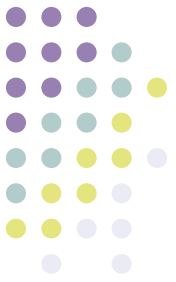
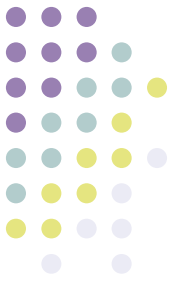


Cours de Bases de la programmation



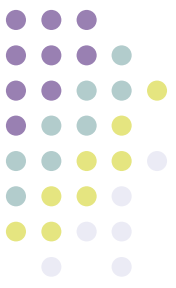
- Modalités de Contrôle :
 - Ecrit1 : 30 %
 - Ecrit2 : 30 %
 - TP : 20 %
 - Mini-projet:20 %
- C++

Ordinateur



- Architecture de John von Neumann et ses collaborateurs.
- Les ordinateurs sont structurés en unités séparées, fonctionnellement différentes :
 - l'unité de calcul (Unité Arithmétique et Logique)
 - l'unité de contrôle
 - la mémoire interne (programme et données)
 - les unités d'Entrées / Sorties).

Ordinateur



L'Unité Arithmétique et Logique Contient :

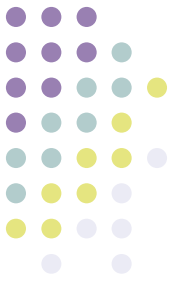
- Les registres qui sont des unités de stockage
- Les circuits de l'UAL (les calculs)
- Les voies de circulation dans l'UAL
 - (bus de commandes et de données)

L'Unité de contrôle

- Cherche dans la mémoire l'instruction suivante (pointeur ordinal)
- Envoyer les bonnes commandes à l'UAL, la mémoire et les contrôleurs d'entrées/sorties
- Mémoriser le résultat

L'ensemble des opérations qu'un processeur peut effectuer est appelé jeu d'instructions

Ordinateur



Le programme et les données sont stockés dans la mémoire :

- Les registres
- La mémoire cache
- La mémoire principale appelée RAM
- Disques durs

Les unités d'entrée/sorties : sous-système qui permet à l'ordinateur d'interagir avec d'autres périphériques et de communiquer avec le monde extérieur.

Le processeur regroupe l'UAL et l'unité de contrôle.

Ordinateur

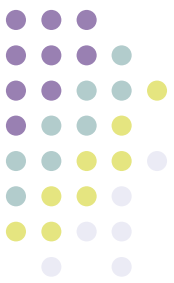
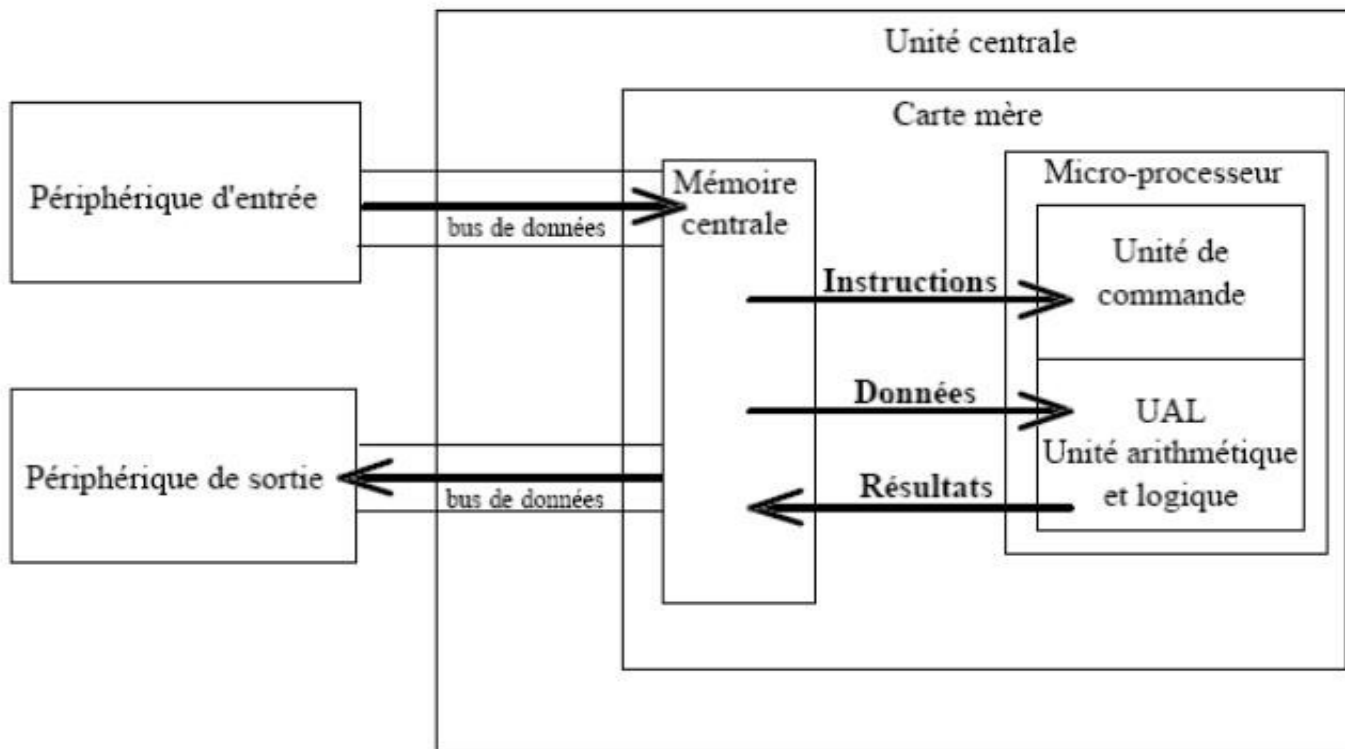
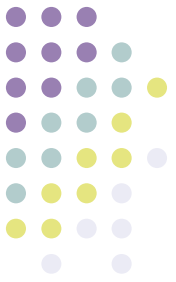


Schéma fonctionnel d'un ordinateur



Système d'exploitation



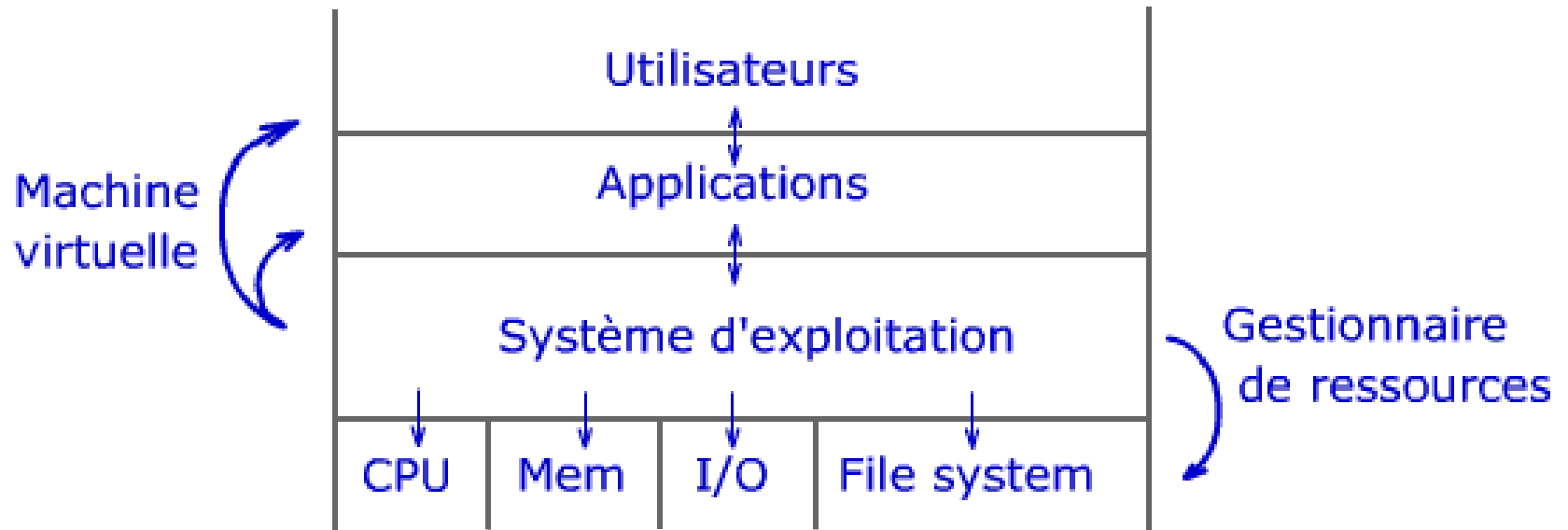
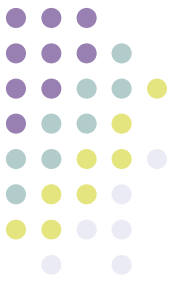
Le système d'exploitation

- Noté SE ou OS [Operating System],
- Chargé d'assurer la liaison entre les ressources matérielles, l'utilisateur et les applications.

Deux tâches :

- Fournir à l'utilisateur une machine étendue ou virtuelle
- Gérer les ressources.
- Deux dimensions de partage (multiplexage) :
 - temps
 - espace

Schéma : système d'exploitation

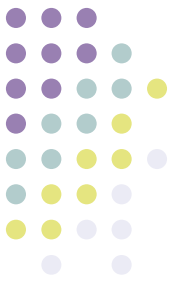


Système d'exploitation



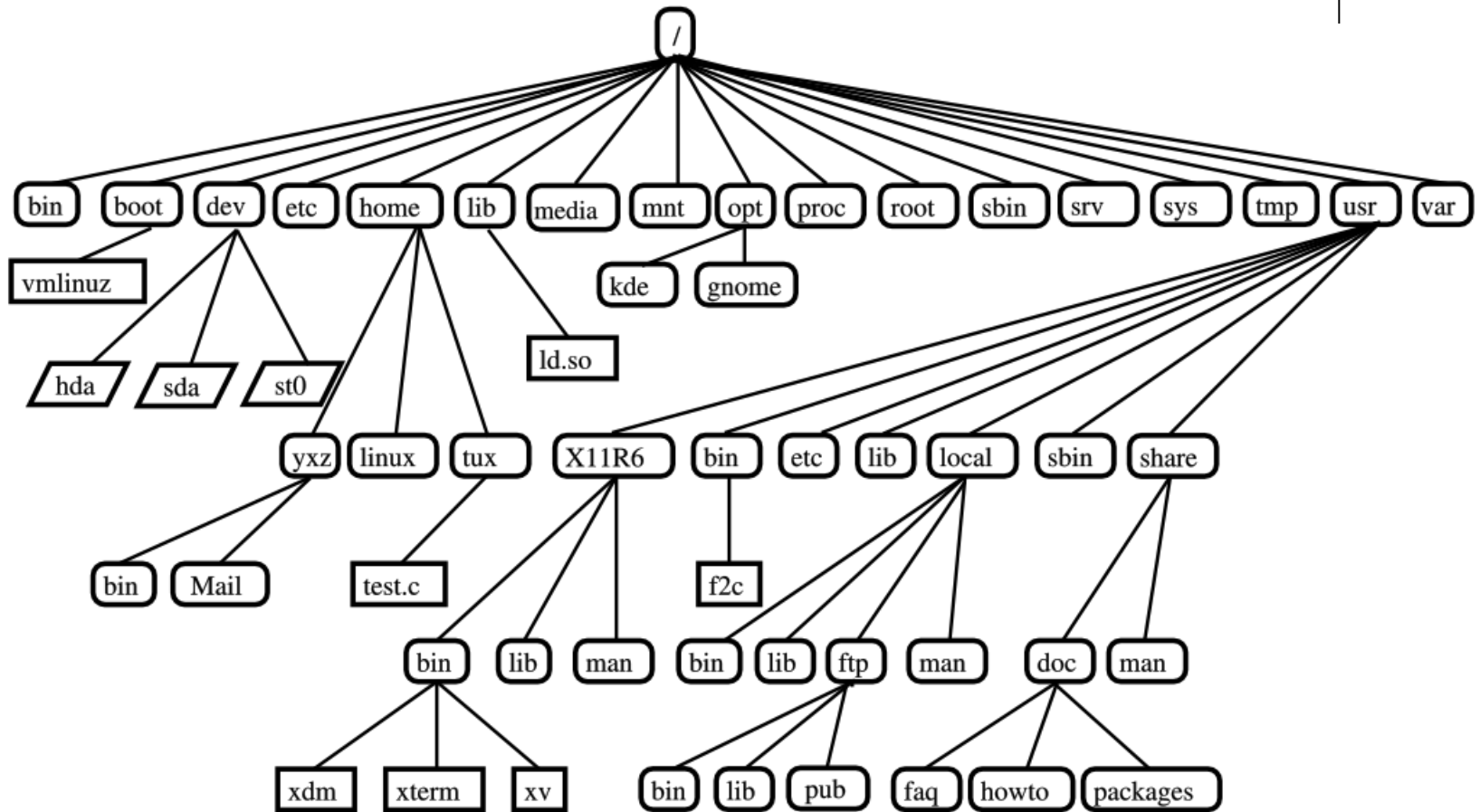
- C'est le chef d'orchestre
- l'OS permet la gestion de :
 - La CPU
 - La mémoire
 - Des fichiers
 - Des programmes en exécution
 - (les processus)
 - Des périphériques (la souris, l'écran, USB..)
 - Le réseau

Au ceri



- Pas de MAC-OS
- Des machines sous windows (stat7 et 8)
- Les autres machines sont sous linux :
 - C'est UNIX sur PC
 - Multi-tâches
 - Multi-utilisateurs
 - Gestion des droits
 - Très sécurisé
 - Libre

La gestion des fichiers sous Linux

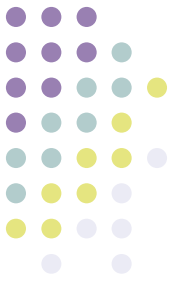


Terminal-console



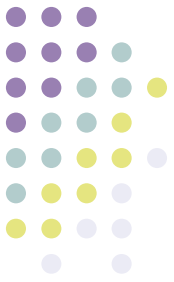
```
matrouf@pc-driss: ~/Programmation/TP1
Fichier Édition Affichage Rechercher Terminal Aide
matrouf@pc-driss:~$
matrouf@pc-driss:~$
matrouf@pc-driss:~$ pwd
/home/matrouf
matrouf@pc-driss:~$ mkdir REP1
matrouf@pc-driss:~$ cd REP1
matrouf@pc-driss:~/REP1$ ls
matrouf@pc-driss:~/REP1$ echo "bonjour" > toto
matrouf@pc-driss:~/REP1$ ls
toto
matrouf@pc-driss:~/REP1$ cd ..
matrouf@pc-driss:~$ mkdir Programmation
matrouf@pc-driss:~$ cd Programmation/
matrouf@pc-driss:~/Programmation$ mkdir TP1
matrouf@pc-driss:~/Programmation$ ls
TP1
matrouf@pc-driss:~/Programmation$ cd TP1
matrouf@pc-driss:~/Programmation/TP1$ ne
nemo                nemo-open-with      netkit-ftp           networkd-dispatcher
nemo-autorun-software neotoppm             netplan              newgrp
nemo-connect-server  neqn                 netstat              newusers
nemo-desktop         netcat               networkctl
matrouf@pc-driss:~/Programmation/TP1$ geany exo1.cpp &
[1] 25470
matrouf@pc-driss:~/Programmation/TP1$
```

Programmer



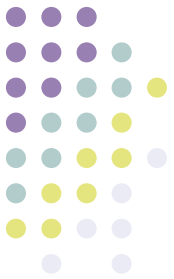
- Réaliser des programmes informatiques
- Votre ordinateur contient différents genres de programmes
 - La calculatrice
 - Le traitement de texte
 - Les jeux vidéo
 - Windows explorer
 - firefox
 - chrome
- Bref, un système informatique est constitué en grande partie de programmes et de données

Programmer



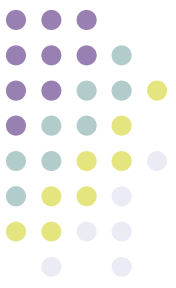
- De quoi avons-nous besoin ?
 - Un éditeur de texte
 - Vous permet de rédiger vos programmes
 - vi, Bloc-Notes
 - Compilateur :
 - ce fameux programme qui permet de traduire votre langage C/C++ en langage binaire !
 - La machine ne comprend le code source
 - Un débbugger pour vous aider à trouver les erreurs
- Les trois composants peuvent être intégrés : 3 en 1, IDE (Integrated Development Environment) ou Environnment de développement

Premier programme



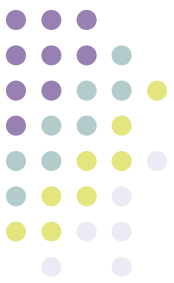
```
1 # include <iostream>
2 using namespace std ;
3 int main()
4 {
5     cout<<"Hello world!\n";
6     return(0);
7 }
8
```

- Laisser une ligne vide à la fin du fichier
- La ligne 1 commence par #, directives de préprocesseur
- include signifie inclure avant de procéder à la compilation
- iostream correspond à une bibliothèque ou librairie
- C'est la ligne 1: qui permet de reconnaître et de réaliser la ligne 5:
- La ligne 5: consiste à réaliser l'affichage
- Un programme est constitué en grande partie de fonctions
- Dans ce programme il y a une seule fonction sont appelée : main
- main (principale) est une fonction particulière, c'est par main que commence l'exécution d'un programme



Premier Programme

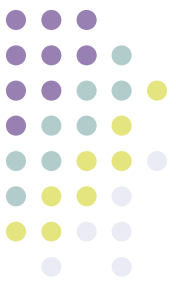
- Les lignes 5 et 6 s'appellent des instructions
- Chaque instruction est une commande à l'ordinateur
- Finalement un programme est une suite d'instructions → ordre donnés à l'ordi
- Chaque instruction se termine par un ;
- A quoi sert la ligne 6 : return (0)



Premier programme

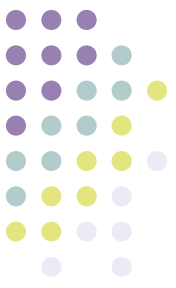
- On procède à la compilation:
 - Ouvrir une console
 - `g++ premier.cpp -o premier`
 - `premier`
Hello world !
- Avez-vous remarqué « `\n` » à la fin de Hello world, ceci ordonne à l'ordinateur de retourner à la ligne
- Pour afficher:
 - Bonjour
Comment ça va ?
 - `cout<< (" Bonjour\nComment ça va?\n ")`
- Les commentaires : `//` Sur une ligne
`/*` Ou sur
plusieurs
`*/`

Mémoire



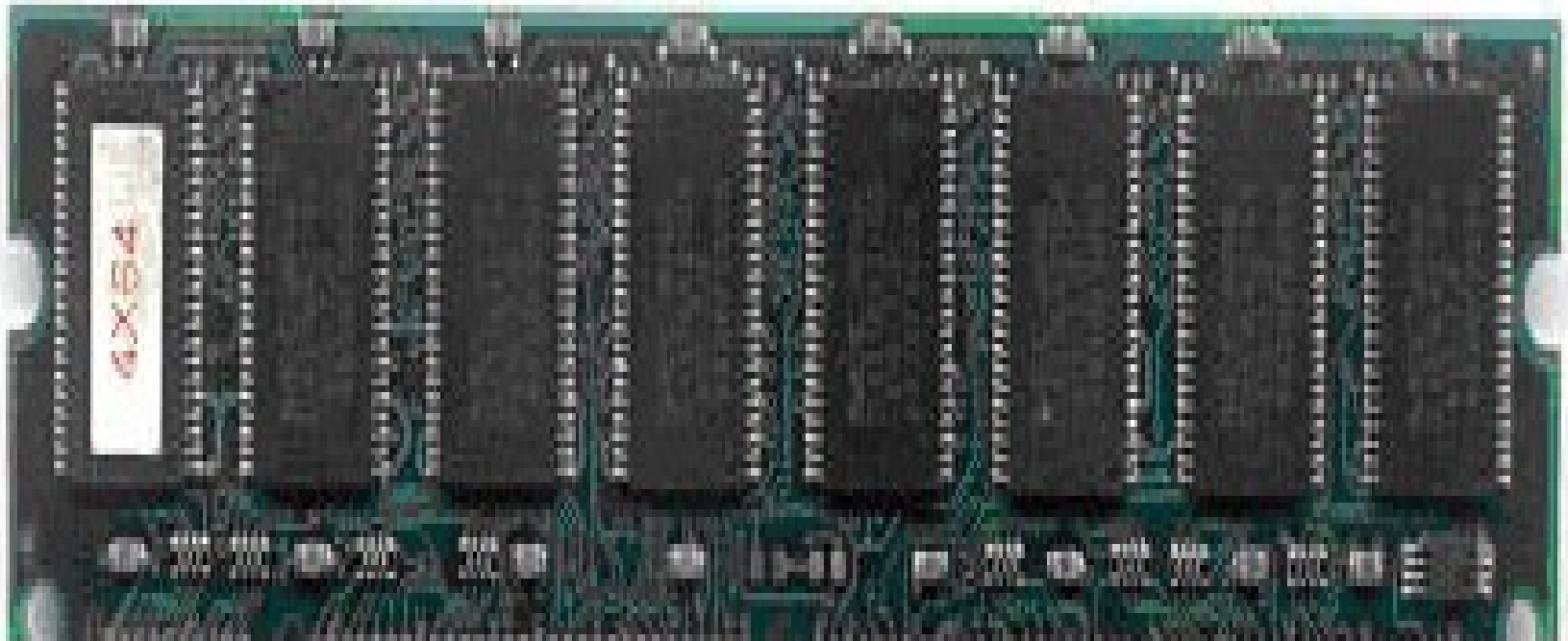
- **Les différents types de mémoire**
 - Les registres : une mémoire ultrarapide située directement dans le processeur.
 - La mémoire cache : elle fait le lien entre les registres et la mémoire vive.
 - La mémoire vive : c'est la mémoire avec laquelle nous allons travailler le plus souvent.
 - Le disque dur : que vous connaissez sûrement, c'est là qu'on enregistre les fichiers (8ms).

Mémoire

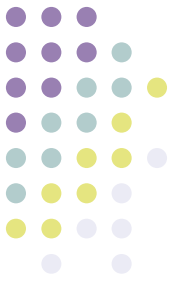


- Les registres sont utilisés par les langages bas niveau (assembleur)
- Les disques durs (mémoire lente) sont accessibles via les fichiers (plus tard)
- C'est dans la mémoire vive que nous allons travailler (langages haut niveau)
- Seuls les disques durs retiennent l'information : les autres mémoires sont temporaires (volatiles)

RAM : Mémoire vive

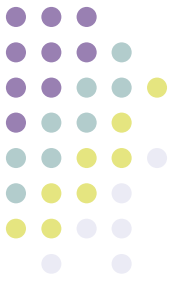


Mémoire vives

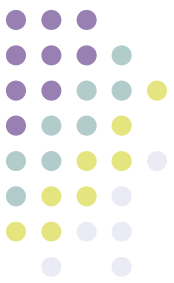


Adresses	Valeurs
0	143
1	3.14
3	12333
122356666	54

Variable

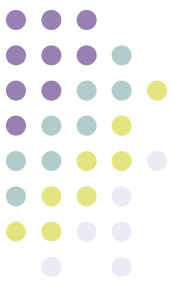


- Imaginer : on veut stocker l'information '3.14'
- L'ordinateur va mettre cette valeur en mémoire (libre) à l'adresse 3 par ex
- Plus tard, pour retrouver cette valeur, on va chercher à l'adresse 3
- Nous pouvons donner un nom à cette information : on vient de créer une variable.
- Une variable : nom et une valeur
- C'est le compilateur qui fait l'association entre le nom, l'adresse et l'emplacement physique de l'information



Nom de variable

- Comment choisir un nom ?
- Vaut mieux un nom qui porte du sens
- Il ne peut y avoir que des lettres minuscules et majuscules et des chiffres (abcABC012...).
- Doit commencer par une lettre.
- Les espaces sont interdits.
 - A la place, on peut utiliser le caractère "underscore" _ .
- Pas d'accents (éàê etc).
- En C/C++, a est différent de A



Types de variables

- Pour interpréter l'information stockée en mémoire, l'ordinateur a besoin d'y associer un type

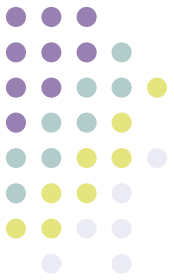
- char \Rightarrow 1 octets

0 1 1 0 1 1 1 0

- int \Rightarrow 4 octets

1 1 0 1 1 1 1 1 1 1 0 1 0 1 1 0 1 1 0 1

- long int \Rightarrow 8 octets
- float \Rightarrow 4 octets
- Double \Rightarrow 8 octets

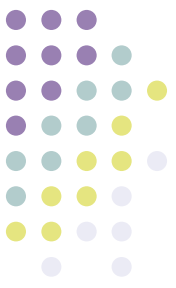


Déclarer une variable

- `int` nbreEtudiants; // nombre d'étudiants
- `unsigned double` distance; // entre villes
- `char` genre ; // m pour masculin, f ...

•

```
3 int main()  
4 {  
5     // Début de la fonction main  
6     int nbreEtudiants; //création d'une variable  
7     return 0;  
8     // Fin de la fonction  
9 }
```

Affecter une valeur à une variable

```
1 int nbreEtudiants;
```

```
2 nbreEtudiants = 20;
```

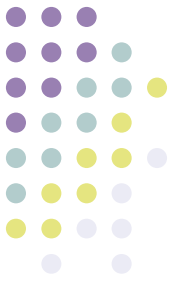
- Déclaration implique création physique en mémoire
- La valeur initiale est aléatoire

```
1 int nbreEtudiants = 20;
```

- Une constante est une variable dont on ne peut pas changer la valeur

```
1 const double PI = 3.14;
```

Dans un exemple C++



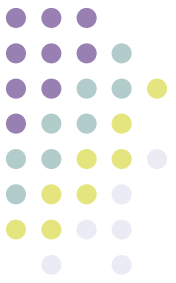
```
1 # include <iostream>
2
3 using namespace std ;
4 int main(int argc, char *argv[])
5 {
6     int age = 0; // On initialise la variable à 0
7     cout << "Quel age avez-vous ? ";
8     cin >> age; // On demande d'entrer l'age
9     cout << "Ah ! Vous avez donc " << age << "  ans ! \n\n";
10    return 0;
11 }
```

Quel age avez-vous ? 20
Ah ! Vous avez donc 20 ans

Les calculs de base



- L'ordinateur n'est pas plus intelligent qu'une vulgaire calculatrice
- Opérations de calcul basiques
 - Addition
 - Soustraction
 - Multiplication
 - Division
 - Modulo ($x\%y$ =le reste de la division de x par y)
- Les autres opérations sont programmées

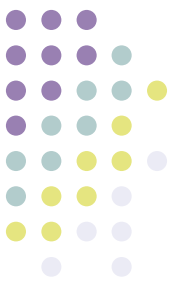


Les calculs de base

```
1 int resultat = 0;  
2 resultat = 5 + 3;  
3 cout<<"5 + 3 = "<< resultat<<endl;
```

$$5 + 3 = 8$$

- Addition : +
- Soustraction : -
- Multiplication : *
- Division : /
- Modulo : %

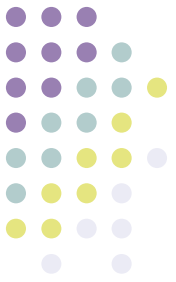


Les calculs de base

```
1 # include <iostream>
2 # include <stdlib.h>
   Using namespace std ;
3 int main(int argc, char *argv[])
4 {
5     int resultat = 0, nombre1 = 0, nombre2 = 0;
6     // On demande les nombres 1 et 2 à l'utilisateur :
7     cout<<"Entrez le nombre 1 : ";
8     cin >> nombre1;
9     cout<<"Entrez le nombre 2 : ";
10    cin >> nombre2;
11    // On fait le calcul :
12    resultat = nombre1 + nombre2;
13    // Et on affiche l'addition à l'écran :
14    cout<< nombre1<<'+'<< nombre2<<'='<< resultat<<endl;
15    return 0;
16 }
```

*Entrez le nombre 1 : 30
Entrez le nombre 2 : 25
30 + 25 = 55*

Les raccourcis



`i = i + 1;` \Leftrightarrow `i += 1;` \Leftrightarrow `i++;`

`i = i - 1;` \Leftrightarrow `i -= 1;` \Leftrightarrow `i--;`

`i = i * 2;` \Leftrightarrow `i *= 2;`

`i = i / 2;` \Leftrightarrow `i /= 2;`

`i = i % 3;` \Leftrightarrow `i %= 3;`

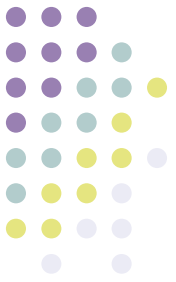
La bibliothèque mathématique



- En C/C++, l'existence des bibliothèques standards
- `<iostream>` est une bibliothèque concernant les entrées/sortie (E/S) :
- `Math.h` correspond à la bibliothèque mathématique standard

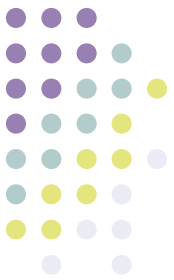
```
# include <math.h>
```

- `Exp, cos, log, fabs, sin, pow, sqrt,`



La condition "if... else"

- == Est égal à
 - > Est supérieur à
 - < Est inférieur à
 - >= Est supérieur ou égal à
 - <= Est inférieur ou égal à
 - != Est différent de
- == est différent de =
 - = est l'affectation
 - == est une comparaison



Un if simple

- SI la variable vaut ça
ALORS fais ceci

```
1 if (/* Votre condition */)
2 {
3     // Instructions à exécuter si la condition est vraie
4 }
```

```
1 if (age >= 18)
2 {
3     cout << ("Vous etes majeur !");
4 }
```

```
1 if (age >= 18) cout << "Vous etes majeur !" ;
```

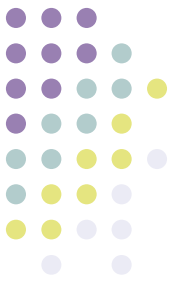
- Il y a une seule instruction → pas besoin d'accolades

Un if simple



```
1 # include <iostream>
2 # include <stdlib.h>
3 using namespace std;
4 int main(int argc, char *argv[])
5 {
6     int age;
7     cout << "Quel est votre age ? ";
8     cin >> age;
9     if (age >= 18)
10    {
11        cout << "Vous etes majeur !\n";
12    }
13    return 0;
14 }
```

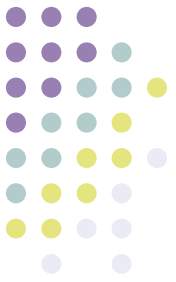
If ... else



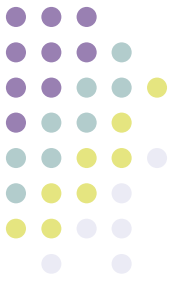
```
1 # include <iostream>
2 # include <stdlib.h>
3 using namespace std;
4 int main(int argc, char *argv[])
5 {
6     int age;
7     cout << "Quel est votre age ? ";
8     cin >> age;
9     if (age >= 18)
10    {
11        cout << "Vous etes majeur !\n";
12    }
13    else
14    {
15        cout << "Vous etes mineur !\n";
16    }
17    return 0;
18 }
```

if ... else if

```
1 # include <iostream>
2 # include <stdlib.h>
3 using namespace std;
4 int main(int argc, char *argv[])
5 {
6     int age;
7     cout << "Quel est votre age ? " ;
8     cin >> age;
9
10    if (age >= 18) {
11        cout << "Vous etes majeur !\n";
12    }
13    else if (age > 11) {
14        cout << "Vous etes ado !\n";
15    }
16    else {
17        cout << "Tu es un enfant !\n";
18    }
19
20    return 0;
21 }
```



Plusieurs conditions à la fois



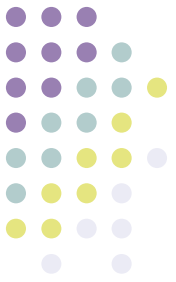
- ET → &&
OU → ||
NON → !
- Si l'âge est supérieur à 18 ET et en même temps inférieur à 25

```
if (age > 18 && age < 25)
```

```
if (moyenne < 10 || une_note < 7.0)
```

```
if (!(age < 18)) ⇔ if (age >= 18)
```

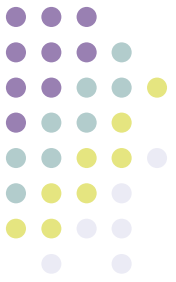
Quelques erreurs courantes



```
1 if (age = 18) {  
2     cout << "Vous venez de devenir majeur !";  
3 }
```

```
1 if (age == 18); // Notez le point-virgule qui ne devrait PAS être là  
2 {  
3     cout << "Vous venez de devenir majeur !";  
4 }
```

Les booléens, le cœur des conditions



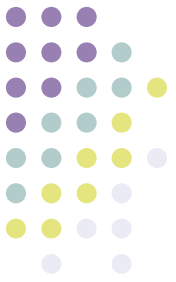
```
1 if (true) {  
2     cout << "C'est vrai" << endl;  
3 }  
4 else {  
5     cout << "C'est faux" << endl;  
6 }
```

C'est vrai

```
1 if (false) {  
2     cout << "C'est vrai" << endl;  
3 }  
4 else {  
5     cout << "C'est faux" << endl;  
6 }
```

C'est faux

Les booléens, le cœur des conditions



- 0 c'est faux, différent de zéro c'est vrai

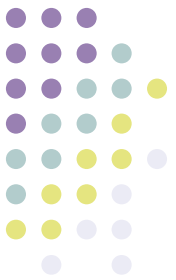
```
1 int age = 20;  
2 int majeur = 0;  
3 majeur = (age >= 18);  
4 cout << "Majeur =" << majeur << endl;
```

`if (toto && 1)` ➔ dépend de toto

`if (toto || 1)` ➔ toujours vrai

`if (toto && 0)` ➔ toujours faux

Sans switch



```
1 if (age == 2) {
2     cout << "Salut bebe !" << endl;
3 }
4 else if (age == 6) {
5     cout << "Salut gamin !" << endl;
6 }
7 else if (age == 12) {
8     cout << "Salut jeune !" << endl;
9 }
10 else if (age == 16) {
11     cout << "Salut ado !" << endl;
12 }
13 else if (age == 18) {
14     cout << "Salut adulte !" << endl;
15 }
16 else if (age == 68) {
17     cout << "Salut papy !" << endl;
18 }
19 else {
20     cout << "Je n'ai aucune phrase de prete pour ton age " << endl;
21 }
```

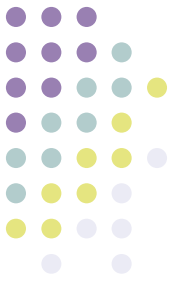
Avec switch



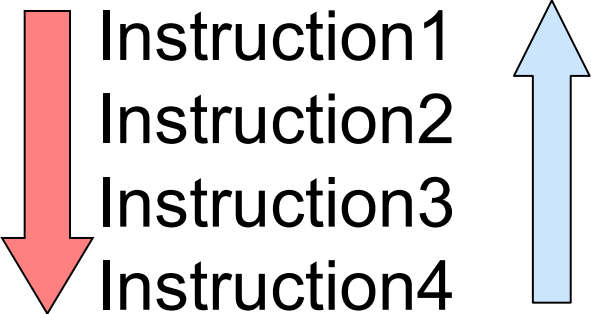
```
1 switch (age)
2 {
3     case 2 :
4         cout << "Salut bebe ! " << endl;
5         break;
6     case 6 :
7         cout << "Salut gamin ! " << endl;
8         break;
9     case 12 :
10        cout << "Salut jeune ! " << endl;
11        break;
12    case 16 :
13        cout << "Salut ado ! " << endl;
14        break;
15    case 18 :
16        cout << "Salut adulte ! " << endl;
17        break;
18    case 68 :
19        cout << "Salut papy ! " << endl;
20        break;
21    default :
22        cout << "Je n'ai aucune phrase de prete pour ton age" << endl;
23        break;
24 }
```

```
1 # include <iostream>
2 # include <stdlib.h>
3 using namespace std;
4 int main(int argc, char *argv[])
5 {
6     int choixMenu;
7     cout << "=== Menu ===\n\n";
8     cout << "1. Royal Cheese\n";
9     cout << "2. Mc Deluxe\n";
10    cout << "3. Mc Bacon\n";
11    cout << "4. Big Mac\n";
12    cout << "\nVotre choix ? ";
13    cin >> choixMenu;
14    switch (choixMenu)
15    {
16        case 1 :
17            cout << "Vous avez choisi le Royal Cheese. ! " << endl;
18            break;
19        case 2 :
20            cout << "Vous avez choisi le Mc Deluxe !" << endl;
21            break;
22        case 3 :
23            cout << "Vous avez choisi le Mc Bacon !" << endl;
24            break;
25        case 4 :
26            cout << "Vous avez choisi le Big Mac !" << endl;
27            break;
28        default :
29            cout << "Vous n'avez pas rentre un nombre correct ! " << endl;
30            break;
31    }
32    return 0;
33 }
```

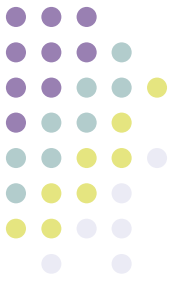




Les boucles

-  Instruction1
Instruction2
Instruction3
Instruction4
- While, do .. While, for
- L'ordinateur lit les instructions de haut en bas
- Puis, une fois arrivé à la fin de la boucle, il repart à la première instruction
- Il recommence alors à lire les instructions de haut en bas...
- ... Et il repart au début de la boucle.

while



```
1 while (/* Condition */)
2 {
3     // Instructions à répéter
4 }
```

- While = tant que

```
1 int nombreEntre = 0;
2 while (nombreEntre != 47)
3 {
4     cout << "Tapez le nombre ! ";
5     cin >> nombreEntre;
6 }
7 cout << "Je sors de la boucle"
```

Tapez un nombre ! 2

Tapez un nombre ! 10

Tapez un nombre ! 47

Je sors de la boucle

while

```
1 # include <iostream>
2 # include <stdlib.h>
3 using namespace std;
4 main(int argc, char **argv)
5 {
6     int i = 0;
7     int som = 0;
8     while (i < 10)
9     {
10         som = som + i;
11         i++;
12         cout << "La valeur de i : " << i << endl;
13     }
14     cout << "La valeur de i à la sortie : " << i << endl;
15     cout << "la somme des dix premiers: " << som << endl;
16     return 0;
17 }
```

La valeur de i : 1

La valeur de i : 2

.....

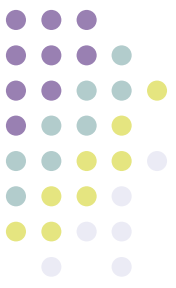
.....

La valeur de i : 9

La valeur de i à la sortie : 10

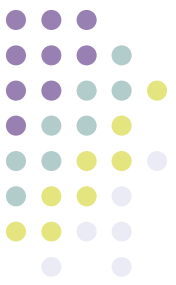
La somme des dix premiers: 45

do ... while



```
1 int compteur = 0;
2 do
3 {
4     cout << "Salut les Zeros !\n";
5     compteur++;
6 }
7 while (compteur < 10);
```

- La même chose que while, sauf que la condition d'arrêt est testée à la fin de chaque itération
- Les instructions sont exécutées au moins une fois



La boucle for

```
1 int compteur = 0;
2 while (compteur < 10)
3 {
4     cout << "Salut les Zeros !\n";
5     compteur++;
6 }
```

```
1 int compteur;
2 for (compteur = 0 ; compteur < 10 ; compteur++)
3 {
4     cout << "Salut les Zeros !\n";
5 }
```