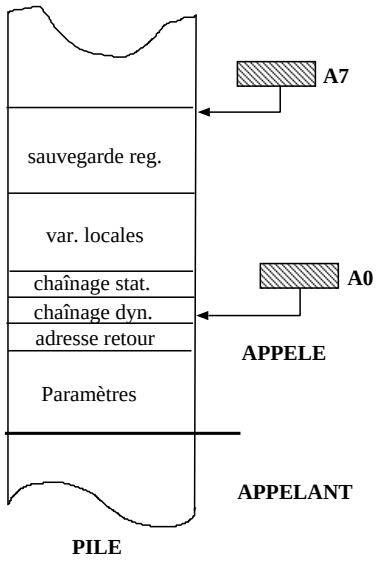
Mémoire à l'exécution

Rappels de cours.

Rappelez à quoi servent dans la pile : chaînage dynamique, chaînage statique, numéro de région, adresse de retour. Ces informations sont-elles toutes indispensables ? Qui les met en place, l'appelant ou l'appelé ?

On considère la structure suivante pour les environnements dans la pile :



Précisez, pour chacune des zones (paramètres, adresse de retour, etc), quelle est sa taille (et si celle-ci est connue à la compilation ou non), et qui (l'appelant ou l'appelé) accède à cette zone.

Mémoire à l'exécution.

Pour chacun des 3 programmes suivants écrits dans un langage à structure de blocs, dessinez la pile à l'exécution pour chacun des points cités dans le source et dans l'ordre d'exécution du programme.

Dans les programmes qui ne sont pas issus du langage C, un bloc est constitué par l'ensemble des instructions comprises entre les mots clés programme ou procedure ou fonction et fin.
Le mot clé var indique un passage de paramètre par adresse ; en l'absence de mot clé, le mode de passage est par valeur.
L'instruction retourner permet la sortie d'une fonction, et la valeur de l'expression donnée en argument est le résultat de la fonction.

Exercice 1.

Un exemple de programme non ANSI C et qui se compile avec gcc...

```
#include <stdio.h>

int I;
void func_D(int);

void func_A() {
  void func_B(int X) {
    int J, K;
    J = X;
    K = I * 10;
       /* POINT 1 */
    func_D(K);
  }
  func_B(I);
}

void  func_D(int X) {
  int I = 0;
  int func_E(int X) {
    I =  X + 100;
    printf(" valeur de I : %d\n", I);
    return(I);
       /* POINT 2 */
  }

  void func_F(int A, int B) {
    int J;
    J = I * A + B;
    J = func_E(5);
  }
  func_F(10, 100);
}

void main() {
  I = 5;
  func_A();
  printf(" valeur finale de I : %d\n", I);
}
```

Exercice 2.

```
Programme EXEMPLE1

  A, B, C: entier

  Procedure P1(var X: entier; Y: entier)
    X := X * 2
    Y := Y * 2
    P2(X, Y)
  fin_P1

  Procedure P2(var R: entier; S: entier)
    Procedure P3(var V: entier; S: entier)
      V := V * 3
      S := S * 3
         /* POINT 1 */
      C := R * V
    fin_P3

    R := R + S
    P3(S, R)

  fin_P2

debut //EXEMPLE1
  A := 10 ;  B := 20;  C := 30
  P1(A, B)   /* POINT 2 */
fin
```

Exercice 3.

Dessinez la mémoire à l'exécution sachant qu'on utilise un DISPLAY.

```
Programme EXEMPLE2

  A, B, C: entier

  Procedure P1(var X: entier; Y: entier)
    X := X * 2
    Y := Y * 3      /* POINT 1 */
    C := P2(X,Y) - X
  fin_P1

  fonction P2(var X: entier; var Y: entier): entier
    X := X * 10
    Y := Y * 100    /* POINT 2 */
    retourner(B + X + Y)
  fin_P2

debut // EXEMPLE2

  A := 10
  B := 100
  C := 1000
  P1(A, B)
fin
```