Pointeurs et allocation dynamique

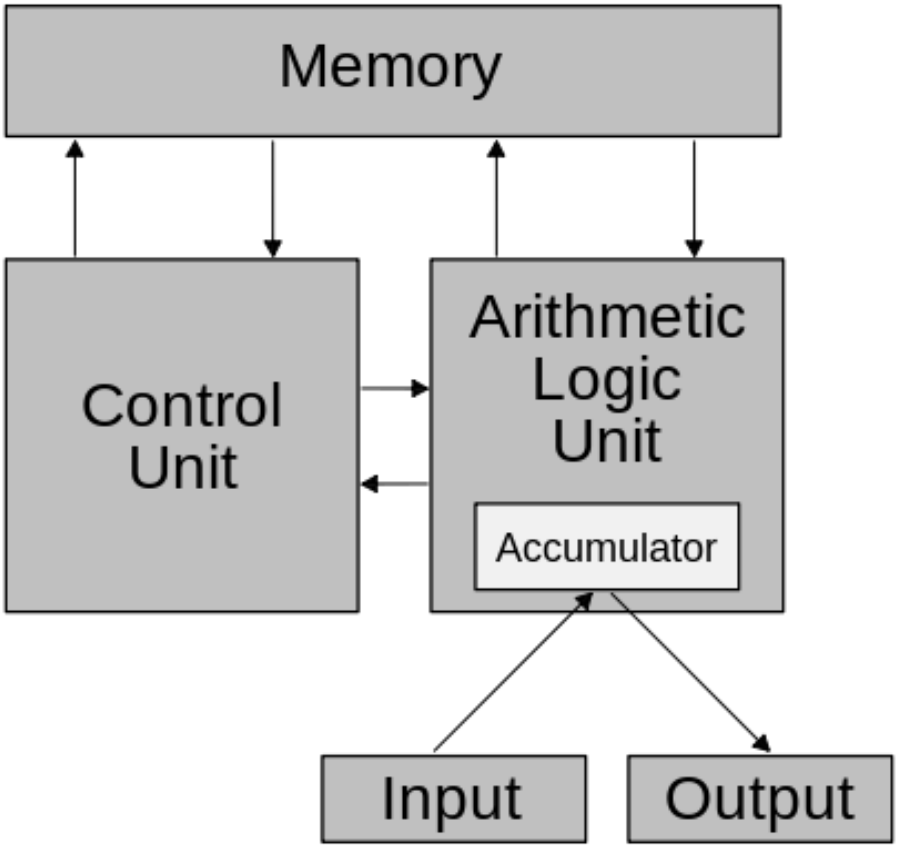
I2 - Algorithmique et Programmation structurée

Julien Saunier, Alexandre Pauchet, Pierrick Tranouez

Plan...

La mémoire

Architecture de Von Neumann



Éléments

- Entrées
- Unité de calcul
- Unité de contrôle
- Mémoire
- Sorties

Différents types de mémoire

Définition

En informatique, la mémoire est un dispositif électrotechnique qui sert à stocker des informations.

Diférents types de mémoire pour différents usages

- Mémoire de masse - stockage à long terme
- Mémoire vive :
  - RAM - mémorisation d'informations liées à l'exécution de programmes
  - Cache - mémoire intermédiaire
  - Registre - interne au microprocesseur pour la réalisation des opérations élémentaires

Et dans un programme ?

Allocation de mémoire

- Les entités utilisées (variables, constantes, (sous-)programmes) par le programme source sont représentées en mémoire (RAM de l'ordinateur)
- Il y a une relation directe entre l'identifiant que l'on utilise et un espace mémoire qui stocke l'information correspondante

Mémoire statique et mémoire dynamique

- Mémoire statique : l'emplacement est réservé dès la compilation.
  - Taille fixe
  - Dans le fichier exécutable
- Mémoire dynamique : l'emplacement est manipulé à l'exécution
  - Taille changeante
  - Dans un espace créé à l'exécution (et libéré ensuite)

Segments

5 segments

statique ou *text* : emplacement pour les programmes, sous-programmes

bss : emplacement pour les variables globales

data : emplacement pour les constantes

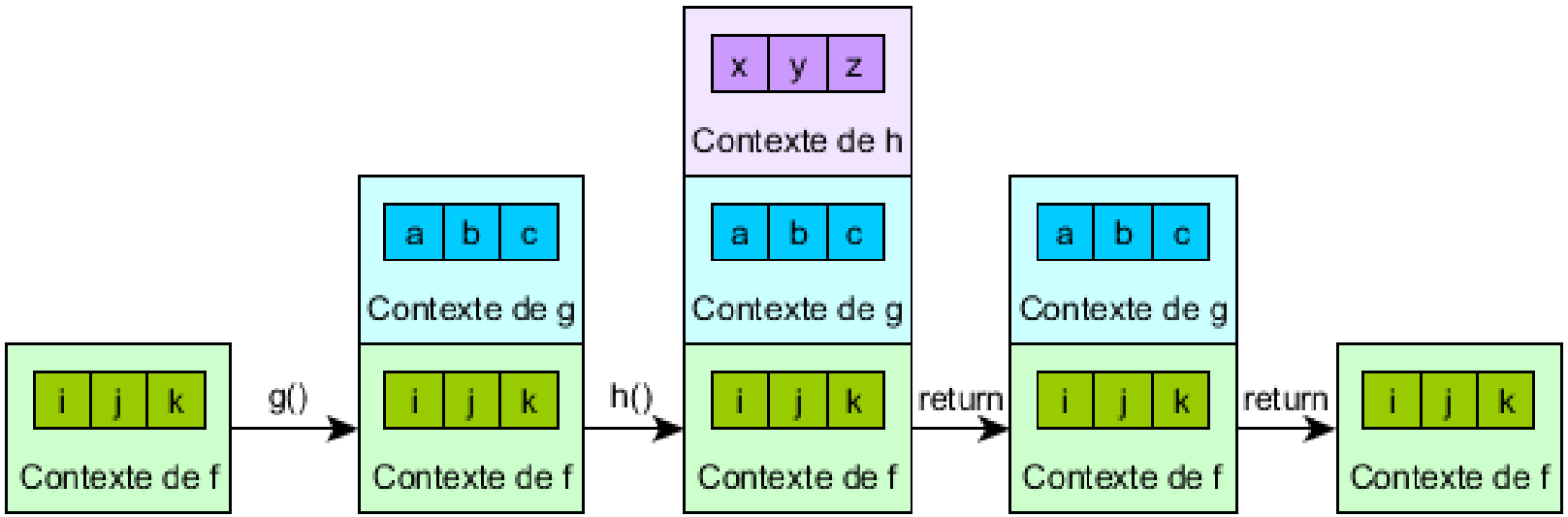
tas ou *heap* : emplacement où sont stockés des espaces mémoires alloués dynamiquement. La taille du tas varie en fonction de l'exécution du programme

pile ou *stack* : emplacement où sont stockées les variables locales et les paramètres formels des sous-programmes. La taille de la pile varie en fonction de l'exécution du programme

La pile

Pile

Fonctionnement LIFO (First-In Last-Out) en fonction de l'ordre d'appel des procédures et fonctions.



Les pointeurs

Utilité de la manipulation directe de mémoire

Limite de l'allocation statique

On ne peut pas toujours prévoir la taille mémoire nécessaire *a priori*
$\rightarrow$ sur-dimensionnement

Possibilité de manipuler dynamiqement la mémoire

- A la charge du programmeur
- Possibilité de réserver de l'espace mémoire (allouer) en fonction des besoins du programme
- Utilisation de l'adresse mémoire : le pointeur
- Deux procédures de base : allouer() et `l`ibérer()
- Dans le tas

Pointeur

Définition

- On nomme un « pointeur » p une variable permettant de référencer une zone mémoire permettant de stocker une information de type T
- Le type de p est nommé « pointeur sur T ». Il est noté (en algorithmique et en pascal) ^ T
- Lorsqu'une variable ne pointe sur aucune zone mémoire, il faut l'initialiser avec la valeur `NIL`

Opérateurs sur les pointeurs (en algorithmique et en pascal)

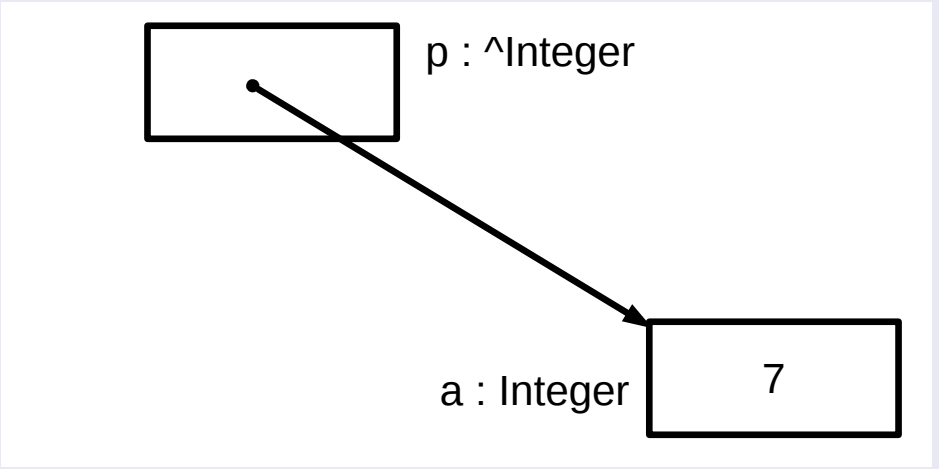
^ opérateur unaire (opérande à gauche de l'opérateur) permettant de « déréférencer » un pointeur (accéder à la valeur de la zone mémoire pointée)

@ opérateur unaire (opérande à droite de l'opérateur) permettant d'obtenir un pointeur sur une variable

Représentation

En pratique

Un pointeur sur entier
var p : ^ *Integer* ;
est une variable classique contenant une donnée de type adresse.



Effets notables :

- Changer la valeur de *a* modifie la valeur *p^* pointée par *p* (et inversement)
- Deux façons d'utiliser les pointeurs :
  - sur la mémoire déjà allouée
  - en utilisant les procédures new(p) et dispose(p)

Allocation dynamique et pointeur

Exercice : que fait la procédure `mystere` ?

```
procédure mystere ()
    Déclaration   i : Naturel
                  pi : ^ Naturel
debut
    i ← 0
    ecrire(i)
    pi ← @i
    pi^ ← pi^ +1
    ecrire(i)
    allouer(pi)
    pi^ ← i
    ecrire(pi^ )
    desallouer(pi)
    ecrire(pi^ )
fin
```

L'allocation dynamique de tableaux

L'allocation dynamique de tableaux

Objectif

Permettre de choisir la taille à allouer lors de l'exécution

Différences

- New/Dispose ne fonctionnent que sur des types de taille statique
- Utiliser getmem(pointeur,taille) et `f`reemem(pointeur)
- La taille est en octets, ou utilise la taille du type d'éléments stockés TABSIZE*SizeOf(Integer)
- Les indices démarrent à 0 !
- Pas de changement de taille dynamique (passer par une copie)

L'allocation dynamique de tableaux : Exemple

```
program test;

Type PInteger = ^Integer;

var monTabDyn:PInteger;
var i, tabsize:Integer;
begin
        readln(tabsize);
        monTabDyn:=GetMem(tabsize*SizeOf(Integer));
        for i:=0 to tabsize-1 do
                        monTabDyn[i]:=i;
        for i:=0 to tabsize-1 do
                        writeln(monTabDyn[i]);

        writeln(monTabDyn^);

        FreeMem(monTabDyn)
end.
```

L'allocation dynamique de tableaux à deux dimensions

Différences

Il faut voir un tableau à 2 dimensions comme un tableau de tableaux.

- Une première dimension est allouée dynamiquement comme un tableau de pointeurs
- Chacun de ces pointeurs doit faire l'objet d'une allocation dynamique d'une taille correspondant à la seconde dimension

Remarques

- Attention à bien libérer après utilisation toutes les allocations (il doit y avoir autant de libération de mémoire que d'allocations)
- On peut gérer une deuxième dimension de taille variable si nécessaire (impossible avec une allocation statique)
- Fonctionnement généralisable à des tableaux à N dimensions avec des tableaux de pointeurs sur des tableaux de pointeurs sur des tableaux

L'allocation dynamique de tableaux 2D : Exemple 1 / 2

```
program tabDyn2D ;

Type PInteger = ^Integer;
        PPInteger = ^PInteger;

var     maMatriceDyn : PPInteger;
        i, j, tabsize : Integer;
begin
        readln(tabsize);

        {allocation et initialisation}
        maMatriceDyn := GetMem ( tabsize * SizeOf ( PInteger ));
        for i:=0 to tabsize-1 do
        begin
                maMatriceDyn [i]:= GetMem ( tabsize * SizeOf (Integer
                for j:=0 to tabsize-1 do
                        maMatriceDyn [i][j] := i * tabsize + j ;
        end ;
```

L'allocation dynamique de tableaux 2D : Exemple 2 / 2

```
        {affichage}
        for i :=0 to tabsize-1 do
        begin
                for j :=0 to tabsize-1 do
                        write(maMatriceDyn[i][j], ' ');
                writeln();
        end ;

        {liberation}
        for i :=0 to tabsize-1 do
                FreeMem(maMatriceDyn[i]);
        FreeMem(maMatriceDyn);
end .
```

Exercice

C

Un tableau statique représentant le triangle de Pascal est par nature à moitié vide.
Ecrire le programme permettant de stocker le triangle dans un tableau dynamique.