Programmation en Langage C

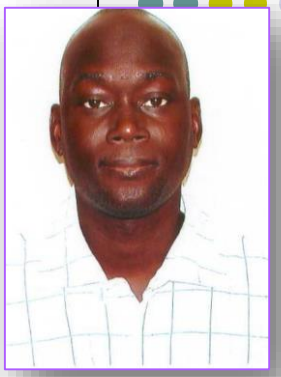
INTRODUCTION GÉNÉRALE

Pr. SALL Ousmane

Cours de Programmation

Cours n°1

A propos de moi



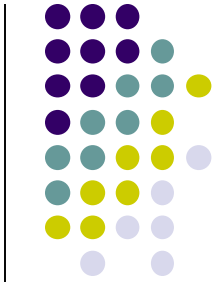
- Enseignant-Chercheur à **l'UFR SET- Université de THIES** http://sites.univ-thies.sn/osall751/
- **Enseignements:**
  - Algorithmique et Programmation(C, Java, PHP)
  - Programmation WEB dynamique(XHTML/Styles CSS, PHP, MySQL, Drupal, Joomla, SPIP, WordPress)
  - Programmation d'Applications Réparties(Tomcat Server, Jboss, JSP, JavaBeans, Servlet, EJB,…)
  - Programmation d'Applications Embarquées avec Java ME et Android
  - Administration de Bases de données sous Oracle

- **Contact:**
  - osall@univ-thies.sn
  - **UFR SET, Université de THIES, BP 967 THIES.**

Une sagesse chinoise…

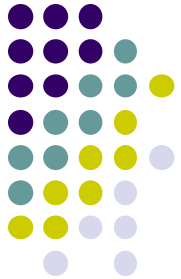
« J'écoute et j'oublie; je lis et je comprends; je fais et j'apprends »
[Proverbe chinois]

Sommaire



- *Introduction générale*
- Premiers pas
- Variables
- Conditions et boucles
- Mise au point
- Pointeurs et fonctions
- Tableaux et chaînes de caractères
- Structures et fichiers
- Débogage d’un programme
- Compléments
- Quelques exemples de programmes

Organisation pédagogique et Modalités d'évaluation



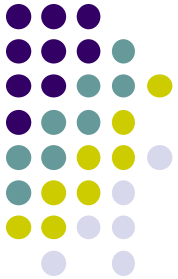
- Pré-requis:
  - Notions de base windows ou linux,…
- Durée du cours : 20h
  - Séances de cours
  - Séances de TD
  - Séances de TP
- Une évaluation finale
  - Questions de cours
  - Exercices

Pourquoi un cours de programmation ?

- **Objectif:** obtenir de la «machine» qu'elle effectue un travail à notre place
- **Problème:** expliquer à la «machine» comment elle doit s'y prendre

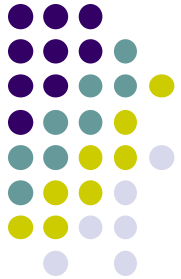
  - *Mais... comment le lui dire ?*
  - *Comment le lui apprendre ?*
  - *Comment s'assurer qu'elle fait ce travail aussi bien que nous ?*
  - *Mieux que nous?*

Objectif de cet enseignement



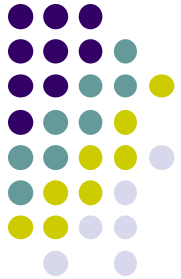
- résoudre des problèmes «comme» une machine
- savoir ***expliciter*** son raisonnement
- savoir ***formaliser*** son raisonnement
- concevoir (et écrire) des ***programmes :***
  - séquence d'instructions qui décrit comment résoudre un problème particulier

Langages de programmation



- **Un langage de programmation** est un outil permettant de donner des ordres (instructions) à la machine :
  - A chaque instruction correspond une action du processeur
- **Intérêt** : écrire des programmes (suite consécutive d'instructions) destinés à effectuer une tache donnée. Exemple: un programme de gestion de comptes bancaires
- **Contrainte**: être compréhensible par la machine

Langages de programmation



- **Deux types de langages**:
  - **Langages procéduraux** : sont à base de procédures. Une procédure est une portion de programme écrit en langage de haut niveau qui accomplit une tâche spécifique nécessaire au programme.
  - **Langages orientés objets** : sont des langages non procéduraux dans lesquels les éléments du programme sont considérés comme des objets qui peuvent s'échanger des messages.
- Choix d'un langage?

Langages de programmation

Plus de neuf mille langages de programmation

En comparaison on retrouve autour de 6800 langues humaines dont 200 seulement existent en forme écrite.

Pour aller plus loin…

- Tapez "cours langage c" sur GOOGLE http://www.google.sn/
- Le site http://developpez.com
- Algorithmique,…
  - http://www.siteduzero.com/tutoriel-3-51781-algorithmique-pour-l-apprenti-programmeur.html
  - http://www.siteduzero.com/tutoriel-3-14668-concevez-votre-site-web-avec-C-et-mysql.html
  - Tapez "cours Algorithmique" sur GOOGLE http://www.google.sn/
- …

Programmation en Langage C

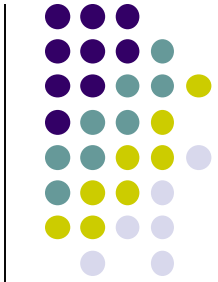
PREMIERS PAS

Pr. SALL Ousmane

Cours de Programmation

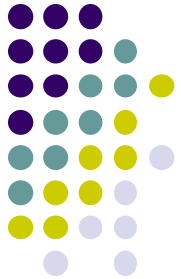
Cours n°2

Sommaire



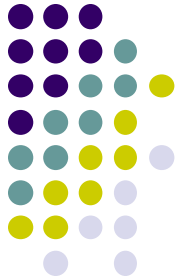
- Introduction générale
- ***Premiers pas***
- Variables
- Conditions et boucles
- Mise au point
- Pointeurs et fonctions
- Tableaux et chaînes de caractères
- Structures et fichiers
- Débogage d'un programme
- Compléments
- Quelques exemples de programmes

Système d'exploitation et C



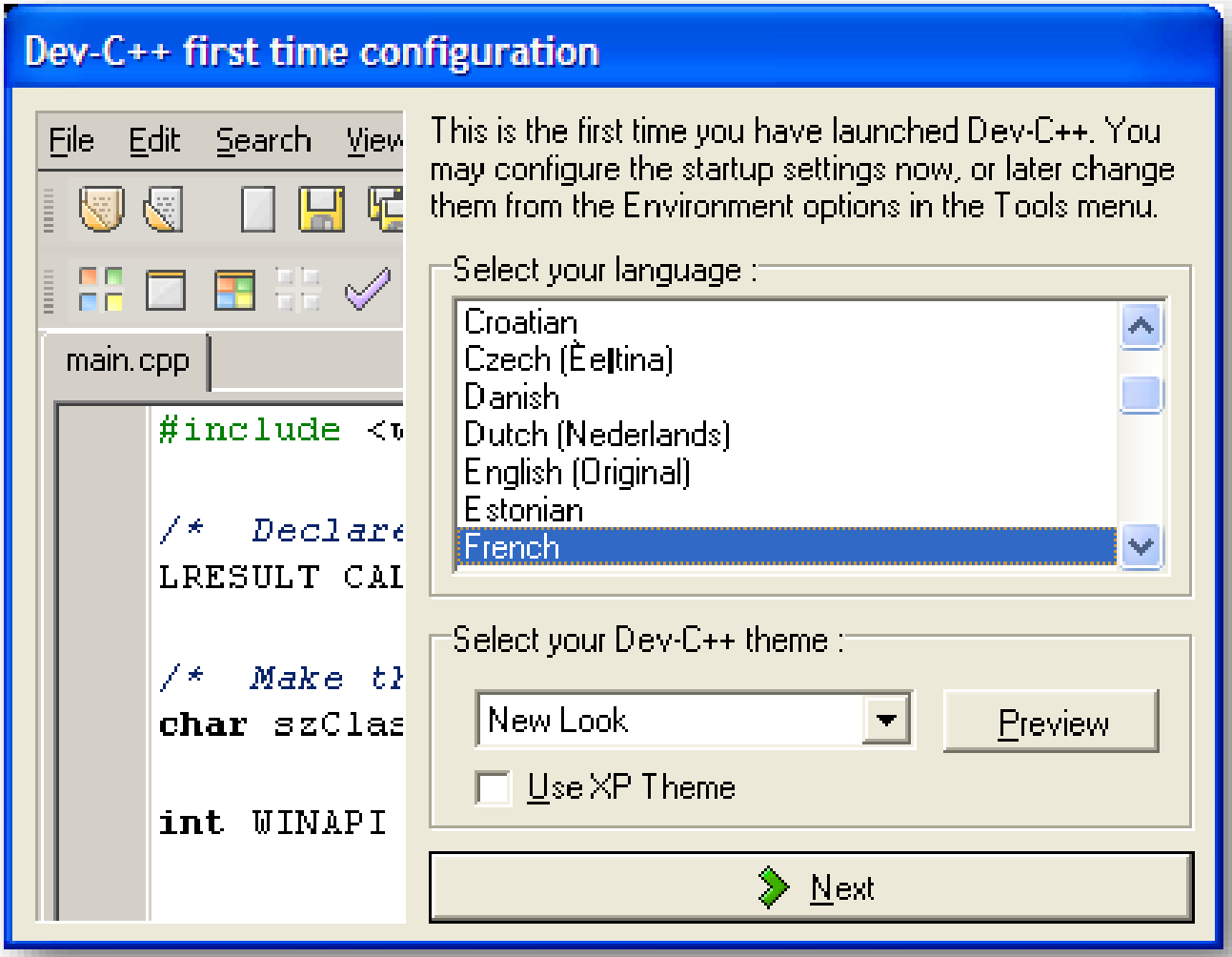
- Pour pouvoir réaliser des programmes en C, il est nécessaire de s'appuyer sur un système d'exploitation.
- Le système d'exploitation utilisé est ici Windows. Néanmoins, la quasi-totalité de ce qui est décrit ici peut être réalisée en utilisant d'autres systèmes d'exploitation.

Utiliser un éditeur sous Windows

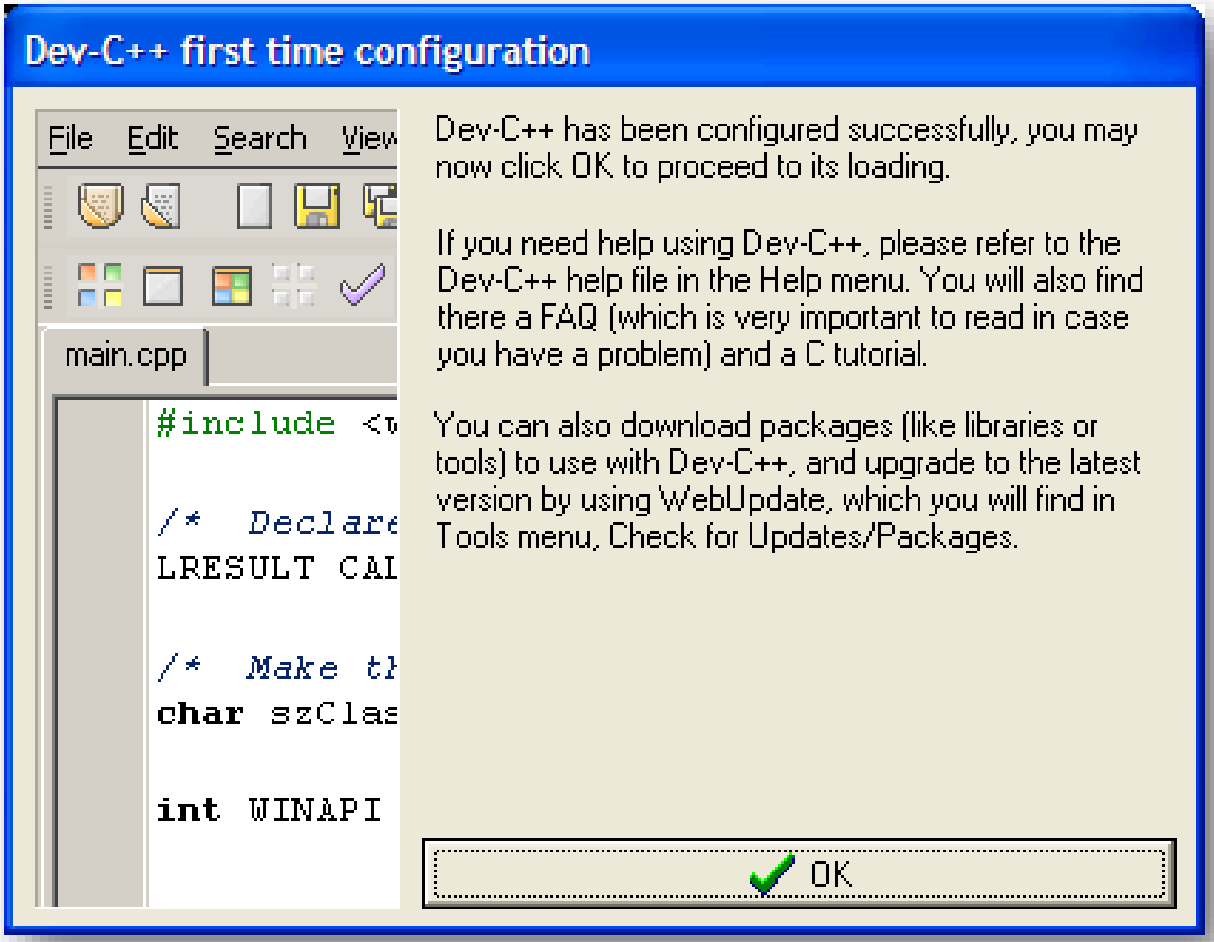


- Utiliser un environnement de développement intégré (EDI), par exemple Dev-C++,
- le programmeur n'appelle pas explicitement les commandes cc, mais elles sont appelées par l'EDI.
- La phase de Compilation se fait à la volée par l'EDI (il y a donc bel et bien une compilation qui est faite implicitement !) lorsque des changements ont été effectués sur les fichiers sources.

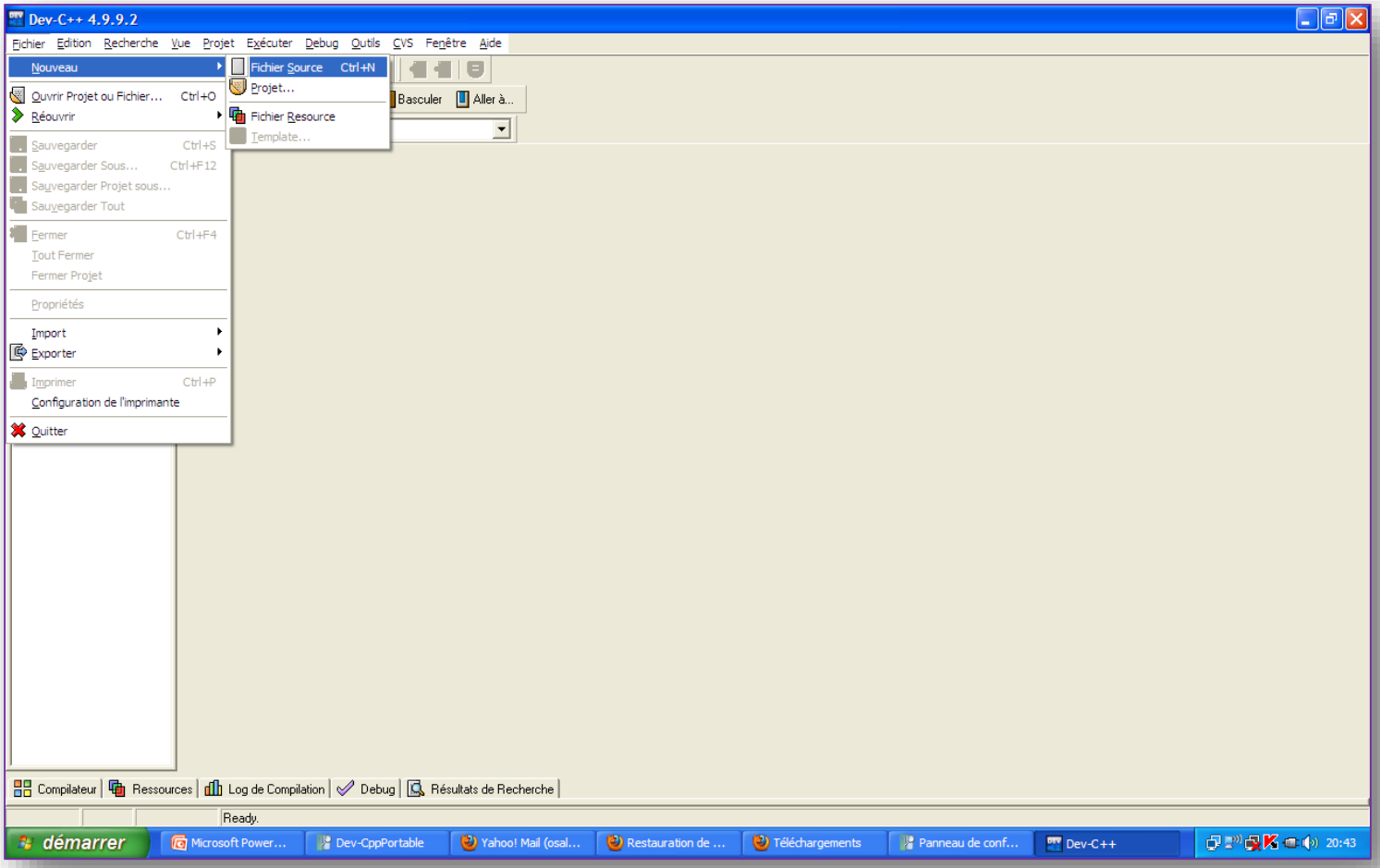
Dev-C++ Installation



Dev-C++ Installation



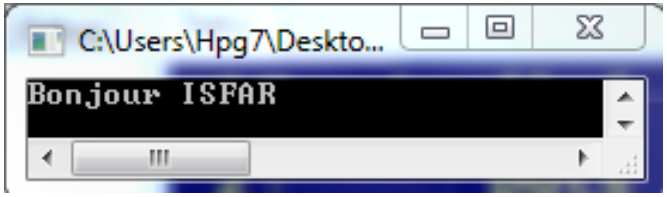
Dev-C++ Utilisation



Exemple de programme

- Voici un premier programme. Il est fonctionnel, même s'il n'est pas normalisé
- Il affiche le mot Bonjour à l'écran. À l'aide de votre IDE, tapez le texte du cadre suivant :

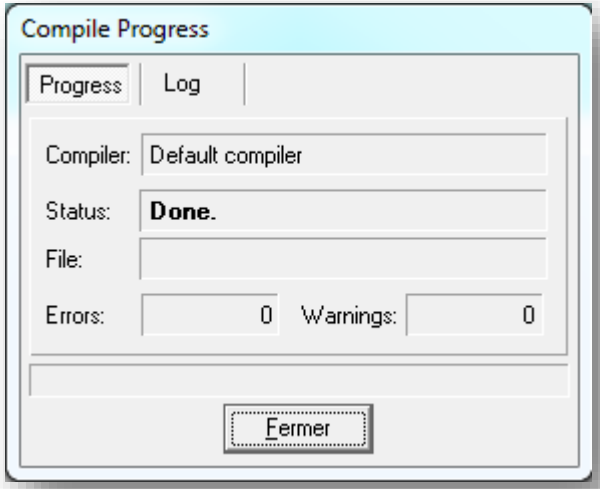
```
main () {
    puts ("Bonjour ISFAR");
    getchar ();
}
```



Puis, sauvegardez ce fichier (raccourci clavier : CTRL + S) sous le nom suivant : *programme1.c*
Une fois le texte du programme tapé, il faut le compiler, c'est-à-dire le transformer en programme exécutable (raccourci clavier : CTRL + F9) .

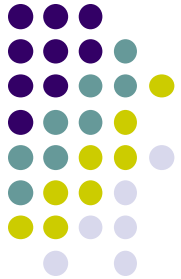
Exemple de programme

- Si vous n'avez pas fait d'erreurs, la ligne précédente provoquera l'affichage d'une fenêtre semblable à celle-ci (pas de nouvelle, bonne nouvelle. . .)



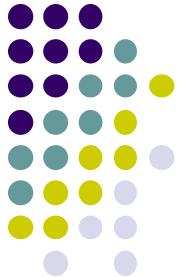
Le fichier programme1.c est un **« fichier source »**. Un fichier source désigne un fichier qu'un être humain peut comprendre par opposition à un exécutable que seule la machine arrive à comprendre. Il ne reste plus qu'à exécuter le programme pour cela appuyer sur la touche F9.
La machine affichera alors Bonjour et attendra que vous appuyiez sur la touche « entrée ».
Remarquez l'apparition d'un nouveau fichier **programme1.exe** qui est un « fichier exécutable».

Normalisation du programme



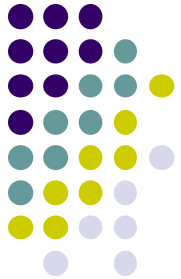
- Jusqu'à présent, nous avons fait un peu « dans le sale ». Nous nous en rendrons compte en demandant au compilateur d'être plus bavard. Lancez la commande :

Normalisation du programme



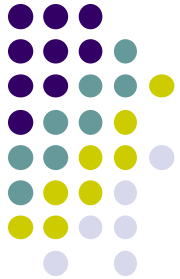
- Le langage C n'est qu'un nombre restreint d'instructions et un ensemble de bibliothèques. Le compilateur y trouve les fonctions et les applications qui lui permettent de créer un programme exécutable.
- Certaines bibliothèques (les plus courantes) sont incluses par défaut, ce qui permet à notre programme de se compiler. La fonction puts est stockée dans la bibliothèque standard d'entrées-sorties, incluse par défaut.
- L'utilisation d'une bibliothèque nécessite que nous informions le compilateur de notre souhait de l'utiliser : il suffit d'ajouter #include <fichier en-tête bibliothèque> en début de programme.

Normalisation du programme



- Ainsi, puisque nous utilisons la fonction puts, qui est dans la librairie standard d'entrées/sorties, nous indiquerons en début de programme 3 : #include <stdio.h>
- Un autre point à corriger est l'ajout de la ligne return 0. Tout programme doit renvoyer une valeur de retour, tout à la fin. Cette valeur de retour permet de savoir si le programme que l'on exécute s'est correctement terminé.
- En général 0 signifie une terminaison sans erreur. Enfin, il faut transformer la ligne main () en int main(). Ce point sera détaillé par la suite lorsque nous parlerons des fonctions...

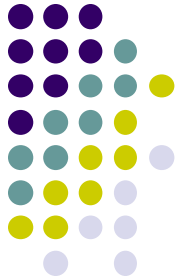
Normalisation du programme



- En rajoutant ces quelques correctifs nous obtenons donc :

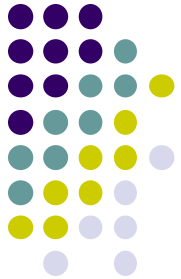
```
1 #include<stdio.h>
2 int main(){
3     puts ("Bonjour ISFAR");
4     getchar ();
5     return 0;
6 }
```

Petit mot sur ce qu'est une bibliothèque



- À l'instar de l'étudiant qui recherche dans des livres, nous pouvons dire que le « .h » représente l'index du livre et le « .c » le contenu du chapitre concerné.
- Après avoir lu (et retenu) le contenu des fichiers .h inclus, si le compilateur rencontre l'appel à la fonction *puts*, il est en mesure de vérifier si la fonction figurait dans un des *include*. Si ce n'est pas le cas, il émettra un avertissement.

Un exemple de fichier en-tête



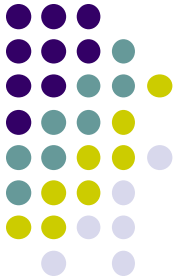
- Appuyer sur la touche contrôle(CTRL) de votre clavier et cliquer sur la ligne avec le texte #include<stdio.h>
- Vous allez ouvrir le fichier stdio.h. On y retrouve notamment la déclaration de puts (en dernière ligne de l'extrait) que nous venons de mentionner et la déclaration de printf que nous verrons dans les chapitres suivants. C'est assez compliqué. . . on y jette juste un oeil, pas plus.)

Compléments

```c
#include<stdio.h>
int main(){
    puts ("Bonjour ISFAR");
    getchar ();/* Permet d'attendre la frappe d'une touche */
    return 0;
}
```

- puts : permet d'afficher du texte suivi d'un retour à la ligne.
- getchar : permet d'attendre la frappe d'une touche suivie d'une validation par la touche ENTREE , ou un simple appui sur la touche ENTREE .
- /* Commentaire */ : met en commentaire tout le texte compris entre /* et */ 1. On trouvera aussi // qui permet de mettre le reste de la ligne courante en commentaire.
- Notre programme affiche donc Bonjour et attend que l'on appuie sur la touche Entrée ou sur une autre touche puis la touche Entrée.

Squelette de programme



- On peut définir le squelette d'un programme C de la façon suivante :

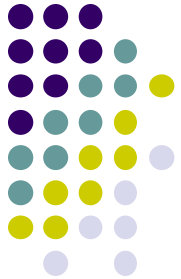
```c
/* Déclaration des fichiers d'en-têtes de bibliothèques */
#include<stdio.h>
int main () {
    /* Déclaration des variables (cf. chapitres suivants...) */
    /* Corps du programme */
    getchar(); /* Facultatif mais permet d'attendre l'appui d'une touche */
    return 0; /* Aucune erreur renvoyée */
}
```

Blocs

- La partie de programme située entre deux accolades est appelée un bloc. On conseille de prendre l'habitude de faire une tabulation après l'accolade. Puis retirer cette tabulation au niveau de l'accolade fermante du bloc.
- Ainsi, on obtient :

```
1 int main () {
2     Tabulation
3     Tout le code est frappé à cette hauteur
4 }
5 Retrait de la tabulation
6 Tout le texte est maintenant frappé à cette hauteur.
```

Commentaires



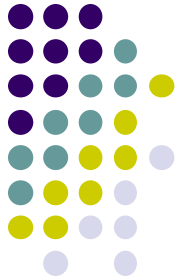
- Bien commenter un programme signifie qu'une personne ne connaissant pas votre code doit pouvoir le lire et le comprendre. Les commentaires sont indispensables dans tout bon programme. Ils peuvent être placés à n'importe quel endroit. Ils commencent par **/*** et se terminent par ***/** :

```
/* Commentaire */
```

- Comme nous l'avons déjà mentionné, vous trouverez aussi parfois des commentaires C++ :

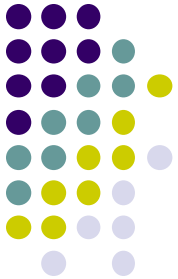
```
// Le reste de la ligne est un commentaire
```

Exercice d'application



- Écrivez un programme qui :
  - affiche « Salut toi, appuie sur une touche s'il te plaît » ;
  - attend l'appui d'une touche ;
  - affiche : « Merci d'avoir appuyé sur une touche ».
- Une fois que vous serez satisfait de votre solution, vous pourrez la comparer avec la solution qui apparaît un peu plus loin.

Correction Exercice d'application



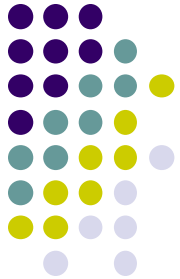
```c
#include <stdio.h>
int main () {
    /* Affiche premier message */
    puts ("Salut toi, appuie sur une touche s'il te plaît");
    getchar (); /* Attend la frappe d'une touche */
    /* Affiche le second message */
    puts ("Merci d'avoir appuyé sur une touche");
    return 0;
}
```

À retenir

- Il serait souhaitable de :
  - Se souvenir que l'éditeur que l'on utilise dans ce cours Dev-C++ et de son fonctionnement;
  - Connaître les fonctions puts et getchar qui apparaissent dans le programme suivant :

```
1 #include <stdio.h>
2 int main () {
3     puts ("Bonjour ISFAR");
4     getchar (); /* Permet d'attendre la frappe d'une touche */
5     return 0;
6 }
```

  - Savoir compiler un code source de programme : CTRL+F9
  - Savoir exécuter un programme avec F9

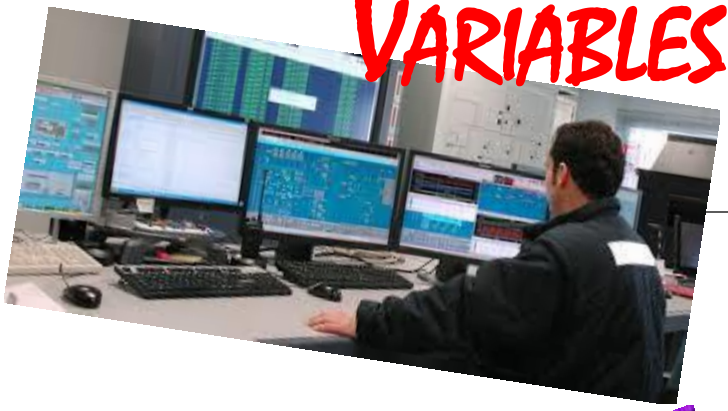


Pour aller plus loin…

- Tapez "cours langage c" sur GOOGLE http://www.google.sn/
- Le site http://developpez.com
- Algorithmique,…
  - http://www.siteduzero.com/tutoriel-3-51781-algorithmique-pour-l-apprenti-programmeur.html
  - http://www.siteduzero.com/tutoriel-3-14668-concevez-votre-site-web-avec-C-et-mysql.html
  - Tapez "cours Algorithmique" sur GOOGLE http://www.google.sn/
- …

Programmation en Langage C

Variables – Partie 1

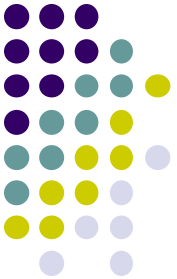


Pr. SALL Ousmane

Cours de Programmation

Cours n°3

Objectif



- Afin de stocker des valeurs, calculer, effectuer un traitement quelconque, il est nécessaire d'enregistrer de manière temporaire des données.
- Cet enregistrement nécessite la déclaration d'un lieu de la mémoire qui servira à ce stockage : **une variable**.

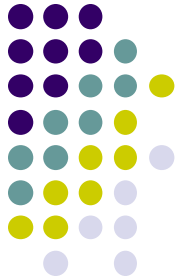
Affichage : la fonction printf

```c
#include <stdio.h>
int main () {
    /* Affiche Coucou c'est moi depuis ISFAR à l'écran puis saute une ligne */
    printf ("Coucou c'est moi depuis ISFAR\n");
    return 0;
}
```

- La fonction printf, tout comme puts vue précédemment, permet d'afficher une chaîne de caractères. Elle est cependant beaucoup plus puissante.

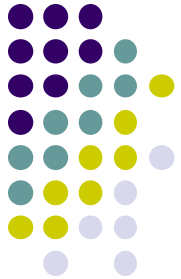
La syntaxe de printf est très complexe et pourrait à elle seule faire l'objet d'un chapitre, nous n'en verrons donc que des applications au fur et à mesure des besoins.

Notion de variable



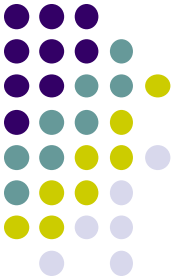
- Comme son nom l'indique, une variable est quelque chose qui varie.
- C'est vrai mais ce n'est pas suffisant. Une variable peut être considérée comme une boîte dans laquelle on met des données. Il est possible de lire le contenu de cette boîte ou d'écrire des données dans celle-ci.
- La manière la plus immédiate de lire le contenu d'une variable est de simplement mentionner son nom.
- La façon la plus simple d'affecter une valeur à une variable est l'opérateur d'affectation =.

Notion de variable



- Essayer d'utiliser une variable à laquelle nous n'avons encore affecté aucune valeur peut donner n'importe quel résultat (si on affiche le contenu d'une variable non initialisée par exemple, on pourra obtenir n'importe quelle valeur à l'écran).
- Affecter une valeur à une variable ayant déjà une valeur revient à la modifier. En effet, une variable ne peut contenir qu'une seule chose à la fois. Si vous mettez une seconde donnée dans la variable, la précédente est effacée.

Déclaration d'une variable



- La déclaration d'une variable se fait en utilisant la syntaxe suivante :

<son type> <son nom> ;

- Comme le montre le programme qui suit, il ne faut pas mettre les < et > comme cela apparaissait dans <son type> <son nom> ;.

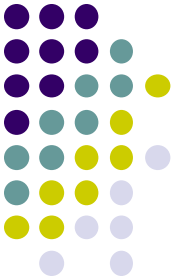
```c
#include <stdio.h>
int main () {
    int i; /* déclare un entier de nom i */
    char c; /* déclare un caractère de nom c */
}
```

Application : exemples(1)

- Premier exemple, avec des variables du type entier

```c
#include <stdio.h>
int main () {
    int i; /* i : variable de type entier */
    int j; /* j : variable de type entier */
    i=22; /* i vaut 22 */
    j=i; /* on recopie la valeur de i dans j */
    /* donc j vaut aussi 22 à présent */
    printf ("i vaut %d\n", i); /* Affiche la valeur de i */
    printf ("i+j vaut %d\n", i+j); /* Affiche la valeur de i+j */
    return 0;
}
```

Application : exemples(1)



- printf ("i vaut **%d**\n", i) ; : **%d** signifie que l'on attend une valeur entière et qu'il faut l'afficher en décimal (base 10). Le %d sera remplacé par la valeur de i. Cette ligne provoquera donc l'affichage suivant :

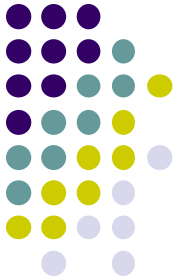
i vaut 22

- printf ("i+j vaut **%d**\n", i+j) ; : dans ce cas, **%d** est remplacé par la valeur de l'expression i+j. Nous obtiendrons l'affichage suivant : i+j vaut 44
- L'exécution complète de ce programme donne donc :

i vaut 22

i+j vaut 44

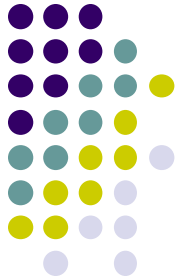
Application : exemples(2)



- Second exemple, avec des variables du type caractère

```c
#include <stdio.h>
int main () {
    char car; /* car: variable de type caractère */
    char f; /* f: variable de type caractère */
    car='E';
    f='e';
    printf("car=%c f=%c\n",car,f);
    return 0;
}
```

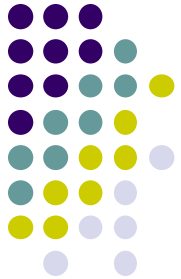
Application : exemples(2)



- car=’E’ : nous mettons dans la variable car la valeur (le code ASCII) du caractère E.
- f=’e’ : nous mettons dans la variable f la valeur (le code ASCII) du caractère ’e’.
- Notons au passage que, f=e signifierait affecter la valeur de la variable e (qui n’existe pas) à f. En oubliant les quotes (guillemets simples). ’...’
- nous aurions donc obtenu une erreur de compilation (variable inexistante).
- L’exécution complète de ce programme affiche donc :

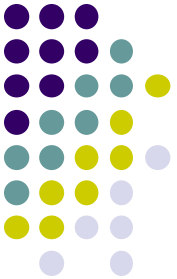
car=E f=e

Utilisation de % dans printf



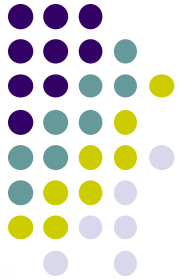
- À l'intérieur du premier paramètre d'un printf (appelé le format), l'emploi de « %y » signifie qu'à l'exécution, %y doit être remplacé par la valeur correspondante qui figure dans les paramètres suivants, après transformation de ce paramètre dans le type puis le format désigné par y. Nous avons à notre disposition plusieurs formats d'affichage, comme : %d, %x, %c. . .

Format d'une variable selon son type



- %d : entier (int)
- %c : caractère
- %f : virgule flottante (float ou double)
- %s : chaîne de caractères

Utilisation de % dans printf



- Exemple :

```
2 int main () {
3     int i;
4     i=65;
5     printf ("Le caractère %d est %c",i,i);
```

- Nous donnera l'affichage suivant :
  - Le caractère 65 est A
    - le %d est remplacé par la valeur numérique de i c'est-à-dire 65 ;
    - le %c est remplacé par la valeur alphanumérique (ASCII) de i c'est à-dire le caractère A. Cette table est très utile car l'ordinateur ne « comprend » que des nombres.
    - Elle donne une correspondance entre les lettres (que nous autres, êtres humains, comprenons) et leur codage par la machine. En résumé, chaque fois que nous manipulerons la lettre A, pour l'ordinateur, il s'agira de la valeur numérique 65. . .

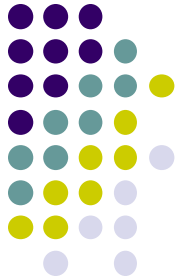
Exercices

Exercice d'application n°2: Ecrivez un programme qui à partir de la valeur du côté d'un carré donné, calcule et affiche son périmètre et sa surface.

Exercice d'application n°3: Ecrivez un programme qui à partir du rayon d'un cercle, calcule et affiche son diamètre, sa surface et sa circonférence.

Exercice d'application n°4: Ecrivez un programme qui à partir de la largeur et de la longueur d'un rectangle calcule et affiche la valeur de sa diagonale.

Exercices

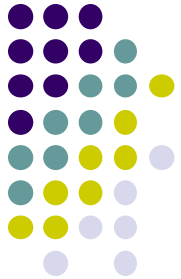


Exercice d'application n°5: Ecrivez un programme qui à partir du prix hors taxe PHT d'un produit et du taux de TVA calcule et affiche le prix toute taxe comprise PTTC

Exercice d'application n°6: Ecrivez un programme permettant de déclarer trois variables A, B, C de type réel, d'initialiser leurs valeurs et ensuite d'effectuer la permutation circulaire des trois variables.

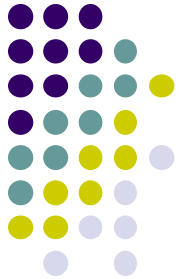
Exercice d'application n°7: Ecrivez un programme permettant de déclarer un entier, d'initialiser sa valeur, puis de calculer et afficher son opposé.

Exercices



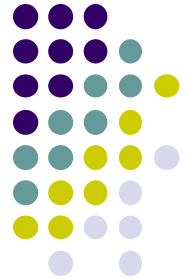
- Dans les exercices qui suivent, vous devez utiliser ce que nous venons de voir sur les variables, les caractères, et sur printf.
  - **Exercice n°1— Déclarer, afficher (a):**
    Déclarez des variables avec les valeurs suivantes 70, 82, 185 et 30 puis affichez le contenu de ces variables.
  - **Exercice n°2— Déclarer, afficher (b):**
    Faites la même chose avec les caractères c, o, u, C, O, U.

Réutilisation d'une variable



- Il est possible de réutiliser une variable autant de fois que l'on veut. La précédente valeur étant alors effacée :
  - i = 3;
  - i = 5;
  - i = 7; Maintenant, i contient 7, les autres valeurs ont disparu.
  - car = 'E' ;
  - car = 'G' ;
  - car = 'h' ; La variable car contient maintenant le caractère 'h'.

Caractères spéciaux



- Certains caractères (« % » par exemple) nécessitent d'être précédés par le caractère \ pour pouvoir être affichés ou utilisés.
  - Pour afficher un % avec printf, nous écririons :

```
3       printf("La réduction était de 20\%");
```

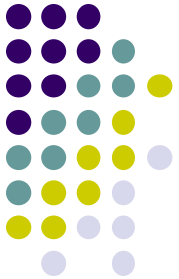
  - Pour désigner le caractère quote ' on écrira :

```
3       char car;
4       car = '\'';
```

  - En effet, le compilateur serait perdu avec une expression du type :

```
3       char car;
4       car = ''';
```

Exercices

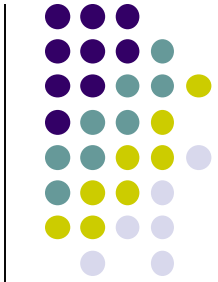


- Exercice n°3 — Erreur volontaire: Essayez d'écrire un programme contenant car=''' et constatez l'erreur obtenue à la compilation.
- Exercice n°4 — Okay !: Réalisez un programme qui permet d'obtenir l'affichage suivant :

```
C
'

e
s
t
Ok i vaut : 1
```

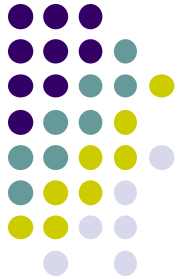
À retenir



- Exemples de types de variables :
  - char permet de définir une variable de type caractère.
  - int permet de définir une variable de type entier.
- Exemple de programme avec variables :

```c
#include <stdio.h>
int main () {
    char caractere;
    int entier;
    caractere = 'c';
    entier = 1;
    printf ("caractere vaut : %c\n",caractere);
    printf ("entier vaut : %d\n",entier);
    return 0;
}
```

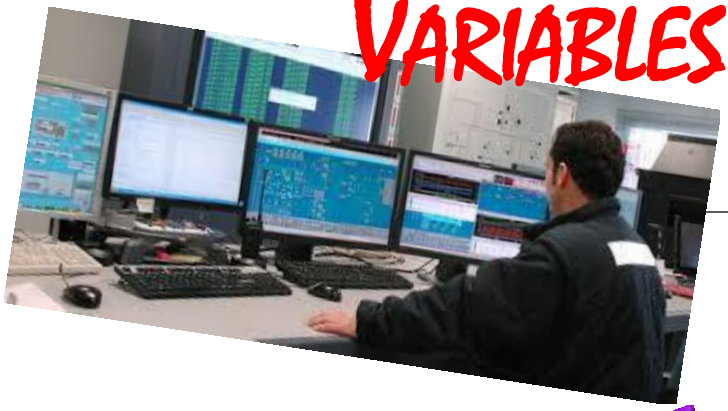
Pour aller plus loin…



- Tapez "cours langage c" sur GOOGLE http://www.google.sn/
- Le site http://developpez.com
- Algorithmique,…
  - http://www.siteduzero.com/tutoriel-3-51781-algorithmique-pour-l-apprenti-programmeur.html
  - http://www.siteduzero.com/tutoriel-3-14668-concevez-votre-site-web-avec-C-et-mysql.html
  - Tapez "cours Algorithmique" sur GOOGLE http://www.google.sn/
- …

Programmation en Langage C

Variables – Partie 2

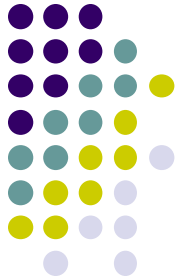


Pr. SALL Ousmane

Cours de Programmation

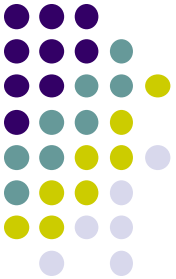
Cours n°4

Exercice de mise en bouche



- Exercice n°1— Introduction à une calculatrice
  - Écrivez un programme qui :
    - écrit « Calculatrice : » et saute 2 lignes. . .
    - écrit «Valeur de a : » et saute 1 ligne
    - attend l'appui d'une touche
    - écrit «Valeur de b : » et saute 1 ligne
    - attend l'appui d'une touche
    - écrit «Valeur de a + b : »
- Normalement, vous n'aurez pas de soucis pour l'écrire. . . comparez ensuite avec la solution en fin de chapitre. . .

Déclaration des variables

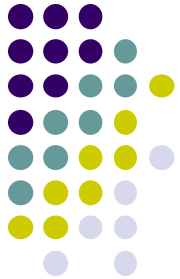


- Exercice n°2— Somme: Complétez le programme précédent en :
  - déclarant 2 variables a et b de type int (entier) ;
  - assignant à ces deux variables les valeurs 36 et 54 ;
  - faisant afficher le résultat de la somme de a+b (attention, n'écrivez pas le résultat 90 dans votre programme !).
- Pour faire afficher le résultat, il est possible d'utiliser la fonction printf en utilisant une troisième variable. Mais pour être plus concis, nous afficherons directement de la façon suivante :

```
3     printf ("Valeur de a + b : %d",a+b) ;
```

- %d sera remplacé par la valeur de l'expression a+b.

Saisie des variables

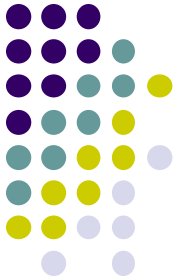


- Si une calculatrice électronique se limitait à calculer la somme de deux nombres fixes, le boulier serait encore très répandu.
- Pour saisir une variable, il est possible d'utiliser la fonction scanf. La fonction scanf s'utilise pour un entier de la façon suivante :

```
3    scanf ("%d", &a); // saisie de la valeur a
```

- Comme pour printf, nous reconnaissons le %d pour la saisie d'un nombre entier. Le & devant le a signifie que nous allons écrire dans la variable a.

Saisie des variables



- Nous allons maintenant saisir les variables a et b.
- Pour exemple, voici le code pour la saisie de a, la saisie de b reste à faire par vos soins à titre d'exercice.

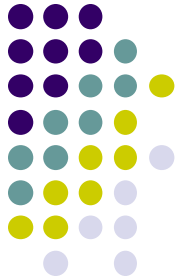
```
3       /* Saisie de la valeur de a */
4       printf ("Valeur de a :\n");
5       scanf ("%d", &a);
```

Saisie des variables

Il est conseillé d'initialiser les variables avant de les utiliser. Au début du programme, après leur déclaration, assignez la valeur 0 à a et à b.

```c
#include <stdio.h>
#include <math.h>
int main () {
    int a=0;
    int b=0;
    printf("Calculatrice :\n\n");
    printf("Valeur de a : \n");
    scanf("%d",&a);
    printf("\n");
    printf("Valeur de b : \n");
    scanf("%d",&b);
    printf("\nValeur de a+b : %d\n",a+b); /*Affichage de la somme*/
    getchar ();
    return 0;
}
```

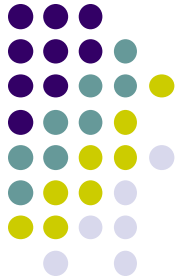
Exercices



- Exercice n°3 —Initialisation:

  Déclarez une troisième variable de type int (pensez à l'initialiser à 0) que nous nommerons simplement s comme somme. Une fois les valeurs de a et b saisies, initialisez s avec la valeur de a+b. Affichez la valeur de s. Nous devrions avoir les mêmes résultats qu'auparavant, bien sûr.

Exercices



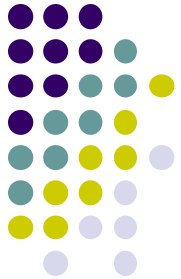
- Exercice n°4 —Obtenir des résultats

  Réalisez un programme qui calcule et affiche :

  - la soustraction et la multiplication de deux nombres saisies au clavier;

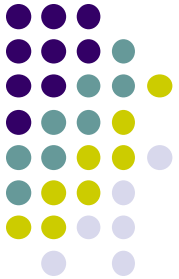
  Une fois que vous avez votre solution, comparez avec la correction proposée.

Les types flottants



- Nous allons étudier un nouveau type de données : les nombres à virgule flottante ou simplement flottants (float), qui permettent de représenter des nombres à virgule. Le type float permet de déclarer un tel nombre.
- Transformez les trois programmes précédents en utilisant le type float au lieu du type int.
- Enfin, si pour les int, nous utilisions le format %d au sein des printf et des scanf, à présent, nous allons utiliser le format %f pour les flottants.

Les types flottants



- Pour vous aider, voici un petit morceau de programme qui permet la saisie de la valeur de a et l'affiche :

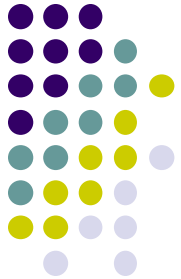
```
1 #include <stdio.h>
2 int main () {
3       float a;
4       printf("Saisie de a :");
5       scanf("%f",&a);
6       printf("\n a vaut : %f\n",a);
7       getchar ();
8       return 0;
9 }
```

Pour un affichage plus agréable il est possible de fixer le nombre de chiffres après la virgule de la façon suivante :

%.[nombre de chiffres après la virgule]f

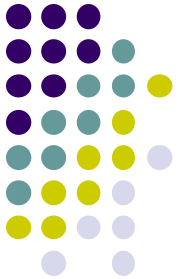
Voici un exemple : **printf ("%.2f",a) ;**

Exercices



- Exercice n°5 — Ouah, les 4 opérations !: Créez un quatrième programme qui réalise la division de deux nombres. Vous pourrez vous amuser à le tester avec une division par 0 ! La solution à ce petit problème sera vu dans le chapitre sur les conditions (if).

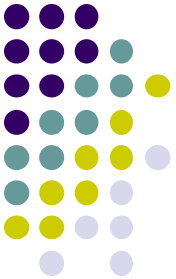
D'autres fonctions utiles avec <math.h>



La fonction abs permet d'obtenir la valeur absolue d'un nombre entier. La fonction fabsf permet d'obtenir la valeur absolue d'un float.

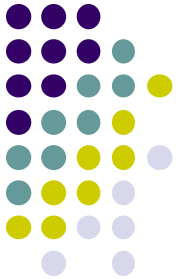
- Exercice n°6 —Absolument !
  - Utilisez cette dernière fonction pour calculer la valeur absolue de (a-b).
- Exercice n°7 —Arrondissez
  - La fonction ceilf permet d'obtenir l'arrondi entier supérieur d'un flottant. Utilisez cette fonction pour calculer l'arrondi supérieur de (a/b).

À retenir



```c
#include <stdio.h>
// ne pas oublier pour l'utilisation des fonctions mathématiques
#include <math.h>
int main () {
    float pi=3.14159;
    int i=10;// déclaration + initialisation
    int j;// déclaration seule (pas d'initialisation)
    // Attention, bien mettre %f et non pas %d
    printf ("pi vaut environ: %f",pi);
    printf("\n i vaut:%d\n",i);
    printf("\n entrez une valeur:");
    scanf("%d",&j);// Attention à ne pas oublier le &
    printf("\n vous venez d'entrer %d",j);
    return 0;
}
```

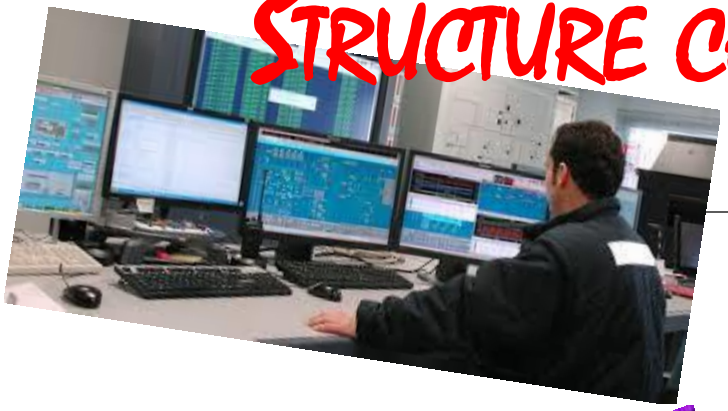
Pour aller plus loin…



- Tapez "cours langage c" sur GOOGLE http://www.google.sn/
- Le site http://developpez.com
- Algorithmique,…
  - http://www.siteduzero.com/tutoriel-3-51781-algorithmique-pour-l-apprenti-programmeur.html
  - http://www.siteduzero.com/tutoriel-3-14668-concevez-votre-site-web-avec-C-et-mysql.html
  - Tapez "cours Algorithmique" sur GOOGLE http://www.google.sn/
- …

Programmation en Langage C

STRUCTURE CONDITIONNELLE

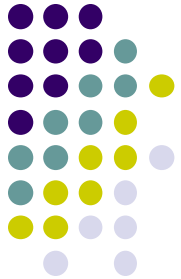


Pr. SALL Ousmane

Cours de Programmation

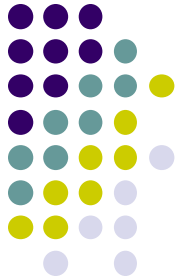
Cours n°5

Conditions



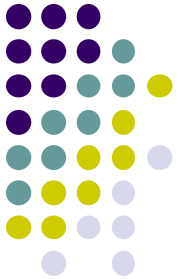
- Nous y sommes, nous savons saisir des nombres, des caractères, faire des calculs. . . Nous savons donc transformer notre ordinateur en une calculatrice. Comme la calculatrice, nous allons faire un gros BEEP, lors de la réalisation d'une division par 0. Il nous faut donc voir comment éviter ce schéma en apportant une réponse claire à ce problème. . .

Conditions - Objectif



- Dans ce chapitre, nous allons voir comment introduire des conditions dans nos programmes, de manière à ce que selon les circonstances, telle ou telle partie du code soit exécutée.

Exercice de mise en bouche

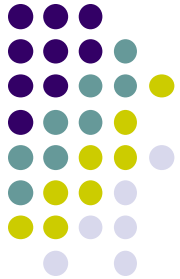


- Écrivez un programme qui met en application le théorème de Pythagore pour calculer la longueur de l'hypoténuse d'un triangle rectangle.
- Rappelons que dans un triangle rectangle, la longueur de l'hypoténuse (le plus grand côté) peut se calculer en appliquant la formule suivante :

$$\text{Longueur hypoténuse} = \sqrt{a^2 + b^2}$$

- où a et b sont les longueurs des deux autres côtés.
- La racine carrée s'obtient par l'utilisation de la fonction sqrt(valeur) contenue dans la bibliothèque de mathématiques (#include <math.h>)

Condition : Si Alors Sinon



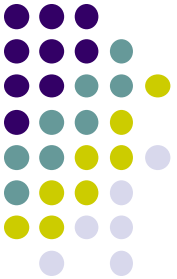
- En français, nous pourrions dire quelque chose de ce genre :

```
si (je travaille)
alors je progresserai
sinon je stagnerai
```

- En se rapprochant un peu du langage C, on traduirait (toujours en français) ce premier programme ainsi :

```
si (je travaille) {
        alors je progresserai
}
sinon {
        je stagnerai
}
```

Condition : Si Alors Sinon

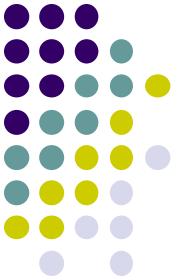


- Enfin, le programme en langage C ressemblera à ceci :

```
if (je travaille) {
    je progresserai
}
else{
    je stagnerai
}
```

- Les conditions s'expriment avec des opérateurs logiques dont nous allons expliquer tout de suite la signification et l'utilisation.

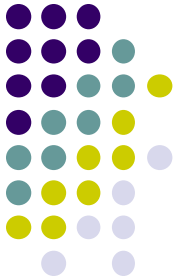
Opérateurs de comparaison



- Les opérateurs de comparaison servent à comparer deux nombres entre eux :

Signification	Opérateur
Inférieur	<
Supérieur	>
Égal	==
Différent	!=
Inférieur ou égal	<=
Supérieur ou égal	>=

Opérateurs de comparaison



- Voici un exemple de programme qui demande à l'utilisateur de saisir deux valeurs puis affiche la plus grande :

```c
#include <stdio.h>
int main () {
    int valeur1;
    int valeur2;
    /* Saisie de valeur1 */
    printf ("Entrez une 1ere valeur : ");
    scanf ("%d",&valeur1);
    /* Saisie de valeur2 */
    printf ("Entrez 2e valeur : ");
    scanf ("%d",&valeur2);
    if (valeur1<valeur2)
    printf("La plus grande valeur est: %d\n",valeur2);
    else
    printf("La plus grande valeur est: %d\n",valeur1);
    return 0;
}
```

Opérateurs logiques

- Les opérateurs logiques permettent de combiner des expressions logiques.

Libellé	Opérateur
Et (and)	&&
Ou (or)	||
Non (not)	!

- « | » se nomme en anglais un pipe (prononcer païpe). Des exemples suivront bientôt. . .

Vrai ou faux

- La valeur Vrai peut être assimilée à la valeur numérique 1. En programmation C, la valeur Vrai est associée à toute valeur non nulle.
- La valeur Faux peut être assimilée à la valeur numérique 0.
- L'opérateur Ou (||) correspond alors à une addition :

Ou	Vrai	Faux
Vrai	Vrai	Vrai
Faux	Vrai	Faux

Vrai ou faux

- L'opérateur Et (&&) correspond alors à une multiplication

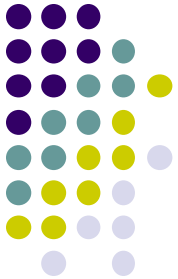
Et	Vrai	Faux
Vrai	Vrai	Faux
Faux	Faux	Faux

Vrai ou faux

- L'opérateur Not (!) permet d'obtenir la négation, ainsi :
  - !(Vrai) = Faux
  - !(Faux) = Vrai

```c
#include <stdio.h>
int main () {
    int i1=1;
    int i2=0;
    printf("i1 || i2 = %d",i1||i2);
    /* affichera 1 car : vrai||faux=vrai et vrai vaut 1 */
    printf("i1 && i2 = %d",i1&&i2);
    /* affichera 0 car : vrai&&faux=faux et faux vaut 0 */
    printf("contraire(1)=%d",!(1));
    /* affichera 0 car : !(vrai)=faux et faux vaut 0 */
    return 0;
}
```

Combinaison



- Toutes les opérations logiques peuvent se combiner entre elles. Il faut néanmoins veiller aux différentes priorités des opérateurs et il faut que la condition dans sa totalité soit entourée de ().
- Les exemples suivants montrent ce qui est possible et ce qui ne l'est pas :

Correct	`if (car == 'a')`
Incorrect	`if car == 'a'`
Correct	`if ( car == 'a' || car == 'A')`
Incorrect	`if (car == 'a') || (car == 'A')`
Correct	`if ((c == 'a' || c == 'A') && (c2 == 'G'))`
Incorrect	`if (c == 'a' || c == 'A') && (c2 == 'G')`

Attention !!!

- Vous verrez souvent ce type de code écrit :

```
3     if (err) {
4        /* Alors faire quelque chose */
5     }
```

- En appliquant ce qui a été vu précédemment, on en déduit que ce code signifie que

```
3     si (err != 0) /* si err différent de 0 */
4     {
5      /* Alors faire quelque chose */
6     }
```

- Ce qui donne en langage C :

```
3     if (err != 0) { /* si err différent de 0 */
4        /* Alors faire quelque chose */
5     }
```

Accolades

- Les accolades entourant les blocs d'instructions d'une condition peuvent être omises si le bloc n'est constitué que d'une seule instruction.

```
3     /* VERSION LOURDE : */
4     if (car == 'b') {
5        printf ("car vaut b.");
6     }else {
7           printf ("car est différent de b.");
8     }
9     /* VERSION LEGERE : */
10    if (car == 'b')
11       printf ("car vaut b.");
12    else
13        printf ("car est différent de b.");
14    }
```

Accolades

- Au contraire, dans l'exemple ci-dessous, il faut impérativement mettre des accolades !

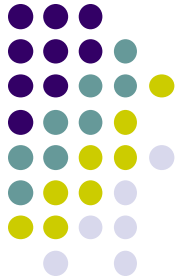
```
if (car == 'b') { /* il y a 2 instructions, il faut →
    ↪  donc mettre des { } */
  printf ("car vaut "); /* 1ère instruction */
  printf(" %c",car); /* 2ème instruction */
}
else
  printf ("car est différent de b.");
```

Exercices

- Exercice n°2 —Variable positive, négative ou nulle:
  Faites saisir une variable de type entier et indiquez à l'utilisateur si celle-ci est strictement positive, strictement négative ou nulle. Votre code contiendra quelque chose comme ceci :

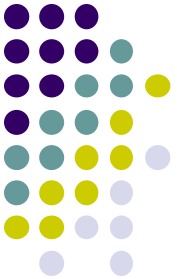
```
3     if (a > 0)
4        printf ("Valeur positive");
5     else
6        printf ("Valeur négative");
```

Exercices



- Exercice n°3 —Voyelles, consonnes
  Faites saisir une variable de type caractère et indiquez à l'utilisateur si celle-ci est une voyelle ou une consonne. On considérera que le caractère saisie est en minuscule.
- Notez que si le caractère saisie est une lettre et n'est pas une voyelle, c'est nécessairement une consonne.

À retenir



- La valeur Vrai peut être assimilée à la valeur numérique 1 ou à toute valeur non nulle.
- La valeur Faux peut être assimilée à la valeur numérique 0.
- Ne pas oublier les parenthèses lorsqu'il y a un if :

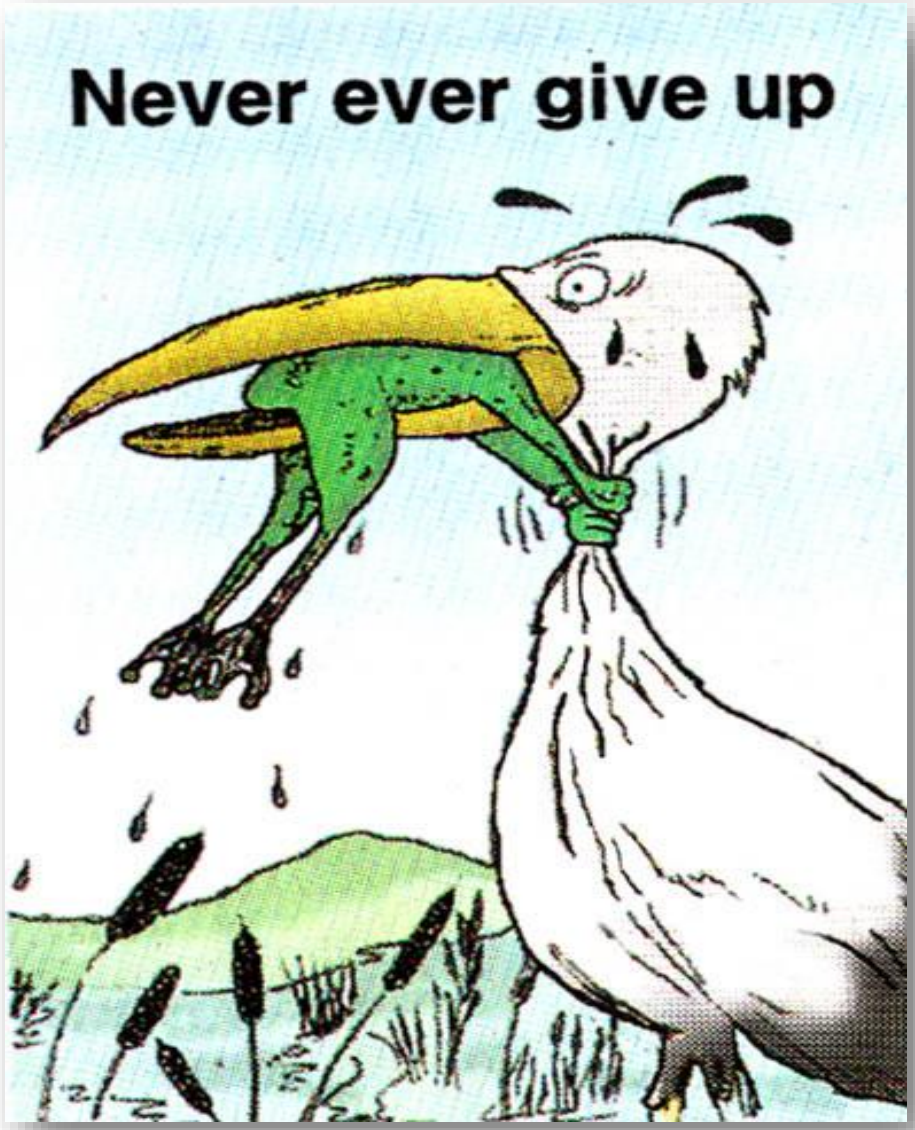
```
3     if a > 0 // ne sera pas compilé !!!
4        printf ("Valeur positive");
5     else
6         printf ("Valeur négative");
```

- au contraire, il faudrait écrire :

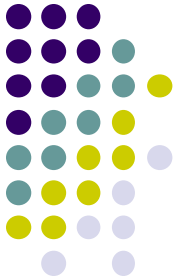
```
3     if (a > 0)
4        printf ("Valeur positive");
5     else
6         printf ("Valeur négative");
```

- Différencier l'opérateur d'affectation = et l'opérateur de comparaison ==.

Dans tous les cas…



Pour aller plus loin…



- Tapez "cours langage c" sur GOOGLE http://www.google.sn/
- Le site http://developpez.com
- Algorithmique,…
  - http://www.siteduzero.com/tutoriel-3-51781-algorithmique-pour-l-apprenti-programmeur.html
  - http://www.siteduzero.com/tutoriel-3-14668-concevez-votre-site-web-avec-C-et-mysql.html
  - Tapez "cours Algorithmique" sur GOOGLE http://www.google.sn/
- …

Programmation en Langage C

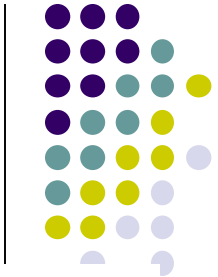
Mise Au Point

Pr. SALL Ousmane

Cours de Programmation

Cours n°6

Structure Conditionnelle: Exercices Complémentaires

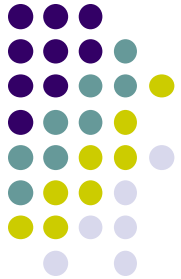


Exercice n°4 — Prix à payer

Ecrivez un programme qui, à partir de la saisie du prix unitaire d'un produit (PU) et de la quantité commandée (QTCOM), affiche le prix à payer (PAP) en détaillant le port (PORT) et la remise (REM), sachant que:

- le port est gratuit si le prix des produits (TOT) est supérieur à 5 000 F. Dans le cas contraire, le port est de 2% de TOT;
- la remise est de 5% si TOT est compris entre 2 000 et 10 000 F et de 10% au-delà.

Structure Conditionnelle: Exercices Complémentaires

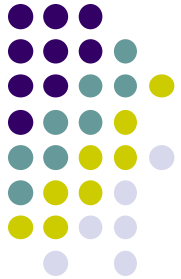


Exercice n°5 — Année Bissextile

Écrire un programme permettant de savoir si une année saisie par l'utilisateur est bissextile ou non.

Rappel : une année est bissextile si elle est divisible par 400 ou bien par 4 mais non divisible par 100.

Structure Conditionnelle: Exercices Complémentaires



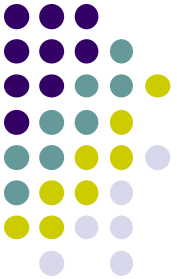
Exercice n°6 — Date du lendemain

Écrire un programme permettant de déterminer la date du lendemain en fonction du jour, mois et année d'une date donnée saisie par l'utilisateur.

Exemple:

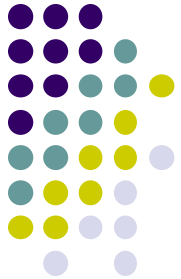
- Date saisie: 31 12 2013
- Date du lendemain: 01 01 2014

Objectif



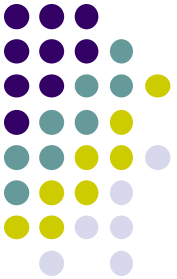
- L’objet de ce chapitre est de réaliser une pause dans l’apprentissage du C et de s’attacher à ce que l’on est capable de réaliser avec le peu de moyens que l’on a.
- Ce chapitre est constitué de trois premiers exercices de difficulté croissante.
- Ils nous permettront d’apprendre à utiliser une nouvelle fonction et de réaliser un exercice complet de programmation.

Plus petit ou plus grand



- Exercice n°1— Plus grand ou plus petit que... Réalisez un programme qui saisit un nombre puis indique à l'utilisateur si le nombre saisi est plus grand ou plus petit qu'un autre nombre défini à l'avance dans le programme.

Retour sur getchar()



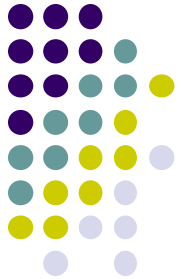
- La fonction getchar() permet d'attendre la frappe d'un caractère au clavier, de le lire et de le renvoyer. Deux utilisations peuvent donc être faites de getchar(),
  - la première est celle permettant d'attendre la frappe d'une touche sans se soucier de sa valeur, simplement pour marquer une pause.

```
8        getchar();
```

  - la seconde est celle permettant de lire un caractère au clavier.

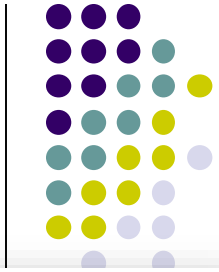
```
3        char car;
4        car = getchar();
```

Boucle : Faire . . . Tant que (condition)



- do ... while (traduisez par Faire . . . Tant que) permet de réaliser une suite d'instructions tant qu'une ou plusieurs conditions sont remplies.
  1. lisez le programme à la diapositive suivante
  2. lisez les explications qui figurent en dessous du programme
  3. exécutez, testez, comprenez ce programme...

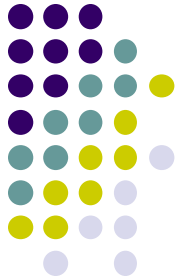
Boucle : Faire . . . Tant que (condition)



```c
#include <stdio.h>
int main () {
    char car=' ';
    int sortie=0;
    do {
        printf ("Appuyez sur S pour sortir !\n");
        car = getchar ();
        /* On le compare pour savoir si l'on peut sortir: */
        sortie = ((car == 's') || (car == 'S'));
    }while (sortie==0);
    return 0;
}
```

- Pour stopper un programme qui boucle, il faut presser simultanément sur les touches CTRL + C (break).

Exercice

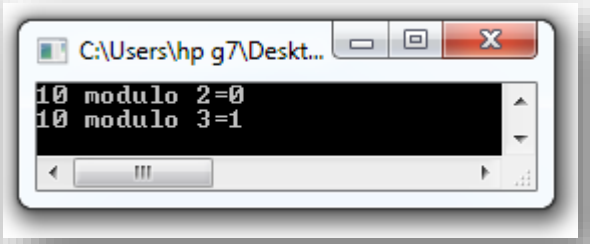


- Exercice n°2— Sortir de la boucle
  En vous inspirant de l'exemple précédent, écrivez un programme de telle sorte que l'on ne sorte de la boucle que lorsque l'utilisateur a saisi le nombre 10.

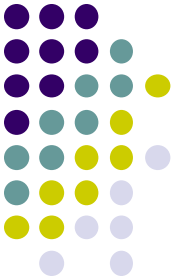
Opérateur modulo

- L'opérateur qui donne le reste de la division entière (opérateur **modulo**) est noté % en C. Ainsi, 10%2 vaut 0 car la division entière de 10 par 2 donne 5 et il n'y a pas de reste. En revanche, 10%3 vaut 1 car la division entière de 10 par 3 donne 3 et il reste 1.

```
3       int z;
4       z=10%2;
5       printf("10 modulo 2=%d\n",z);
6       z=10%3;
7       printf("10 modulo 3=%d\n",z);
```



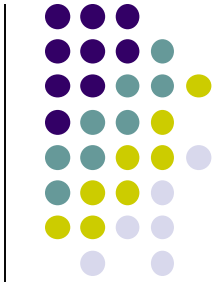
Nombres pseudo-aléatoires



- Voici un petit exemple de programme qui permet d'obtenir des nombres pseudo-aléatoires entre 0 et 99 :

```c
#include <stdio.h>
#include <stdlib.h> // sert pour les fonctions srand et rand
#include <time.h>
int main() {
    int nb_alea=0;
    /* Initialisation du générateur de nombres
    basée sur la date et l'heure */
    srand (time (NULL));
    nb_alea = rand() % 100;
    printf ("Nombre aleatoire : %d\n",nb_alea);
    getchar();
    return 0;
}
```

Nombres pseudo-aléatoires

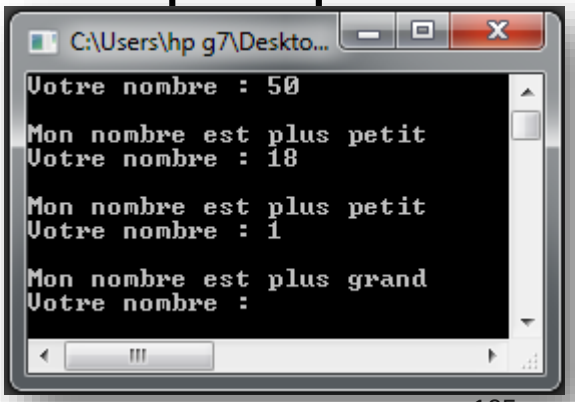


- srand (time (NULL)) permet d'initialiser le générateur de nombres pseudo-aléatoire. Nous reviendrons sur ce point par la suite.
- rand() renvoie un nombre entier compris entre 0 et RAND_MAX.
- rand()%100 est donc le reste de la division entière d'un nombre pseudo-aléatoire (éventuellement très grand) par 100, c'est-à-dire un nombre compris entre 0 et 99...

```c
#include <stdio.h>
#include <stdlib.h>
int main() {
    /* Pour notre information */
    printf ("RAND_MAX : %ld\n", RAND_MAX);
    return 0;
}
```

Exercices

- Exercice n°3— Deviner un nombre
  En vous aidant de ce qui a été fait précédemment, réalisez un petit jeu qui :
  - Initialise un nombre entre 0 et 99.
  - Tente de faire deviner ce nombre à l'utilisateur en lui indiquant si le nombre à trouver est plus petit ou plus grand que sa proposition.



Boucle While

- De la même façon que :
  - do {...} while(condition);
- il est possible d'utiliser :
  - while(condition) {…}

Donnée seule, la ligne : while(condition); signifie tant que condition est vraie, on revient à la même ligne soit sur soi-même. Si la condition est toujours vraie, on tombe alors dans un puits (le programme reste bloqué). Sinon, on passe immédiatement à la ligne/instruction suivante.

```
5       char car = ' ';
6       while ((car != 's') && (car != 'S')) {
7               car = getchar ();
8       }
```

Exercices

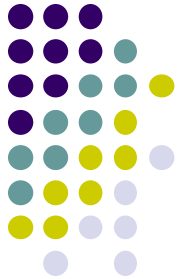
- Exercice n°4 —Touche-touche
  Traduisez en langage C, complétez avec les variables nécessaires, compilez, exécutez, comprenez :

```
Faire
      Saisir une touche
Tant Que (touche != S) et (touche != s)
```

- Exercice n°5 —Saisir un nombre
  Traduisez en langage C, complétez avec les variables nécessaires, compilez, exécutez, comprenez :

```
Faire
      Saisir un nombre
Tant Que (nombre != 10)
```

Fonction toupper()



- Le problème de la comparaison de la minuscule et de la majuscule de l'exercice 4 peut être contourné par l'utilisation de la fonction toupper qui transforme un caractère minuscule en majuscule.

- Pour l'utiliser, il faut inclure le fichier d'en-tête ctype.h par : #include <ctype.h>.

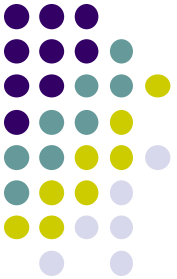
Fonction toupper()

- La fonction toupper s'utilise de la façon suivante :

```c
#include <stdio.h>
#include <ctype.h>
int main () {
    char car;
    char car_min;
    car_min = 'a';
    car = toupper (car_min);
    printf ("%c",car);
    return 0;
}
```

- Ce programme affichera : A

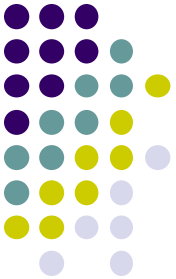
Exercices



- Exercice n°6 — Recommencez ! (a)
  Écrivez le programme : Tant que je ne tape pas un nombre impair compris entre 1 et 10 je recommence la saisie d'un nombre.

- Exercice n°7 — Recommencez ! (b)
  Écrivez le programme : Tant que je ne tape pas une voyelle je recommence la saisie d'une touche.

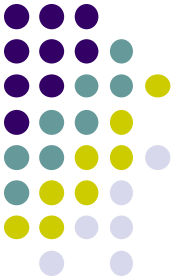
Notion de compteur



- Dans les exercices qui suivent, la notion de compteur intervient. Un compteur est une variable numérique que l'on décrémente (-1) ou incrémente (+1) suivant nos besoins.
- Par exemple :

```
4       int i;
5       i=1;
6       i=i+1;
7       printf("i vaut: %d",i); // affichera 2
```

Notion de compteur

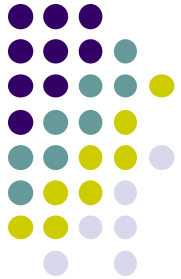


- Pour gagner du temps, le langage C nous permet de remplacer une expression comme : i=i+1 par l'expression suivante : i++ qui fera exactement la même chose.

```
4       int i;
5       i++; /* Incrémente le compteur i */
6       i--; /* Décrémente le compteur i */
```

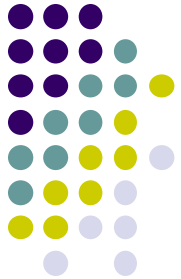
- Dans les exemples précédents, le nombre de caractères entrés peut donc être comptabilisé en ajoutant 1 à une variable à chaque fois qu'une touche est frappée.

Exercices



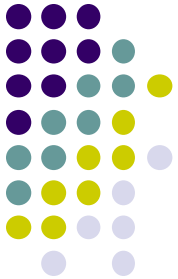
- Exercice n°8 —Recommencez ! (c)
  Écrivez le programme : Tant que je n'ai pas saisi 10 nombres, je recommence la saisie d'un nombre.
- Exercice n°9 —Recommencez ! (d)
  Écrivez le programme : Tant que je n'ai pas saisi 10 caractères, je recommence la saisie d'un caractère.

Exercices



- Exercice n°10 —Recommencez ! (e) Écrivez le programme : Tant que je n'ai pas saisi 10 voyelles, je recommence la saisie d'une touche. Vous prendrez soin d'indiquer à l'utilisateur combien de voyelles il lui reste à entrer.
- Exercice n°11— Recommencez ! (f) Écrivez le programme : Tant que je n'ai pas saisi 10 chiffres premiers (2,3,5 ou 7), je recommence la saisie d'un chiffre. Vous prendrez soin d'indiquer à l'utilisateur combien de chiffres premiers il lui reste à entrer.

À retenir



- Voici un exemple de programme qui résume ce qui a été vu dans ce chapitre. Ce programme doit afficher à l'écran tous les nombres pairs inférieurs à 100 :

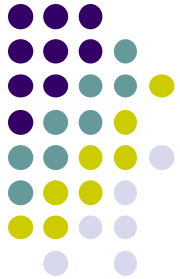
```c
#include <stdio.h>
int main () {
    int i=0;
    while (i != 100) {
        if (i%2==0) /* reste de la division de i par 2 */
        printf("%d ",i);
        /* pas de else ni de {} ici, c'est inutile...*/
        i++;
    }
    return 0;
}
```

À retenir

- Voici une autre version (meilleure) :

```c
#include <stdio.h>
int main () {
    int i=0;
    while (i!=100) {
        printf("%d ",i);
        i+=2;
    }
    return 0;
}
```

Pour aller plus loin…



- Tapez "cours langage c" sur GOOGLE http://www.google.sn/
- Le site http://developpez.com
- Algorithmique,…
  - http://www.siteduzero.com/tutoriel-3-51781-algorithmique-pour-l-apprenti-programmeur.html
  - http://www.siteduzero.com/tutoriel-3-14668-concevez-votre-site-web-avec-C-et-mysql.html
  - Tapez "cours Algorithmique" sur GOOGLE http://www.google.sn/
- …

Programmation en Langage C

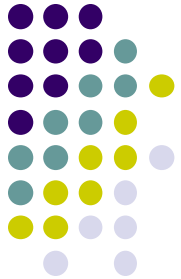
BOUCLES

Pr. SALL Ousmane

Cours de Programmation

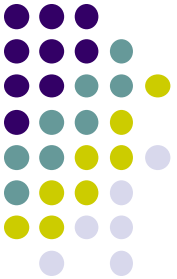
Cours n°7

Objectifs



- Désormais, nos programmes s'arrêtent maintenant quand nous le désirons, nous saisissons des chiffres, des lettres et savons comment les reconnaître. Ces méthodes sont suffisantes, mais parfois d'autres peuvent être utiles.
- Dans ce chapitre, nous étudierons la possibilité de réaliser des actions un certain nombre de fois.

Boucles



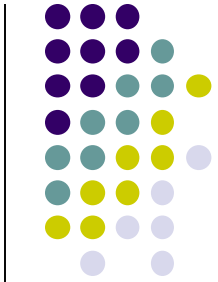
- Afin d'effectuer un certain nombre de fois une tâche, nous utilisons l'instruction for de la façon suivante (avec i, une variable de type entier (int par exemple)) :

```
for (i=point de départ; i<point d'arrivée; i=i+pas) {
    instruction(s) répétées(s);
}
```

- Ceci est rigoureusement équivalent à :

```
3       i=point de départ;
4       while (i<point d'arrivée) {
5           instruction(s) répétées(s);
6           i=i+pas;
7       }
```

Boucles



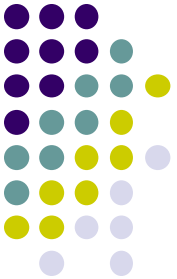
- Par souci de simplicité, nous dirons simplement que la suite d'instructions suivante :

```
for (i=0; i<15; i++) {
    instr;
}
```

- signifie que l'on va exécuter instr pour i variant de 0 à 14 inclus (<15) c'est-à-dire 15 fois. Par exemple :

```
3   int i;
4   for (i=0; i<15; i++){
5       printf("Je me répète pour i valant %d\n",i);
6   }
7   printf("L'instruction a été répétée... 15 fois\n");
8   return 0;
```

Syntaxe



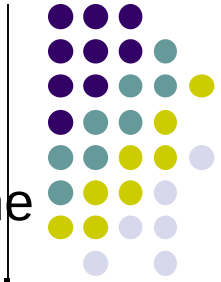
- De la même façon que le if, le for ne nécessite pas d'accolades s'il n'y a qu'une instruction à répéter.
- Nous pouvons utiliser cette fonctionnalité dans le programme précédent en remplaçant:

```
for (i=0; i<15; i++) {
    printf("Je me répète pour i valant %d\n",i);
}
```

- Par

```
for (i=0; i<15; i++)
    printf("Je me répète pour i valant %d\n",i);
```

Notion de double boucle



- Il est possible de remplacer les instructions contenues dans une boucle par une autre boucle afin de réaliser une double boucle. Nous obtenons donc :

```
Pour i allant de ... à ... {
  ...
  Pour j allant de ... à ... {
    ...
  }
}
```

```
3     int i;
4     int j;
5     for (i=0; i<5; i++){
6         printf("\nJe suis dans la boucle i, i vaut %d\n",i);
7         for (j=3; j>0; j--)
8             printf("Je suis dans la boucle j, j vaut %d\n",j);
9     }
10    return 0;
```

Exercices

- Exercice n°1— Étoiles
  En utilisant une double boucle, écrire un programme qui affiche une étoile, passe à la ligne, affiche deux étoiles, passe à la ligne, affiche trois étoiles. . . jusqu'à cinq étoiles afin d'obtenir ceci :

```
*
**
***
****
*****
```

Exercices

- Exercice n°2— Sapin
  À l'aide d'une double boucle, réalisez un cône pour dessiner le haut du sapin sur 10 lignes.

Aide :
- sur la ligne no 1, afficher 9 espaces puis 1 étoile ;
- sur la ligne no 2, afficher 8 espaces puis 3 étoiles ;
- sur la ligne no 3, afficher 7 espaces puis 5 étoiles ;
- sur la ligne no i, afficher 10-i espaces puis 2*i-1 étoiles.

Exercices

Exercice n°3 — Nombres dans un intervalle

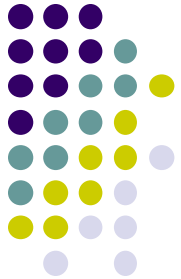
Écrire un programme permettant de saisir puis d'afficher une valeur comprise entre 1 et 31 ; on recommencera la saisie jusqu'à ce que la valeur soit bien dans les bornes imposées.

Exemple : valeurs saisies 43

résultat affiché : valeur non comprise entre 1 et 31

recommencez… Valeur saisie 15 , affichage 15 ok !

Exercices



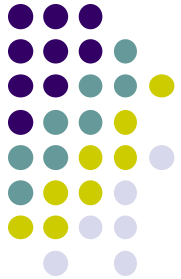
Exercice n°4 — Table de multiplication

Écrire l'programme d'un programme permettant d'afficher la table de multiplication d'un nombre saisi par l'utilisateur. Exemple :

valeur saisie 5

résultat affiché : 0 5 10 15 20 25 30 35 40 45 50

Exercices



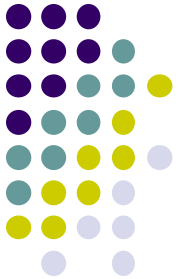
Exercice n° 5 — Somme de nombres impairs

Écrire un programme permettant de calculer la somme des nombres impairs de 1 à n. Quel lien pouvez-vous établir entre la valeur obtenue et n ?

Exercice n°6 — Carré de nombres pairs

Écrire un programme permettant de saisir 20 nombres au clavier et d'afficher le carré des nombres pairs uniquement. Attention, on ne mémorisera pas les 20 valeurs saisies.

Pour aller plus loin…



- Tapez "cours langage c" sur GOOGLE http://www.google.sn/
- Le site http://developpez.com
- Algorithmique,…
  - http://www.siteduzero.com/tutoriel-3-51781-algorithmique-pour-l-apprenti-programmeur.html
  - http://www.siteduzero.com/tutoriel-3-14668-concevez-votre-site-web-avec-C-et-mysql.html
  - Tapez "cours Algorithmique" sur GOOGLE http://www.google.sn/
- …