FIPA IA - 2017 / 2020

ALGORITHMIQUE ET PROGRAMMATION EN C (ET PYTHON ?)

Pierre TELLIER

OBJECTIFS

- Une façon de raisonner
- Automatiser la résolution de problèmes
- Maîtriser les concepts de l'algorithmique
- Pas faire des spécialistes d'un langage

DÉFINITIONS

- Algorithme
  - Séquence finie d'opérations qui conduisent à la réalisation d'un calcul
  - Ensemble complet des règles permettant la résolution d'un problème donné
- Programme
  - Traduction de l'algorithme dans un langage de programmation

MOYENS

- Intellectuels
  - Créativité, imagination, abstraction, modélisation, structuration, déduction, induction …
- Pratiques
  - Entraînement intensif …
  - Expérimentation : choix d'un langage

LANGAGE

- Langage : syntaxe, grammaire
  - réalise certaines opérations (élémentaires …)
  - expressions
  - composition d'expressions
  - désignation d'objets
- Déclaratifs vs. impératifs
- Le langage C et son infinité de possibilités d'erreurs
- Un peu de Python ?

PLAN

- Intro, Objets et opérations élémentaires, branchements. Analyse descendante
- Fonctions, itérations et récursivité.
- Tableaux statiques, pointeurs et tableaux dynamiques
- Parcours, recherches, tris.
- Tableaux multidimensionnels. Application : images. Fichiers. Compilation séparée
- Listes. Types abstraits.

OBJETS ÉLÉMENTAIRES

- Nombres entiers : 26, -172.
- Nombres réels : 2.7, 3.14, 2.0, -3.4e-3.
- Caractères : 'A', 'a', '0', ',', '.', '\t ', '\n', ...
- Chaînes de caractères : "bonjour", "ceci est une chaîne".
- Booléens : VRAI ou FAUX = 0 ou 1

EXPRESSIONS ARITHMÉTIQUES

- 2 + 3
- 17 * 73 + 2
- 7 modulo 2 (reste de la division entière, vaut 1)
- 7 div 2 (vaut 3 : division entière)
- 7. / 2 (vaut 3.5 : division réelle)
- 0.3 * 168.2 + (4. + 0.11)/5.

- C vs. Python ...

C VS. PYTHON

- Nombres
  - types : `int, float`, et variantes !
  - opérateurs `+,-,/,*,%,==,!=, <,>, <=,>=`
- Caractères
  - `char`
  - `==,!=,<=,>=`
- Chaînes : "Hello" mais on verra + tard …

- Nombres
  - types : int, float, **`automatiques`**
  - opérateurs `+,-,/,//,*,%,**, ==,!=,<,>,<=,>=`
- Caractères
  - `chaînes de 1 caractère`
  - `==,!=,<=,>=,<,>`
- Chaînes : "Hello" ou 'Hello', +, ...

C VS. PYTHON

- Booléens
  - `int` (0 : FAUX, autre : VRAI), (FAUX : 0, VRAI : 1)
  - `&&, ||, !`
- Accumulation
- Opérations sur les bits
  - `&,|,~,<<,>>,^ (*/)`

- Booléens
  - `bool: true, false ()`
  - `or, and, not`
- Accumulation
- Opérations sur les bits ???
  - `&,|,~,<<,>>,^`
- `Appartenance`
  - `in, not in`

IDENTIFICATEURS

- Suite "presque" (chiffres, accents) quelconque de caractères, comme **fact, pgcd, produit, somme,** etc.
- Ils doivent OBLIGATOIREMENT être choisis de la façon la plus parlante possible, ou respecter des conventions de nommage.
- Dans tous les langages, certains identificateurs sont réservés.

CONSTANTES EN C

- const
  - const int nbMax = 100;
  - const int faux=0,vrai=1;
  - const float pi=3.14159;
- #define
  - #define NBMAX 100
  - #define FAUX 0
  - #define VRAI 1
  - #define PI 3.14159
- 2.0 * pi
- PI / 4.0
- 4.5 * (float)nbMax

CONSTANTES EN PYTHON

VARIABLES

- En maths : noms symboliques (**paramètres à valeur connue ou spécifiée plus tard**, inconnues, **variables de fonctions**)
- Cette notion existe dans presque tous les langages
- Sauf notion d'inconnue (excepté Prolog).

DÉCLARATION

- Objets désignés par un identificateur
  - Occupent un emplacement mémoire
- Servent à la résolution de problèmes
  - Résultats intermédiaires

EN C : DÉCLARATION

```
int  nombreEtudiants;

int age, nbFreresSoeurs, tailleFamille;

float rayon; float a, b, c;
```


EN PYTHON

C : DÉCLARATION & INITIALISATION

```
int nombreEtudiants = 20;

int age, nombreFreresEtSoeurs=0, tailleFamille = 3;

float rayon=-1.0, piSur2=pi/2.0;
```


PYTHON : INITIALISATION = DÉCLARATION

```
nombreEtudiants = 20

nombreFreresEtSoeurs=0

tailleFamille = 3

rayon=-1.0

piSur2=pi/2.0
```

EXPRESSIONS

- `nombreFreresEtSoeurs+2+1`
- `2.0 * Pi * rayon`
- `Pi * rayon * rayon`

- `C vs. Python`

AFFECTATION

- Consiste à **donner, associer** une valeur (le résultat d'une expression) à une variable
- Ranger valeur dans l'emplacement mémoire occupé par la variable

rayon ← 5.0

nombreFrèresEtSoeurs ← 4

tailleFamille← nombreFrèresEtSoeurs+1+2

EN C

```
rayon = 5.0;
nombreFrèresEtSoeurs = 4;
tailleFamille=nombreFreresEtSoeurs+1+2;
```

EN PYTHON

```
rayon = 5.0
nombreFrèresEtSoeurs = 4
tailleFamille=nombreFreresEtSoeurs+1+2
```


SÉQUENTIALITÉ

- Les instructions sont exécutées les unes après les autres
- Elles sont terminées par « ; » en **C**, Pascal, Php, etc
- ;; en Caml
- Numéro de ligne en Basic
- Ordre des lignes en Fortran et **Python**

  (mais le ';' ne gène pas en Python)

AFFECTATIONS

```
rayon = 5.0;
nombreFreresEtSoeurs = 4;
tailleFamille=nombreFreresEtSoeurs+1+2;


rayon = 5.0
nombreFreresEtSoeurs = 4
tailleFamille=nombreFreresEtSoeurs+1+2
```


VARIABLE INFORMATIQUE

- *Accumulation. exemple X ← X*2;*
  - En maths : $X_{n+1} = 2 * X_n$ …
  - En C & Python : `X = 2*X` ou `X *= 2`

VARIABLE INFORMATIQUE

- Echange des contenus de 2 variables
  - *tmp ← a;*
  - *a ← b;*
  - *b ← tmp;*

ECHANGE 2 NOMBRES …

- a ← a-b;
- b ← a+b;
- a ← b-a;

ENTRÉES/SORTIES

- *afficher();*
  - *afficher(x); afficher("texte à l'écran");*
  - *afficher(y+z);*
- *lire();*
  - *x← lire();*
- Interfaces graphiques, revamping
- Utilisation de navigateurs (HTML + CGI)

AFFICHER

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

```
printf("voici ");
printf("mon texte");
```

```
voici mon texte
```

```
printf("voici\n");
printf("mon texte");
```

```
voici
mon texte
```

```
print("voici ")
print("mon texte")
```

```
voici
mon texte
```

```
print("voici",end='')
print("mon texte")
```

```
voici mon texte
```



```
int x=2;
printf("%d",x);
printf("\n");
printf("x vaut : ");
printf("%d",x);
printf("\n");
```

```
2
x vaut : 2
```

```
printf("%d\n",2*x-5);
```

```
-1
```

```
x=2
print(x)

print("x vaut : ",end='')
print(x)
```

```
2
x vaut : 2
```

```
print(2*x-5)
```

```
-1
```



```
float y=2.;
printf("%f",y);
```

```
2.000000
```

```
printf("%f",2.*sqrt(y)-1.19);
printf("\n");
```

```
1.638427
```

```
y=2
print(y)
```

```
2.000000
```

```
print(2*y**0.5-1.19)
```

```
1.638427
```

```
printf("racine de ");
printf("%d",x);printf(" vaut ");
printf("%f",sqrt((float)x));
printf("\n");
```

```
racine de 2 vaut 1.414214
```

```
printf("racine de %d vaut %f\n",
   x,sqrt((float)x);
```

```
print("racine de ", x, end='')
print(" vaut ", end='')
print(float(x)**0.5)
```

```
racine de 2 vaut 1.414214
```

```
print("racine de ",x, " vaut ",
   x**0.5)
```


- Lecture d'un entier :

```
int x;
scanf("%d",&x);
```

- lecture d'un réel :

```
float y;
scanf("%f",&y);
```

- Lecture d'un entier :

```
x=int(input())
```

- lecture d'un réel :

```
y=float(input())
```


- Lecture d'un entier et d'un réel

```
int x;
float y;
scanf("%d %f\n",&x,&y);
```

- Lecture d'un caractère …
  - `scanf("%c", &car); // Attention !!!`
  - `car=getchar();`

- Lecture d'un entier et d'un réel

```
x = ...
y = ...
```

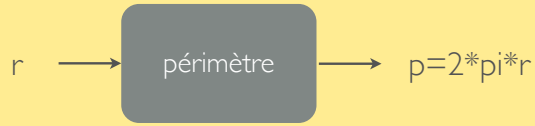
- Lecture d'un caractère …
  - `car = ...`

PROGRAMMES ÉLÉMENTAIRES

- (Constantes)
- Variables
- Début
  - Instructions;
- Fin

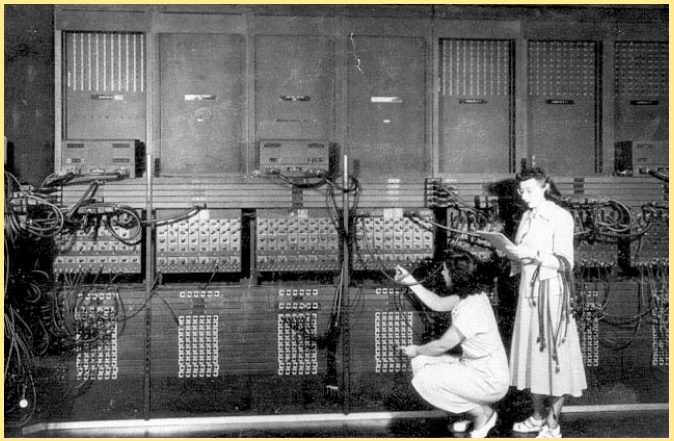
PÉRIMÈTRE D'UN CERCLE

- Spécification ?



- variables
- initialisation
- calcul
- affichage du résultat

PASSONS AU CODAGE



PERIMETRE.C

```
// ne m'oubliez pas !
#include <stdio.h>
int main(){
   const float PI = 3.141592;
   float r, p;
   printf(" Calcul du périmètre \n");
   printf(" Rayon ? ");
   scanf("%f", &r);
   p = 2.0*PI*r;
   printf(" périmètre : %f", p);
   return 0;
}
```

PYTHON

```
PI = 3.141592
r = ...
p = 2.0*PI*r
print(" périmètre d'un cercle de rayon", r, " : ", p)
```


PERIMETRE.PY

```
# ne m'oubliez pas
PI = 3.141592
print(" Calcul du périmètre")
r = float(input(" Rayon ? "))
p = 2.0*PI*r
print(" périmètre : ", p)
```


COMPILATION

- Compilateur : gcc, djgpp, vc++, …

- gcc perimetre.c : génère a.out
- gcc perimetre.c -o perimetre : génère perimetre
- gcc -Wall perimetre.c -o perimetre
- exécution : perimetre (./perimetre)

INTERPRÉTATION

- Interpréteur python. Exemple python3.5
- en mode commande : saisie et exécution
  - python3.5, saisir instructions
- exécuter un fichier :
  - python3.5 perimetre.py

LES NOMBRES

- décimaux
- binaires, petits indiens, boutisme
- hexadécimaux, octaux
- décimo-binaires (calculettes)
- réels (norme IEEE 754 8-23, 11-52)

COMPAREZ

```
#include <stdio.h>
int main() {
   float x =
   522219616.000000;
   float a = 10.00000;
   float res;

   res = x - a;
   printf("%f\n",res);

   return 0;
```

```
x = 522219616.000000
a = 10.00000

res = x - a
printf(res)
```

```
522219616.000000
```

```
522219606
```

FAUX !

```
int main () {
  int x,y;
  y = x+1;
  printf("y vaut : %d", y);
  return 0;
}
```

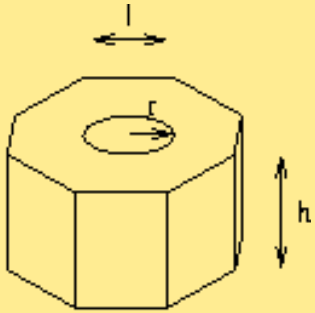
- contrôle de la syntaxe, pas de la sémantique
- Compilez avec l'option –Wall -Wextra

LE 1ER BUG

ANALYSE DESCENDANTE

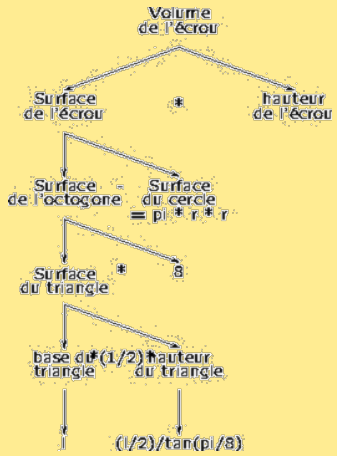
- Décomposer un problème en sous problèmes
  - Élémentaires
  - Faciles à résoudre

EXEMPLE



- Volume d'un écrou à 8 pans

DÉCOUPAGE FONCTIONNEL



CONDITIONNELLE

- Vérification des valeurs (domaine de définition)
  - racine carrée : nombre positif
  - cohérence des valeurs de l'écrou
  - horaires
  - …
- Pas le même traitement pour tout le monde
  - paie, primes, impôt, nature du revenu, situation familiale, assurance maladie (taux : soins, droits, …), mentions au bac

CONDITIONNELLE

- Structure de contrôle offerte par tous les langages de programmation
  - C :

```
if (cond) { <instructions C> }
    [else { <instructions C> }]
```

  - Php : idem (+ variantes)
  - Python :

```
if cond:
  <instructions Python>
[else:
  <instructions C> ]
```

  - Caml, Algol60 : `let v=if cond then expr else expr`
  - Pascal, Fortran, Basic, Cobol, VB…

MINIMUM DE 2 VALEURS

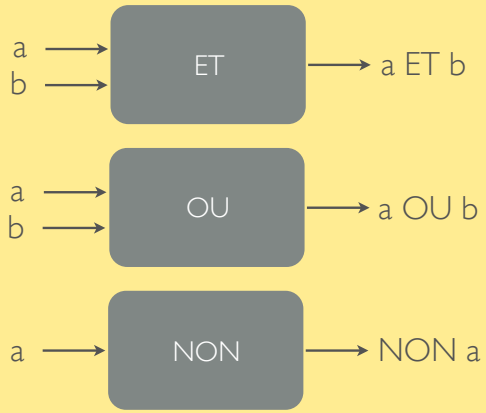

```
int res;
if (a < b) {
        res = a;
}
else {
        res = b;
}
```



```
int res;
if (a < b) res=a;
else res = b;
```

```
if a < b  :
        res = a
else :
        res = b
```

ET, OU, NON AVEC SI



CONDITIONS ET OPÉRATIONS BOOLÉENNES

- a ET b

b \ a	a ET b	VRAI	FAUX
	VRAI	VRAI	FAUX
	FAUX	FAUX	FAUX

C = A && B;

```
if (a!=0) {
  if (b!=0) res=1;
  else res=0;
}
else {
  if (b!=0) res=0;
  else res=0;
}
```

```
if (a) res = b;
else res = 0;
```

```
if a:
   res = b
else :
   res = false
```


- a OU b

b \ a	a OU b	VRAI	FAUX
	VRAI	VRAI	VRAI
	FAUX	VRAI	FAUX

C = A || B

```
int res;

if (a) res = 1;

else res = b;
```


- NON a

a	NON a
VRAI	FAUX
FAUX	VRAI

C = !A

```
int res;

if (a) res = 0;

else res = 1;
```


ENUMÉRATION DES CAS

```
switch (m) {
    case 1: nbj = 31; break;
    case 2: if(bissextile(a)) nbj=29; else nbj=28; break;
    case 3: nbj = 31; break;
    case 4: nbj = 30; break;
    case 5: nbj = 31; break;
    case 6: nbj = 30; break;
    case 7: nbj = 31; break;
    case 8: nbj = 31; break;
    case 9: nbj = 30; break;
    case 10: nbj = 31; break;
    case 11: nbj = 30; break;
    case 12: nbj = 31; break;
    default : printf("erreur\n");
}
```

```
if m==1:
        nbj = 31
elif m==2:
        if bissextile(a) :
                nbj=29
        else :
                nbj=28
elif m==3:
        nbj = 31
elif m==4:
        nbj = 30
elif m==5:
        nbj = 31
# ...
```

TYPE ÉNUMÉRÉ

- Définir un type par l'ensemble des valeurs possibles dans ce type
  - Booléen = {VRAI,FAUX}
  - Jour = {Lundi, … , Dimanche}
- Intérêt
  - Modélisation : pas de codes « obscurs »
  - Etude de cas exhaustive
- Inconvénient
  - Pas d'opérations arithmétiques

EN C

```
typedef enum {LUNDI, MARDI, MERCREDI, JEUDI,
  VENDREDI, SAMEDI, DIMANCHE} Jour;


Jour j;

// …

if (j==DIMANCHE) printf("youpi!");
```


REGROUPEMENT DES CAS

```
switch (m) {
      case 1:
      case 3:
      case 5:
      case 7:
      case 8:
      case 10:
      case 12: nbj=31; break;
      case 2: if (bissextile(a)) nbj = 29;
                  else nbj = 28; break;
      case 4:
      case 6:
      case 9:
      case 11: nbj = 30; break;
      default : sortirSurErreur("nbJoursDuMois");
}
```

```
if m==1 or m==3 or m==5 or m==7 or m==8 or m==10 or m==12 :
        nbj=31
elif m==2:
        if bissextile(a) :
                nbj = 29
        else:
                nbj = 28
elif m==4 or m==6 or m==9 or m==11 :
        nbj=30
else :
     :sortirSurErreur("nbJoursDuMois")
```


INTERPRÉTEZ

```
switch(m) {
            case 12: nbj+=30;
            case 11: nbj+=31;
            case 10: nbj+=30;
            case  9: nbj+=31;
            case  8: nbj+=31;
            case  7: nbj+=30;
            case  6: nbj+=31;
            case  5: nbj+=30;
            case  4: nbj+=31;
           case  3: if (bissextile(a)) nbj+=29; else nbj+=28;
            case  2: nbj+=31;
    }
```

MINIMUM DE 3 VALEURS

- Ecrivez un programme qui calcule le minimum de 3 nombres entiers

TEMPS DE PARCOURS

- Soustraction en base 60

```
   12h 30m 20s
-  10h 40m 30s
-------------
    1h 49m 50s
```


EQUATION DU 2ND DEGRÉ

ax2+bx+c =0


```
#include <math.h>

M_PI

pow, exp, sqrt, cos, sin, tan, acos, asin, atan, atan2,...

$ gcc test.c -o test -lm
$ man math
```

```
import math

math.pi    # ou pi

pow, exp, sqrt, cos, sin, tan, acos, asin, atan, atan2,...

>>> dir(math)
https://docs.python.org/
...
```