Conférence d'introduction à la programmation et aux langages de programmation

C. Dubois

Plan

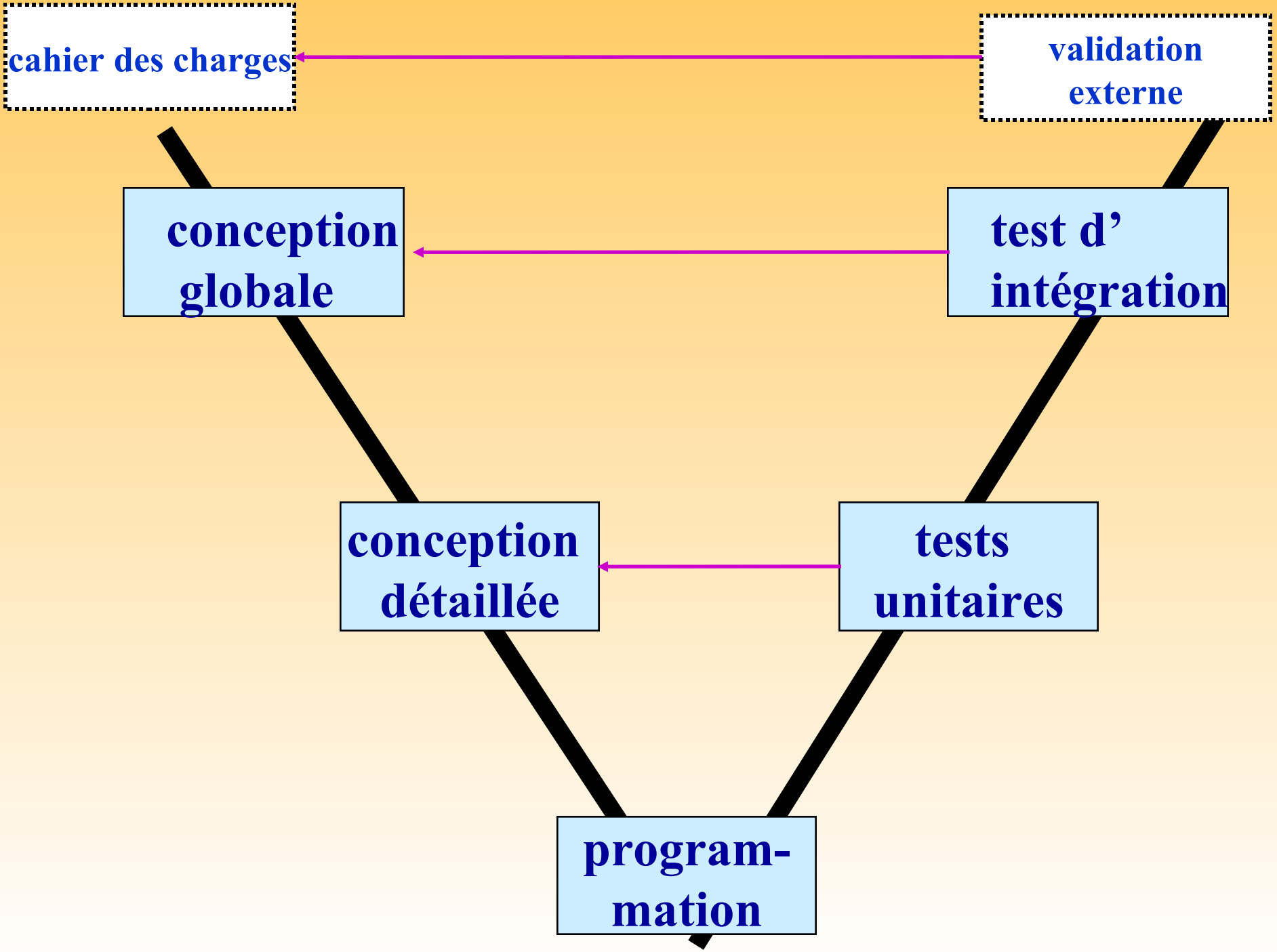
- Programmation et Développement de logiciel
- Les langages (un petit historique)
- Programmation fonctionnelle
- Programmation impérative
- Programmation objets
- Programmation logique

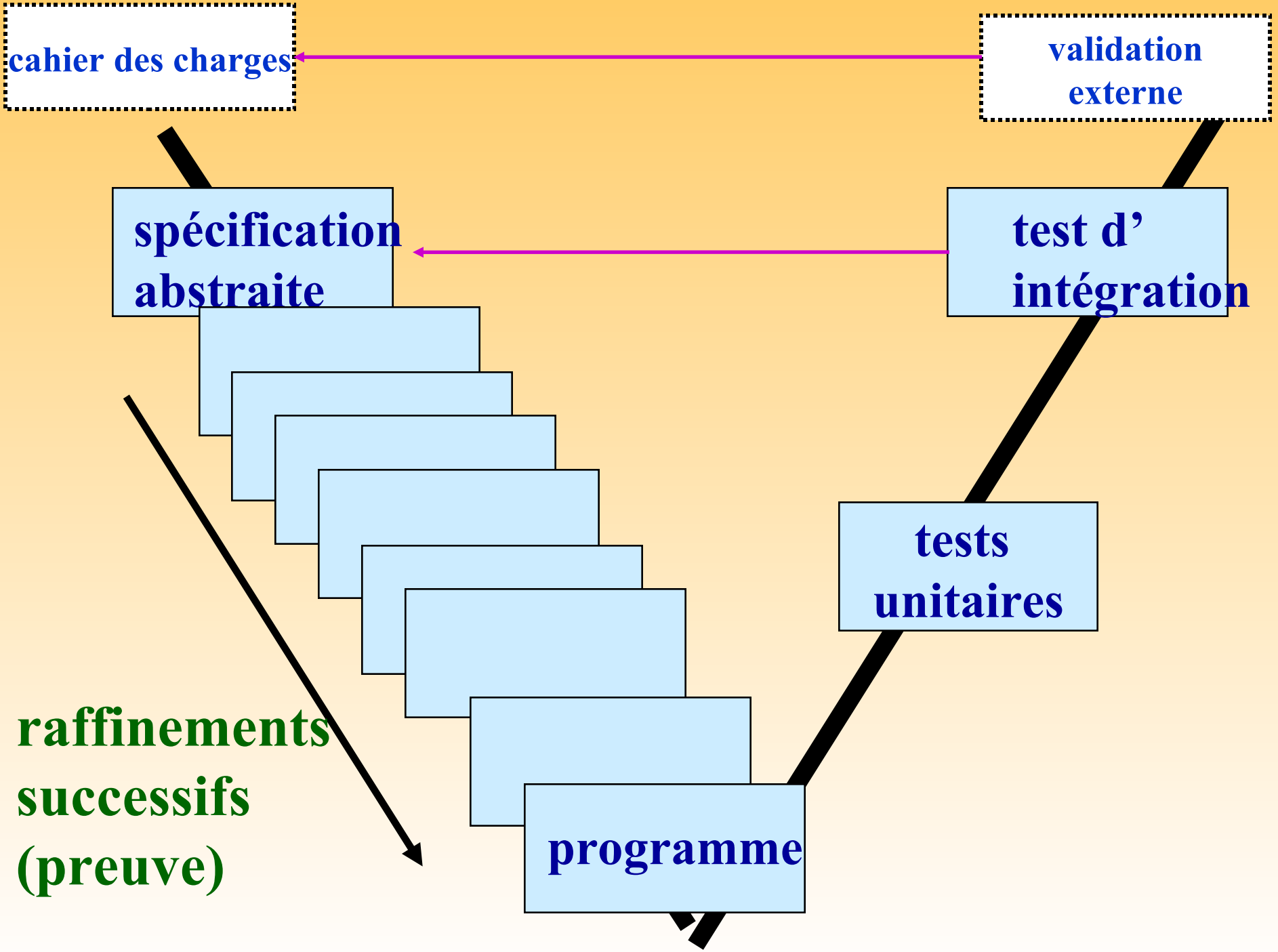
Programmation et Développement de logiciel

Qu'est-ce que la programmation ?

Description d'un calcul (traitement) dans un langage compréhensible par la machine (langage de programmation)

Une étape du cycle de vie du logiciel


cahier des charges
validation externe
conception globale
test d' intégration
conception détaillée
tests unitaires
program-mation


cahier des charges
validation externe
spécification abstraite
test d' intégration
tests unitaires
raffinements successifs (preuve)
programme

Un langage de programmation doit aider à écrire des programmes de bonne qualité

Programme de Bonne Qualité ?

- **correct**
- **robuste**
- **lisible, bien documenté**
- **facile à modifier, extensible**

Autres critères

- **efficacité**
- **portabilité**
- **intégrité**
- **réutilisabilité**
- **ergonomie, utilisation et apprentissage**

=> souvent nécessité de compromis

programmation structurée

types

mécanisme d'exceptions

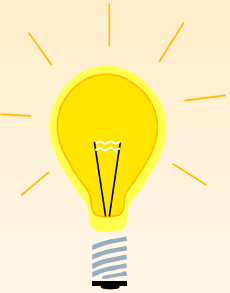
généricité, polymorphisme

surcharge

Programmation à petite échelle

$\neq$

Programmation à grande échelle



pour maintenir de grands programmes :

structure et organisation

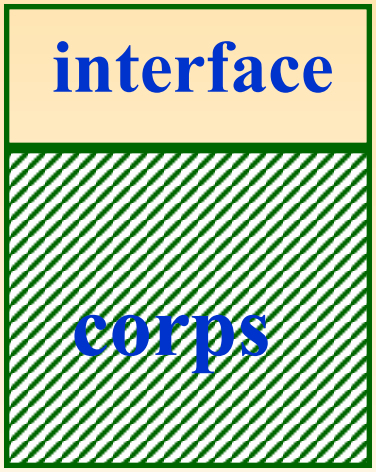
décomposer un grand programme en morceaux (modules)

connectés entre eux par des interfaces bien définies

mais aussi indépendants que possible

module = ensemble de ressources (données, types, opérations) liées sémantiquement

interface = mode d'emploi du module

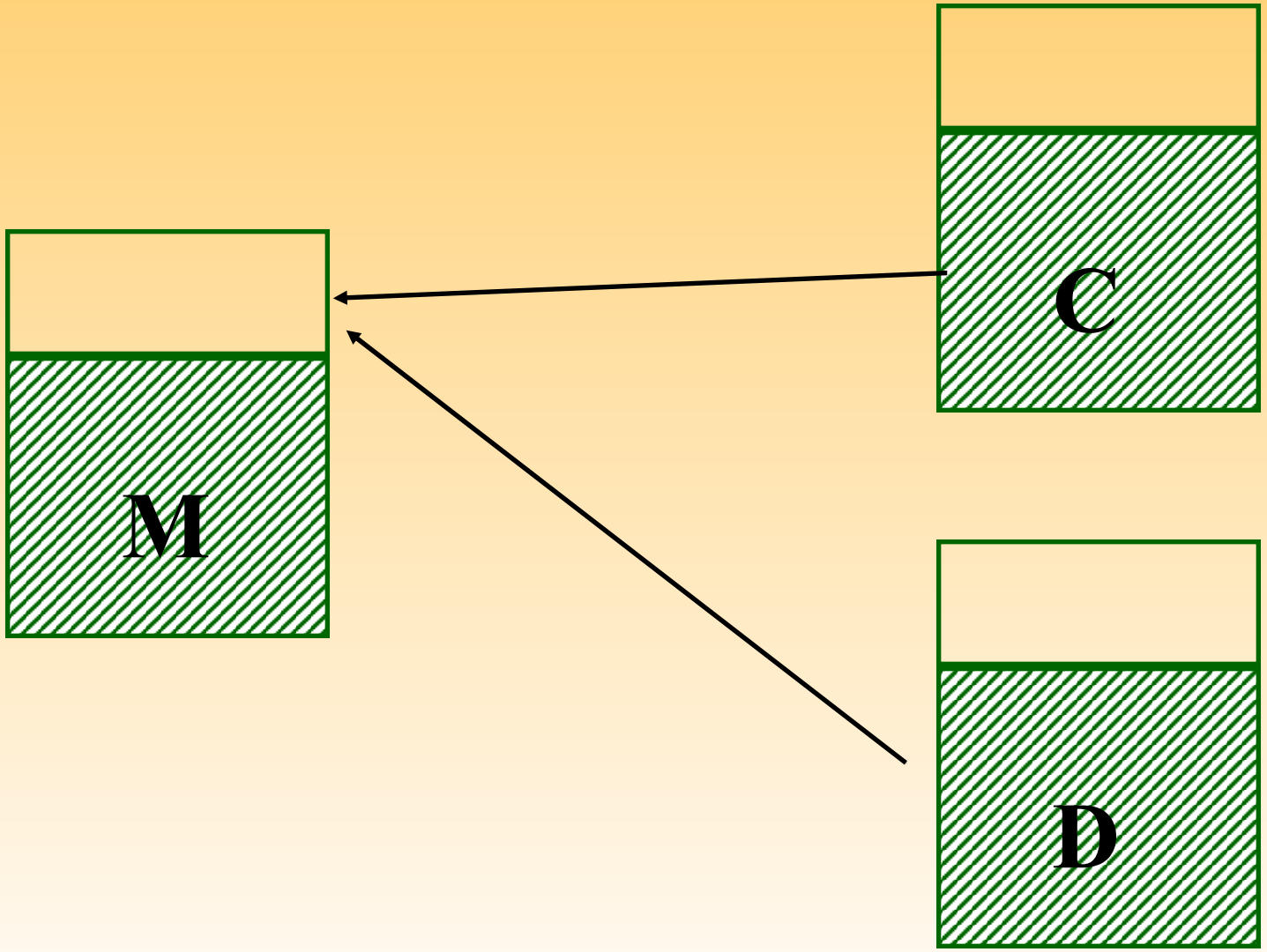


partie publique

partie secrète

principe du masquage des informations

Pourquoi ce principe ?



Langages de programmation

Les fonctions du langage

(d'après G. Berry, http://www.college-de-france.fr/default/EN/all/inn_tec2007/cours_n3_les_langages_de_progr.htm)

- Un outil pour écrire des programmes

par des hommes ou des programmes
pour la machine ou les autres hommes à différents niveaux d'abstraction

- Le support d'interaction avec la pensée

styles distincts (impératif, fonctionnel, logique, temporel, etc...)

- Le support de guerres de religions

On a toujours un style de prédilection

«l'histoire» montre un effort continu d'abstraction

=> échapper aux particularismes des machines

=> utiliser des concepts de haut niveau

Les langages de haut niveau sont caractérisés par des concepts tels que :

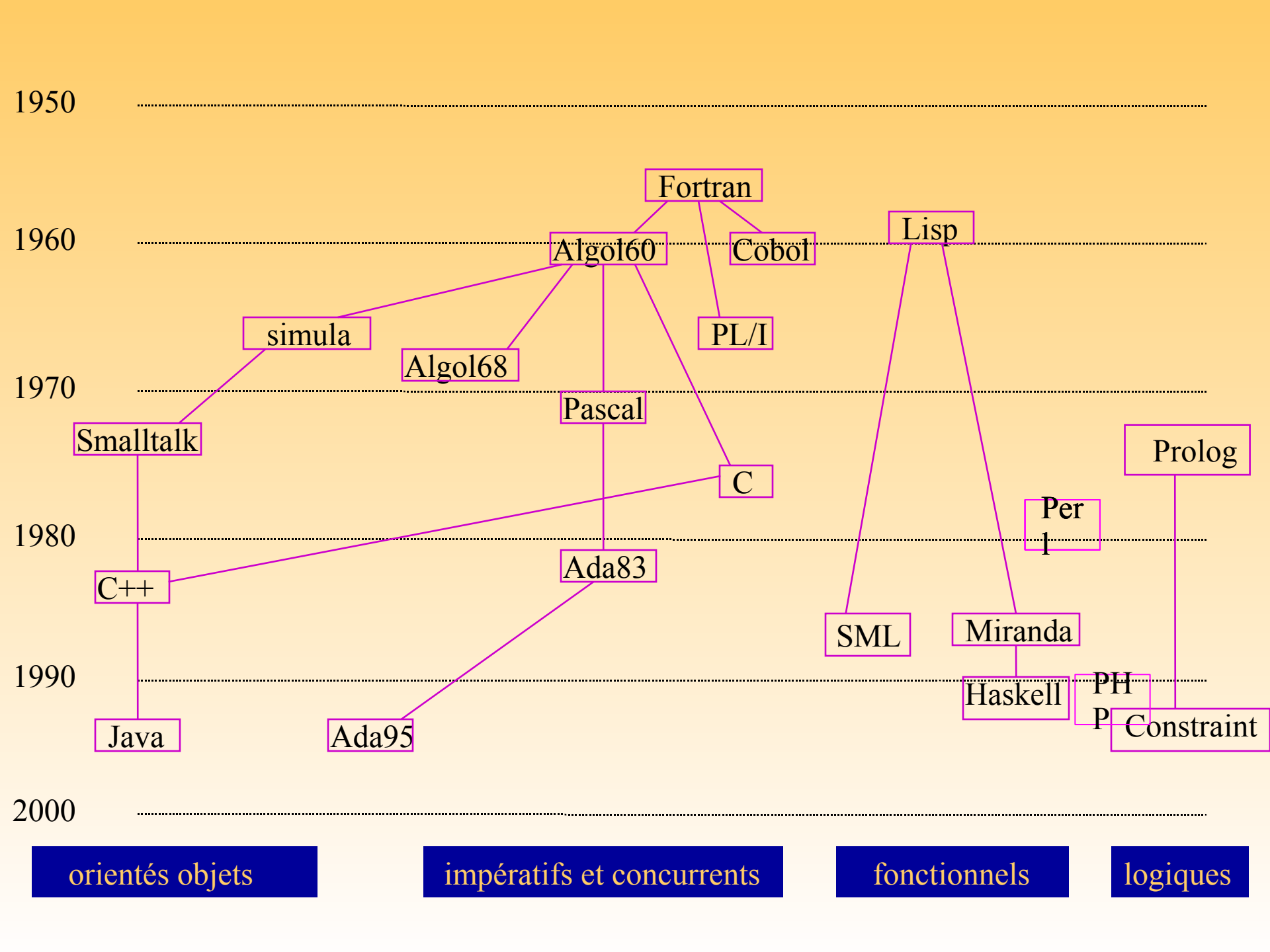
- **valeurs, types, expressions,**
- **variables, instructions, structures de contrôle**
- **liaison, portée, déclaration**
- **fonctions, procédures, paramètres**
- **encapsulation, modules, objets**
- **concurrence, tâches, communication**

Les concepts sont assemblés de différentes façons

différents paradigmes de

langages de programmation

Paradigmes	Concepts principaux
impératif	variables, procédures
concurrent	tâche/processus, communication
objets	objets, méthodes, héritage
fonctionnel	fonctions
logique	prédicats


1950
1960
1970
1980
1990
2000
Fortran
Algol60
Cobol
Lisp
simula
Algol68
PL/I
Pascal
Smalltalk
C
Prolog
Per
l
C++
Ada83
SML
Miranda
Haskell
PH
P
Java
Ada95
Constraint
orientés objets
impératifs et concurrents
fonctionnels
logiques

1950 : Invention de l'assembleur (Cambridge)

Avant, la programmation s'effectuait directement en binaire.

1951 : Invention du premier compilateur A0 par Grace Murray Hopper qui permet de générer un programme binaire à partir d'un code source.

Fortran (1958) :

- notation symbolique des variables,
- expressions arithmétiques

COMPILATEUR

Algol (1960)

- passage de paramètres
- structures de données plus riches, types

==========> Pascal (1970) C Ada (1980)

LISP (1958)

- manipulation symbolique
- récursivité, gestion automatique de la mémoire
- interactivité

=========> nombreux dialectes (Scheme)

BASES THEORIQUES DE LA PROGRAMMATION

(fin 60-début 70)

sémantique des langages, systèmes de preuve (λ-calcul, logique)

Prolog (72)

- fondé sur la logique des prédicats
- calculer avec des relations
- exécution = démonstration

==========> contraintes (90)

ML (langage de commande pour un démontrateur de théorèmes 78)

- système de types très puissant,
- inférence de types

==========> Caml (84, ENS/INRIA)

Modula2 (79)

modules

C++ (86)

C sous ensemble de C++

Eiffel

amène la notion de contrat

Java (95)

s'appuie sur C++, mais « nettoyé »

PHP (95)

langage de scripts multi-plateformes

pages HTML dynamiques et interactives

JavaScript (95)

insertion de code dans les pages HTML

C# (2000)

programmation internet – plateforme .net

Pourquoi tant de langages ?

• Beaucoup d'aspects à traiter ensemble
données, actions = programming in the small
architecture = programming in the large
comment réduire et détecter les bugs ?
gestion de la mémoire

• Beaucoup d'innovations successives
fonctionnel, polymorphisme, modules, objets, parallélisme,...

• Enormément de compromis possibles
plusieurs grandes classes et beaucoup de dialectes

• Mais, heureusement, des théories solides
automates, grammaires, lambda-calcul, logique

Pour finir (avec le sourire)

- Plus de 2500 langages ont été référencés en 2003

(en 67, 120 avec 15 utilisés seulement)

Langages délirants et exotiques (exemple : le langage cow « moo »)

- Une chanson à boire (99 bottles of beer) dont le texte a été écrit à l'aide de plus de 1120 langages de programmation (http://99-bottles-of-beer.ls-la.net)

Lyrics of the song 99 Bottles of Beer

99 bottles of beer on the wall, 99 bottles of beer. Take one down and pass it around, 98 bottles of beer on the wall.

98 bottles of beer on the wall, 98 bottles of beer. Take one down and pass it around, 97 bottles of beer on the wall.

....

1 bottle of beer on the wall, 1 bottle of beer.Take one down and pass it around, no more bottles of beer on the wall.

No more bottles of beer on the wall, no more bottles of beer. *Go to the store and buy some more, 99 bottles of beer on the wall*

Programmation fonctionnelle

Lisp, Scheme,

CamlI, SML,

Miranda, Haskell

etc

programme = ensemble de définitions de fonctions
résultat = application d'une fonction à un jeu particulier de données

composant de base : la fonction

opération de base : l'application

notion de fonction : celle des maths

définie par abstraction let f(x)=e

application f(a) ==> e[x/a] (substitution)

paramètre ou argument

corps

fonction : citoyen de première classe

Programmation impérative

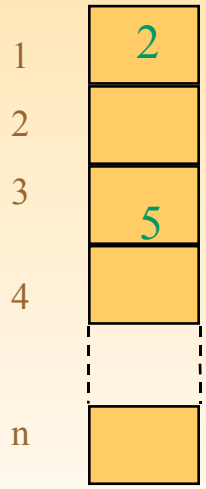
Pascal, Ada, C

etc

programme = suite d'instructions qui font évoluer l'état mémoire
le résultat est dans l'état final

mémoire = séquence de «locations» (cellules)

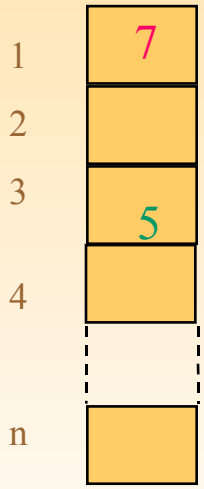
location –> adresse, valeur



c'est l'instruction d'affectation
qui fait évoluer la mémoire

x = x + y

@x = !x + !y



Instruction = ordre donné à la machine

affectation
primitives d'entrées/sorties
séquence

héritées de la structure des ordinateurs

instruction conditionnelle
boucle
appel de sous-programme

contrôle

Programmation objets

C++, Java

Eiffel, Smalltalk

Ocaml

etc

programme = collection d'objets qui communiquent entre eux par message
le résultat = un message envoyé à un objet particulier

objet = données **attributs, variables d'instance**

\+

opérations (sur ces données) **méthodes**

généralement, les variables d'instance sont cachées (encapsulées)

message $O_2\#m_2$ (O_2 objet, m_2 méthode de l'objet O_2)

exemple : un objet rectangle

variables d'instances :

longueur, largeur, point inférieur gauche

méthodes :

get_longueur, get_largeur, sommet_xx,

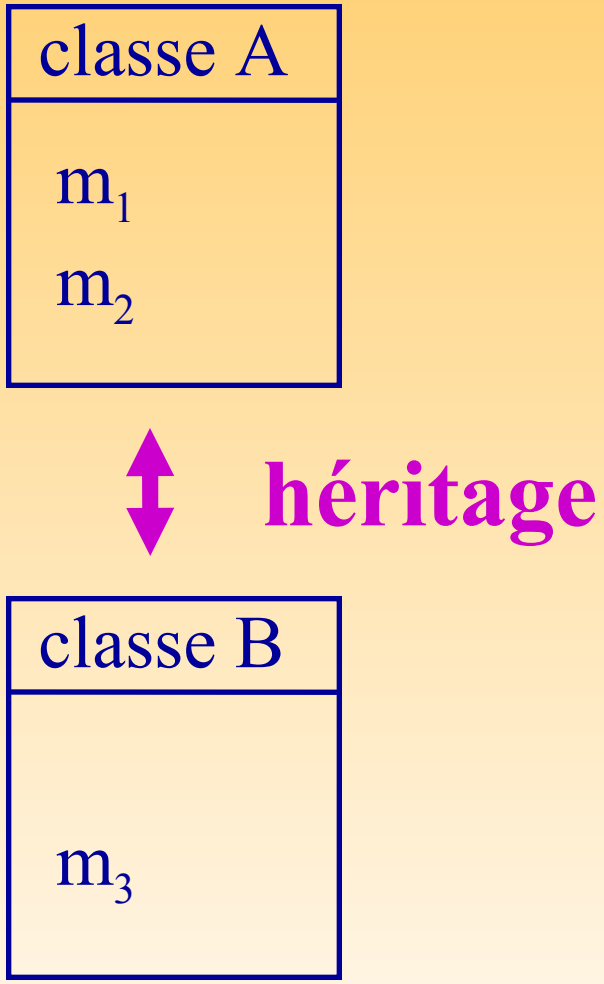
déplacer, étirer, allonger

Héritage

Créer une nouvelle classe en enrichissant, en spécialisant une classe existante

L'enrichissement peut porter sur :

- données : on ajoute de nouvelles variables d'instances
- méthodes :
  - on ajoute de nouvelles méthodes
  - on spécialise des méthodes existantes



A tout objet de la classe B, on peut envoyer les messages m_1, m_2 et m_3

Programmation logique

Prolog

et ses dialectes

langages à contraintes

programme = des règles et des faits

résultat = la ou les réponses à une requête

Composants de base =

prédicats (relations entre valeurs)

Exemple: somme (une relation entre 3 naturels)

somme(X,Y,Z) *Z est la somme des entiers X et Y*

somme(X,zero,X). ***un fait***

Si somme(X,succ(Y),Z)

alors somme(succ(X),Y,Z) . ***une règle***

Des exemples de requêtes

somme(1,2,R) **Z=3**

somme(1,2,4) **No**

somme(1,Y,4) **Y=3**

somme(X,Y,4) **X=0, Y=4**

X=1, Y=3

X=2, Y=2

X=3, Y=1

X=4, Y=0