UNIVERSITÉ PARIS DESCARTES

IAP - Introduction à l'Algorithmique et à la Programmation

Equipe pédagogique
Marie-José Caraty, Denis Poitrenaud, Julien Rossit,
Camille Kurtz, Jacques Alès-Bianchetti, Mohamed Chelali

Cours Projet (2/2)

Les fondamentaux de la programmation impérative

Analyse du projet

http://www.tutorialspoint.com/c_standard_library/

Quelques dates qui vous intéresseront…

Semaines 6 et 7 semaine (du 11 octobre et du 18 octobre)

Les deux séances de TP (IAP2 et IAP3) sont dédiées au projet

Date des recettes

La **semaine du 18 octobre**
en séance de IAP3 (2ème TP de IAP)

Date de remise des dossiers de développement

Le **lundi 8 novembre** au secrétariat (impératif, retard pénalisé)

Exécution de l'application

Exécution à partir de Visual Studio 2019

Aller dans le menu **Démarrer**
*(icône **Windows** en bas à gauche de l'écran)*
*puis répertoire de programmes **Visual Studio 2019***
puis choisir l'application
Developer Command prompt for VS2019
Aller dans le répertoire contenant le programme

```
cd …
```

puis

```
d:
cl sprint1.c
```
(compilation en ligne de commande)

Exécution du Sprint1 :

```
sprint1.exe <inSp1.txt >run.txt
```


Passage de la recette

Recette de l'application

Validation par un observateur (le client)
en l'occurrence votre enseignant de TP
du Sprint de plus haut niveau atteint en phase de développement

Les JDT de recettes sont plus élaborés que les JDT de développement

Recette de l'application

Préparation à la recette

Préparez un projet Windows de votre application sur une machine de votre choix (machine de l'IUT, portable personnel accepté).

Recette

En présence de votre enseignant de TP, vous testerez votre programme avec un JDT de recette (`inSP`***`n`***`.txt`) correspondant au niveau **`n`**(`1≤n≤5`) du sprint le plus haut que vous avez validé en phase de développement avec les JDT de référence qui vous ont été communiqués pour chacun des sprints. Un nouveau JDT (de recette) vous sera communiqué par votre enseignant. Vous nommerez **`run.txt`** le fichier de sortie de votre programme.
Si votre fichier **`run.txt`** est **identique** au fichier résultat de référence **`refSp`***`n`***`.txt`** (donné par votre enseignant), votre logiciel est accepté : **le `Sprint#`*`n`***est validé**.
Sinon votre logiciel est refusé, vous avez 5 minutes pour corriger vos erreurs.
A la suite de ce délai, si vous ne passez toujours pas la recette, vous aurez à valider le **`Sprint#`*`n-1`***.
Rem **:** utilisez un comparateur de fichier (par exemple `diff`) pour comparer les fichiers. Ce comparateur affiche les lignes qui diffèrent.

Le problème posé

Objectif Programmer un *interpréteur de commandes*
pour la *gestion du classement WTA des joueuses de tennis*

Dans le classement WTA, les joueuses peuvent gagner des points lors de chaque tournoi en fonction du niveau atteint. Ces points s'ajoutent aux points précédemment gagnés au cours des 52 dernières semaines. Dans notre cas où nous ne prenons en compte que les 4 tournois du Grand Chelem, ***le classement WTA est établi à partir des points gagnés dans les 4 derniers tournois.***
Chaque tournoi voit s'opposer **128 joueuses** lors de **127 matchs** (*64 en 64èmes de finale, 32 en 32èmes de finale, 16 en 16èmes de finale, 8 en 8èmes de finale, 4 en quarts de finale , 3 en demi-finales et 1 en finale)*

Vainqueur	Finale	1/2 Finale	1/4 Finale	1/8 Finale	1/16 Finale	1/32 Finale	1/64 Finale
2000	1200	720	360	180	90	45	10

Spécifications (1/2)

Toute spécification donnée doit être suivie
sous peine de pénalisation

Déclaration et documentation des constantes utilisées

Objectif : *Eviter les nombres magiques par le nommage et dimensionnement des constantes utilisées*

Nombre magique : sont considérés comme nombre magique
toute constante numérique littérale
différente de 0 et de 1

Spécifications (2/2)

Autre spécification

« Les champs d'information d'une commande seront lus dans la fonction correspondante »

Objectif : *Structuration du code par unification du traitement des commandes*

Structuration des données

La structuration des données est obligatoire

Pénalisation en cas de non structuration : 0/20

Rappel

Une structure de données permet de regrouper les données liées à un type d'information

Raison majeure de la nécessité de structuration des données

Les fonctions, leur prototypage et leur appel

Passer en paramètre une variable de type structuré permet un accès (en lecture ou écriture suivant le type de passage [in], [out] ou [in out]) à chacun de ses champs

Exemple :
```
affichage_matchs_tournoi(TournoiWTA* t);
affichage_matchs_tournoi(const TournoiWTA* t);
//mieux (cf. cours 5)
```


Les constantes du problème (Sprint#1)

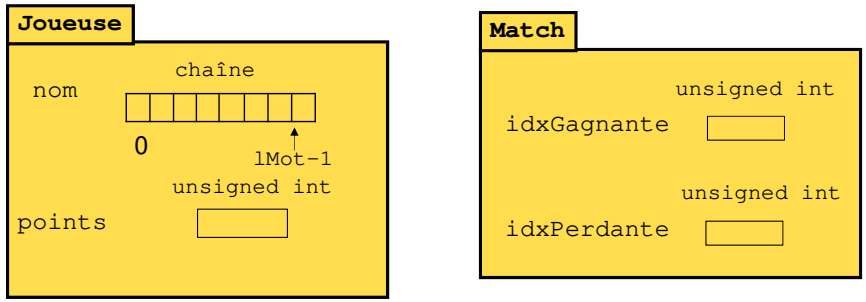
Nombre maximum de tournois	**10**
Nombre de match par tournoi	**127**
Nombre de joueuses du tournoi	**128**

Longueur maximum d'une chaine de caractère **30**

Chaine utilisée pour un nom, une date

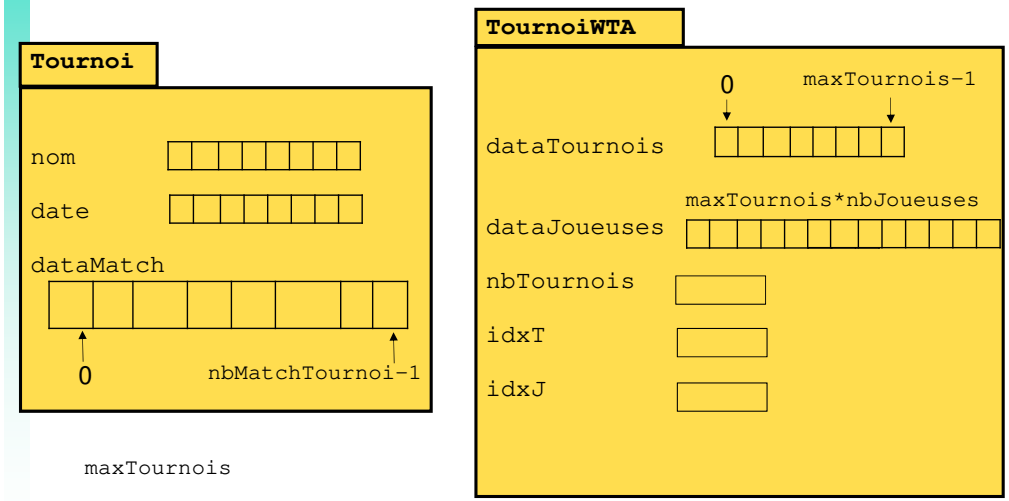
Les types utiles au Sprint#1

Structuration des données (Sprint#1) (1/2)

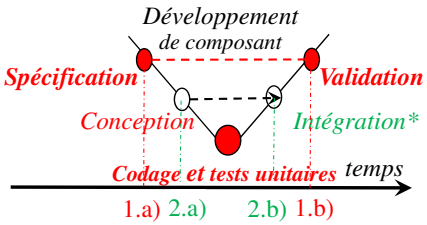


Les types utiles au Sprint#1

Structuration des données (Sprint#1) (2/2)



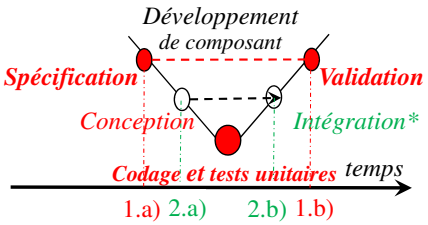
Développement agile



Le développement « agile » est l'un des grands principes du Génie logiciel

Dépasse le cadre de la programmation/codage
Méthodologie de développement « sûr »

Développement agile



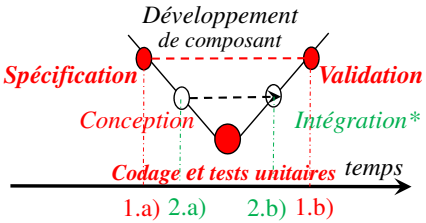
Choix d'un développement agile

Développement par Sprints,
chacun représentant un incrément de fonctionnalité de l'application

Cinq sprints sont définis et pour chacun des Sprints :

1) une analyse fonctionnelle
2) une spécification
3) un codage correspondant
4) un test de validation

Validation d'un Sprint

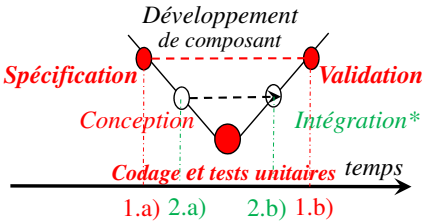


Principe de validation d'un **Sprint#*n*** à partir :
Jeu de Données de Test (JDT) : `inSpn.txt`
et des sorties attendues : `outSPn.txt`

L'exécution de l'exécutable du Sprint#n par redirection
- des entrées à partir de `inSpn.txt`
- des sorties vers `run.txt`

Si **`outSPn.txt`** coïncide avec **`run.txt`**
le Sprint#*n* est 0-défaut, il est validé

Validation d'un Sprint



A maîtriser

La redirection des entrées et sorties (cf. page 3).

PS : Vous pouvez glisser le nom de l'exécutable (nommé "Application" dans le dossier <debug> sous Visual Studio) dans la fenêtre de commande DOS (cmd)

De même, vous pourrez glisser le fichier "inSp#.txt" (où # est le n° de Sprint) dans cette fenêtre

Eléments de qualité d'un projet

Qualité d'un projet

- Architecture logicielle
- Structuration des données du programme
- Analyse des fonctionnalités/fonctions
- Code
- Documentation
- Livrable (dossier de programmation)

- **Notion différée : Architecture logicielle**

IOCCC-International Obfuscated C Code Contest
https://fr.wikipedia.org/wiki/International_Obfuscated_C_Code_Contest

IOCCC' 1987 – The winner was…

```
#define _ -F<00||--F-OO--;
 int F=00,OO=00;main(){F_OO();printf("%1.3f\n",4.*-
F/OO/OO);}F_OO()
 {
                  _-_-_-_
              _-_-_-_-_-_-_-_
          _-_-_-_-_-_-_-_-_-_-_-_
      _-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_
    _-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_
   _-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_
  _-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_
 _-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_
 _-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_
 _-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_
  _-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_
   _-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_
    _-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_
      _-_-_-_-_-_-_-_-_-_-_-_-_-_-_
          _-_-_-_-_-_-_-_-_
              _-_-_-_-_
 getch();
 }
```


IOCCC-International Obfuscated C Code Contest
https://fr.wikipedia.org/wiki/International_Obfuscated_C_Code_Contest

IOCCC' 1998 – The winner was…

```
#include                                <math.h>
#include                                <sys/time.h>
#include                                <X11/Xlib.h>
#include                                <X11/keysym.h>
                                        double L ,o ,P
                                       ,_=dt,T,Z,D=1,d,
                                       s[999],E,h= 8,I,
                                       J,K,w[999],M,m,O
                                       ,n[999],j=33e-3,i=
                                       1E3,r,t, u,v ,W,S=
                                       74.5,l=221,X=7.26,
                                       a,B,A=32.2,c, F,H;
                                        int N,q, C, y,p,U;
                                       Window z; char f[52]
                                       ; GC k; main(){ Display*e=
 XOpenDisplay( 0); z=RootWindow(e,0); for (XSetForeground(e,k=XCreateGC (e,z,0,0),BlackPixel(e,0))
; scanf("%lf%lf%lf",y +n,w+y, y+s)+1; y ++); XSelectInput(e,z= XCreateSimpleWindow(e,z,0,0,400,400,
0,0,WhitePixel(e,0) ),KeyPressMask); for(XMapWindow(e,z); ; T=sin(O)){ struct timeval G={ 0,dt*1e6}
; K= cos(j); N=1e4; M+= H*_; Z=D*K; F+=_*P; r=E*K; W=cos( O); m=K*W; H=K*T; O+=D*_*F/ K+d/K*E*_; B=
sin(j); a=B*I*D-E*W; XClearWindow(e,z); t=2*E+ D*B*W; j+=d*_*O-_*F*E; P=W*E*B-T*D; for (o+=(I=D*W+E
*I*B,E*d/K *B+v*B/K*I*D)*_; pcyj ); I=p[s]+i; E=c-p[w]; D=n[p]-L; K=D*m-B*T-H*E; if(p [n]+w[ p]<pi
]== 0|K <fabs(W=T*r-I*E +D*P) |fabs(D=t *D+Z *T-a *E)> K)N=1e4; else{ q=W/K *4E2+2e2; C= 2E2+4e2/ K
 *D; N-1E4& XDrawLine(e ,z,k,N ,U,q,C); N=q; U=C; } ++p; } L+=_* (X*t +P*M+m*1); I=X*X+ l*1+M *M;
  XDrawString(e,z,k ,20,380,f,17); D=v/l*15; i+=(B *1-M*r -X*2)*_; for(; XPending(e); u *=C5!=N){
                                       XEvent z; XNextEvent(e ,&z);
                                     ++*((N=XLookupKeysym
                                      (&z.xkey,0))-IT?
                                      N-LT? UP-N?& E:&
                                      J:& u: &h); --*(
                                      DN -N? N-DT ?N==
                                      RT?&u: & W:&h:&J
                                      ); } m=15*F/l;
                                      c+=(I-M/ 1,1*H
                                      +I*M+a*X) *_; H
                                      =A*r+v*X-P*l+(
                                      E=.1+X*4.9/l,t
                                      =T*m/32-I*T/24
                                       )/S; X=F*M+(
                                       h* 1e4/l-(T+
                                       E*5*T*E)/3e2
                                       )/S-X*d-B*A;
                                       a=2.63 /l*d;
                                       X+=( d*l-T/S
                                        *(.19*E +a
                                        *.64+J/1e3
                                        )-M* v +A*
                                        Z)*_; l +=
                                        K *_; W=d;
                                        sprintf(f,
                                        "%5d  %3d"
                                        "%7d",p =1
                                       /1.7,(C=9E3+
   O*57.3)%0550,(int)1); d+=T+(.45-14/l*
  X-a*130-J* .14)*_/125e2+F*_*v; P=(T*(47
  *I-m* 52+E*94 *D-t*.38+u*.21*E) /1e2+W*
  179*v)/2312; select(p=0,0,0,0,&O); Y-=(
   W*F-T*(.63*m-I*.08+m*E*19-D*25-.11*u
    )/107e2)*_; D=cos(o); E=sin(o); } getch() }
```


Dossier de développement logiciel

❑ Respect des spécifications

Respect des spécifications données pour le dossier de développement logiciel (livrable)

Respect du cahier des charges de présentation
Présence dans le dossier a) une page de garde indiquant le nom et le groupe des membres du binôme/trinome, l'objet du dossier, illustration de la page de garde, b) une table des matières de l'ensemble du dossier (incluant les annexes) avec la pagination de toutes les rubriques, la pagination est continue du début à la fin du dossier, c) présentation de l'application, e) organisation des tests, f) bilan de validation des tests de développement, g) bilan de projet. Brochez vos dossiers.

❑ Présentation de l'application

L'art de synthétiser le projet dans un format donné (1 page). On devra trouver dans cette présentation son rôle fonctionnel (ce que fait l'application), ses entrées et ses sorties (au moins pour le Sprint validé).

Qualité de code (1/2)

❑ Lisibilité du code

Code lisible
Code parfaitement indenté (tabulation) [in-out] pour un paramètre d'entrée-sortie]
Ligne de code limitée à 80 colonnes

❑ Documentation

Du fichier source (c'est le cartouche)
(en début de fichier : nom de fichier, nom et groupe(s) des auteurs, date de création)
Des types structurés et de leurs champs
Des variables importantes
Des fonctions :
rôle de la fonction, des paramètres formels, leur mode ([in], [out] et [in-out])
rôle du paramètre de retour éventuel ([return])

```
/*
 *  Calcul du maximum de deux valeurs
 *  [in]  x, 1ere valeur
 *  [in]  y, 2ème valeur
 *  [out] mx, le maximum          // [in-out] pour un paramètre d'entrée-sortie]
 */                               // [return] pour un paramètre de retour
void max(int x, int y, int* mx);
```


Qualité de code (2/2)

❑ Commentaires dans le code

Savoir commenter les parties « complexes » du code, les structures de données,…

❑ Absence de nombres magiques

Absence de littéraux (autre que 0 et 1), utilisation des constantes ou de #define motivée par des changements possibles des constantes considérées

❑ Longueur des fonctions

Le `main` doit être court (à très court).
Un `main` est à un niveau macroscopique (constitué d'appels de fonction).
Un main trop long traduit un manque au niveau de l'analyse (des fonctions auraient dû être conçues et être appelées par le main).
Par exemple, on n'y développe pas une interface (e.g., un menu).
Une fonction ne doit pas dépasser une vingtaine de lignes. Une fonction trop longue indique souvent un manque d'analyse fonctionnelle et s'accompagne de redondance : dans ce cas, on relit et on analyse le code en visant un niveau mascroscopique (de l'algorithme) pour introduire les fonctions (à coder) et qui seront à appeler (la longueur du code diminuera).

Les tests

❑ Code redondant

Introduire la/les fonction(s) qui généralisent les traitements.

❑ Tests

D'une manière générale, vous devez préciser votre stratégie de test : comment sont organisés vos tests.
Un test comprend : un objectif (ce que l'on veut tester), un JDT (jeu de test), un résultat attendu (résultat de référence), un résultat (**trace d'exécution** de votre programme) et un **bilan de validation** (ce que valide le test).
Il faut rédiger cette partie.

Dans le rapport, on doit trouver une partie de « **Bilan de validation des tests** » où on résume tous les tests et ce qu'ils valident.
But : vérifier votre compréhension des tests.

Des JDT « personnels » servent à améliorer la couverture des tests (non demandés)

❑ Bilan de projet

Retour d'expérience. Les difficultés rencontrées, ce qui est réussi, ce qui peut être amélioré

Evaluation des projets – Barème indicatif

[15 pts] NS	Note de Sprint (du plus haut niveau #n) validé lors de la recette
[1 pt] MOD	Modulation d'un Sprint de niveau n validé relativement au source du sprint#n+1 (si présent dans le dossier)
[1 pt] PA	Présentation de l'application
[1 pt] BV	Bilan de validation
[1 pt] BP	Bilan de projet
[2 pts] QC	Qualité du code
[-1 pt] PN	Pénalités de non respect des spécifications (cf. texte du projet) Ex. N° de groupe absent sur la page de garde, pagination absente..

Barème indicatif
NS [/15 points] Barème établi sur le Sprint de plus haut niveau validé à la recette
Sprint 5 : 15 points
Sprint 4 : 13 points
Sprint 3 : 11 points
Sprint 2 : 08 points
Sprint 1 : 05 points
Sprint 1 non atteint : 0 point

ATTENTION 0/20 si le dossier de développement n'est pas rendu
0/20 si le code n'est pas présent dans le dossier