c;
const char msg[] = "c = %c\n";
```

| *La boucle for en C* | *La boucle while en C* | *Assembleur* |
|---|---|---|
| `for (c = 'z'; c >= 'a'; c--)`<br>`  printf(&msg[0], c);` | `c = 'z';`<br><br>`while (c >= 'a')`<br>`{`<br>`  printf(&msg[0], c);`<br><br>`  c--;`<br>`}` | `  mov  c, 'z'`<br><br>`debutboucle:`<br>`  cmp  c, 'a'`<br>`  jnge  finboucle`<br><br>`  movsx eax, c`<br>`  push  eax`<br>`  push  offset msg`<br>`  call  dword ptr printf`<br>`  add  esp, 8`<br><br>`  dec  c`<br>`  jmp  debutboucle`<br><br>`finboucle:` |

En C, comme en assembleur, on peut écrire directement un caractère entre apostrophes plutôt que sa valeur dans la table ASCII.

Un caractère est stocké dans une variable du type char. Cependant, la fonction printf attend que

toute valeur entière transmise en paramètre soit codée sur 4 octets. Ceci explique pourquoi l'instruction MOVSX EAX, c est utilisée dans le code en assembleur pour étendre à 4 octets la valeur numérique représentant le caractère.

## La table ASCII :

L'ASCII (American Standard Code for Information Interchange) est un système de codage qui associe aux caractères de la langue anglaise ainsi qu'à certains symboles (&, @, ?, #, ...) une valeur binaire sur 7 bits.

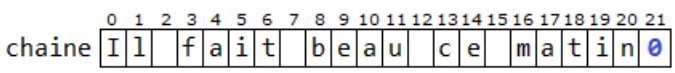
| *Déc* | *Hex* | *Car* | *Déc* | *Hex* | *Car* | *Déc* | *Hex* | *Car* | *Déc* | *Hex* | *Car* | *Déc* | *Hex* | *Car* |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 32 | 20 | SP | 51 | 33 | 3 | 70 | 46 | F | 89 | 59 | Y | 108 | 6C | l |
| 33 | 21 | ! | 52 | 34 | 4 | 71 | 47 | G | 90 | 5A | Z | 109 | 6D | m |
| 34 | 22 | " | 53 | 35 | 5 | 72 | 48 | H | 91 | 5B | [ | 110 | 6E | n |
| 35 | 23 | # | 54 | 36 | 6 | 73 | 49 | I | 92 | 5C | \ | 111 | 6F | o |
| 36 | 24 | $ | 55 | 37 | 7 | 74 | 4A | J | 93 | 5D | ] | 112 | 70 | p |
| 37 | 25 | % | 56 | 38 | 8 | 75 | 4B | K | 94 | 5E | ^ | 113 | 71 | q |
| 38 | 26 | & | 57 | 39 | 9 | 76 | 4C | L | 95 | 5F | _ | 114 | 72 | r |
| 39 | 27 | ' | 58 | 3A | : | 77 | 4D | M | 96 | 60 | ` | 115 | 73 | s |
| 40 | 28 | ( | 59 | 3B | ; | 78 | 4E | N | 97 | 61 | a | 116 | 74 | t |
| 41 | 29 | ) | 60 | 3C | < | 79 | 4F | O | 98 | 62 | b | 117 | 75 | u |
| 42 | 2A | * | 61 | 3D | = | 80 | 50 | P | 99 | 63 | c | 118 | 76 | v |
| 43 | 2B | + | 62 | 3E | > | 81 | 51 | Q | 100 | 64 | d | 119 | 77 | w |
| 44 | 2C | , | 63 | 3F | ? | 82 | 52 | R | 101 | 65 | e | 120 | 78 | x |
| 45 | 2D | - | 64 | 40 | @ | 83 | 53 | S | 102 | 66 | f | 121 | 79 | y |
| 46 | 2E | . | 65 | 41 | A | 84 | 54 | T | 103 | 67 | g | 122 | 7A | z |
| 47 | 2F | / | 66 | 42 | B | 85 | 55 | U | 104 | 68 | h | 123 | 7B | { |
| 48 | 30 | 0 | 67 | 43 | C | 86 | 56 | V | 105 | 69 | i | 124 | 7C | \| |
| 49 | 31 | 1 | 68 | 44 | D | 87 | 57 | W | 106 | 6A | j | 125 | 7D | } |
| 50 | 32 | 2 | 69 | 45 | E | 88 | 58 | X | 107 | 6B | k | 126 | 7E | ~ |

## Parcourir une chaîne :

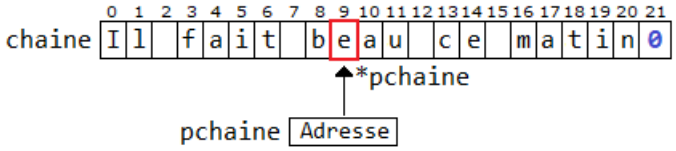
Exemple: afficher séparément les caractères d'une chaîne en allant du premier caractère jusqu'au dernier.

```
#include<stdio.h>

char chaine[] = "Il fait beau ce matin";
char *pchaine;

void main()
{
	for (pchaine = &chaine[0]; *pchaine != 0; pchaine++)
		printf("Le caractere est %c\n", *pchaine);
}
```

En C, la valeur ASCII de chaque caractère est stockée dans une cellule d'un vecteur d'octets. La cellule située immédiatement après les caractères de la chaîne contient la valeur 0 pour indiquer la fin de la chaîne.



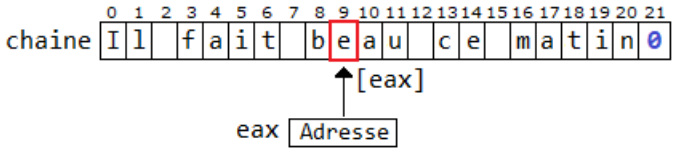
Dans ce programme, le pointeur pchaine pointe vers la cellule dans le vecteur chaine contenant le caractère courant. L'adresse que contient ce pointeur est incrémentée à chaque tour de boucle pour passer d'une cellule à la suivante.




Voici le programme en assembleur équivalent :

```
#include<stdio.h>

char chaine[] = "Il fait beau ce matin";
char *pchaine;
const char formatAffichage[] = "Le caractere est %c\n";

void main()
{
	_asm
	{
		mov	pchaine, offset chaine

	debutboucle:
		mov	eax, pchaine
		cmp	byte ptr [eax], 0
		je	finboucle

		movsx ebx, byte ptr[eax]
		push	ebx
		push	offset formatAffichage
		call	dword ptr printf
		add	esp, 8

		inc	pchaine
		jmp	debutboucle

	finboucle :
	}
}
```

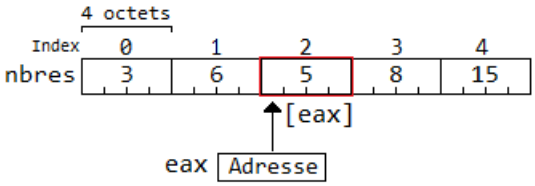
Un pointeur, pour le processeur, est un registre général de 32 bits utilisé en tant que pointeur. Ceci explique pourquoi, dans ce programme, EAX et non pas pchaine est utilisé pour pointer vers la cellule contenant le caractère courant de la chaîne.



## Parcourir un vecteur de cellules du type int :

Exemple: obtenir la somme des valeurs.

```
int nbres[5] = { 3, 6, 5, 8, 15 }, *pnbres = &nbres[0], i = 0, somme = 0;

void main()
{
	/*while (i < 5)
	{
		somme = somme + *pnbres;
		pnbres++;
		i++;
	}*/

	_asm
	{
	debutwhile:
		cmp	i, 5
		jge	finwhile

		mov	eax, pnbres
		mov	ebx, dword ptr[eax]
		add	somme, ebx

		add	pnbres, 4
		inc	i
		jmp	debutwhile

	finwhile:
	}
}
```

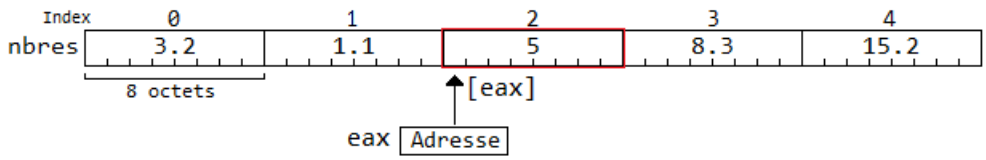



En C, l'adresse figurant dans un pointeur vers un vecteur dont les cellules sont du type int est augmentée automatiquement de 4 lorsqu'on applique l'opérateur ++ sur ce pointeur. Dans ce programme en assembleur, on doit ajouter explicitement 4 à pnbres à chaque tour de boucle pour qu'il stocke l'adresse de la cellule suivante du vecteur.

## Parcourir un vecteur de cellules du type double :

Exemple: obtenir la somme des valeurs.

```
double nbres[5] = { 3.2, 1.1, 5, 8.3, 15.2 }, *pnbres = &nbres[0], somme = 0;
int i = 0;

void main()
{

	/*while (i < 5)
	{
		somme = somme + *pnbres;
		pnbres++;
		i++;
	}*/

	_asm
	{
	debutwhile:
		cmp   i, 5
		jge   finwhile

		mov   eax, pnbres
		movsd xmm0, qword ptr[eax]
		addsd xmm0, somme
		movsd somme, xmm0

		add   pnbres, 8
		inc   i
		jmp   debutwhile

	finwhile:
	}
}
```




En C, l'adresse figurant dans un pointeur vers un vecteur dont les cellules sont du type double est augmentée automatiquement de 8 lorsqu'on applique l'opérateur ++ sur ce pointeur. Dans ce programme en assembleur, on doit ajouter explicitement 8 à pnbres à chaque tour de boucle pour qu'il stocke l'adresse de la cellule suivante du vecteur.

Saisir une chaîne de caractères :

Pour saisir un mot unique, on peut invoquer la fonction scanf avec le spécificateur de format %s. Cependant, scanf ne permet pas la saisie d'une chaîne dans laquelle il y a des espaces. Pour faire cela, il faut invoquer la fonction fgets dont le prototype est le suivant : *fgets(adresse chaîne, nombre maximum de caractères, stdin)*. Les 3 paramètres attendus par la fonction fgets sont : l'adresse du vecteur dans lequel sera stockée la chaîne, le nombre maximum de caractères y compris le zéro de fin et le flux d'entrée standard stdin. Un inconvénient de la fonction fgets, c'est qu'elle ajoute, après les caractères entrés, mais avant le zéro de fin, le caractère \n (caractère enter). Pour former une chaîne "propre", il faut remplacer ce caractère \n par un zéro de fin.

```
#include <stdio.h>
#include <string.h>

char msg[20];

void main()
{
	_asm
	{
		// récupérer le flux d'entrée stdin

		push dword ptr 0
		call dword ptr __acrt_iob_func
		add  esp, 4

		// fgets(msg, 20, stdin);

		push  eax               // le flux d'entrée
		push  dword ptr 20      // le nombre maximum de caractères + 2
		push  offset msg        // adresse de la cible où stocker les caractères
		call  dword ptr fgets
		add   esp, 12

		// strlen(msg) pour récupérer la taille de la chaîne

		push  offset msg
		call  dword ptr strlen
		add   esp, 4

		mov  msg[eax - 1], 0    // enlever le '\n' ajouté à la fin par fgets
	}
}
```

Ce programme permet la saisie d'une chaîne contenant au maximum 18 caractères. À la fin de la chaîne, la fonction fgets ajoute les 2 octets suivants : le caractère de retour à la ligne '\n' et le zéro de fin de chaîne. Comme on veut supprimer ce caractère de retour à la ligne '\n', on récupère la taille de la chaîne avec la fonction strlen, puis on remplace le caractère '\n' par un zéro de fin de chaîne.

# Exercices

## Exercices sur les vecteurs d'entiers et de doubles :

1. Afficher le contenu d'un vecteur de doubles.

2. Copier dans un vecteur de doubles uniquement les valeurs négatives figurant dans un vecteur d'entiers (par exemple, si VecEntiers contient [-9, 4, 8, -2, -7, 5], alors on aura dans VecDoubles [-9, -2, -7]).

3. Insérer une valeur saisie au clavier au bon endroit dans un vecteur d'entiers dont les valeurs sont déjà ordonnées dans un ordre croissant (par exemple, si VecEntiers contient [2, 8, 11, 15, 20] et que la valeur à insérer est 10, alors on aura dans VecEntiers [2, 8, **10**, 11, 15, 20]).

## Exercices sur les chaînes de caractères :

1. Vérifier si un mot est un palindrome. Un palindrome est un mot qui peut se lire indifféremment de gauche à droite ou de droite à gauche (par exemple, le mot *radar* est un palindrome).

2. Échanger 2 à 2 les caractères dans une chaîne de caractères (par exemple, pour la chaîne "bonjour", on aura "objnuor").

**Rappel** : les fonctions des librairies telles que printf, scanf, etc, modifient le contenu des registres généraux.