GÉNÉRALITÉS

.....0100111010100.....

LES BASES

DÉFINITION 1:

L'algorithme est une suite d'instructions élémentaires, qui une fois exécutée correctement, conduit à un résultat donné.

- Il vient du mathématicien et astronome perse Muhammad ibn al-Khawarizmi, le père de l'algèbre, qui formalisa au 9e siècle la notion d'algorithme ;
- L'algorithme le plus célèbre est l'algorithme d'Euclide (permettant de calculer le PGCD de deux nombres dont on ne connait pas la factorisation).

LES BASES

Les ***instructions*** et les ***données*** sont codées sous forme de nombres binaires qu'on appelle des ***mots***.

- Un ordinateur ne manipule que deux valeurs : 0 ou 1. En effet, nos ordinateurs sont constitués de circuits intégrés qui sont composés de nombreuses pistes dans lesquelles passe un courant électrique. Or, dans ces circuits il n'y a que deux possibilités : soit le courant passe et dans ce cas cela équivaut à une valeur de un (1), soit le courant ne passe pas, et dans ce cas c'est la valeur zéro (0) qui est retenue. C'est du binaire. Une unité binaire s'appelle un bit (binary digit), un mot inventé par **Claude Shannon** en 1948.
- Nos ordinateurs savent alors communiquer en ***langage binaire.***

LES BASES

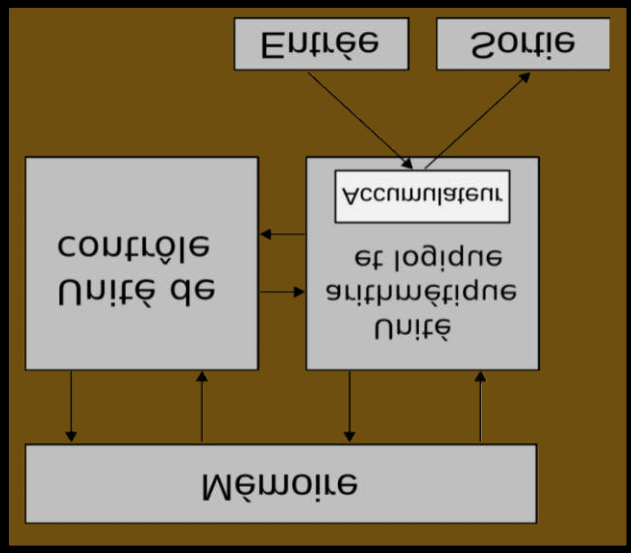
Les bits ne sont pas stockés individuellement dans une case mémoire. Ils sont regroupés, généralement par multiples de huit (8) c'est-à-dire en octet (qui représente les valeurs de 0 à 255).

- Avec l'augmentation des espaces de stockages, de la quantité de mémoire, du besoin de représentation de nombres de plus en plus grands, d'un accès plus rapide à la mémoire ou encore de plus d'instructions, il a fallu augmenter la taille des valeurs à manipuler. De 8, puis 16, puis 32, certains microprocesseurs peuvent manipuler des valeurs de 64 voire 128 bits, parfois plus... Ces valeurs deviennent difficiles à décrire et à représenter. Pour ces valeurs, on parle de mot mémoire.

LES BASES

Un ordinateur est un système de traitement de l'information programmable qui fonctionne par la lecture séquentielle d'un ensemble d'instructions, organisées en programmes, qui lui font exécuter des opérations arithmétiques et logiques.

Les ordinateurs actuels sont tous basés sur des versions améliorées de l'architecture de **Von Neumann** (1944).



LES BASES

L'architecture de Von Neumann est composée de 4 parties distinctes.

- l'unité arithmétique et logique (UAL ou ALU en anglais) ou unité de traitement qui a pour rôle d'effectuer les opérations de base. Certaines documentations lui rajoute des registres (quelques cases mémoires intégrés) et lui confère le nom de de processeur (CPU) ;
- l'unité de contrôle ou de commande (control unit) qui est chargée du séquençage des opérations ou le déroulement du programme. Elle récupère les instructions en mémoire et donne des ordres à l'ALU ;
- la mémoire (une suite de petites cases numérotées appelées registre) contient à la fois les données et le programme indiquant à l'unité de contrôle les calculs à faire. Pour pouvoir accéder à la mémoire, il suffit de connaitre son adresse;
- les dispositifs d'entrée-sortie permettent de communiquer avec le monde extérieur. Il peut s'agir d'un clavier pour entrer les données et d'un écran pour visualiser les résultats.

LES BASES

DÉFINITION 2:

La programmation peut être vue comme l'art de déterminer un algorithme (une démarche) pour résoudre un problème et d'exprimer cet algorithme au moyen d'un langage de programmation (exemple C, C++, PYTHON, PHP, JAVA, etc.).

- Les langages de programmation permettent alors de parler à l'ordinateur plus simplement qu'en binaire par l'intermédiaire d'un compilateur qui se charge de traduire les instructions en binaire.
- La programmation est donc une activité fondamentale en informatique.

LES BASES

L'efficacité d'un algorithme est fondamentale pour résoudre effectivement des problèmes. L'efficacité d'un algorithme est mesurée par son coût (complexité) en temps et en mémoire.

- La complexité d'un algorithme est en temps, le nombre d'opérations élémentaires effectuées pour traiter une donnée de taille n, en mémoire, l'espace mémoire nécessaire pour traiter une donnée de taille n.
  - **Exemple**: lancé un dé est un algorithme très simple, court, concis et rapide. Ce n'est pas toujours le cas pour d'autres algorithmes. Certains pourront nécessiter beaucoup de temps et de ressources.

LES BASES

LE LANGAGE C++

Le langage C++ est une évolution du langage C. disons simplement que le C est la plupart du temps du « C++ » sans la partie objet. Il est donc un langage de bas niveau.

Il est multi-paradigmes. Ce qui veut dire qu'il respecte à la fois l'impératif et l'objet.

- **Pour la petite histoire**:
  - Le langage C++ a été développé vers les années 80 par ***Bjarne Stroustrup*** afin d'améliorer le C. il fut nommé pour la première fois, '***C with classes***' et a été normalisé pour la première fois en 1998.
  - Depuis, il a beaucoup évolué. nous constatons de nombreuses correction et ajout de fonctionnalité.

LE FORMALISME

LE FORMALISME

Prenons l'exemple du jeu dé pour mieux représenter les différentes étapes.

- 1ère étape : lancer le dé;
- 2ème étape : saisir une valeur;
- 3ème étape : si la valeur saisie est différente de la valeur du dé, retourner à la *première étape*, sinon continuer;
- 4ème étape : afficher "bravo".

LE FORMALISME

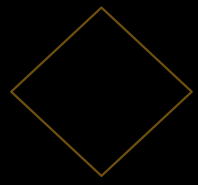
LA REPRÉSENTATION GRAPHIQUE

Un **organigramme** est constitué de symboles dont les formes sont normalisées. Ces symboles sont reliés entre eux par des lignes fléchées qui indiquent le chemin. Ainsi, nous avons:

Représente un début ou une fin

Utilisé pour la lecture ou l'écriture

Utilisé pour les calculs

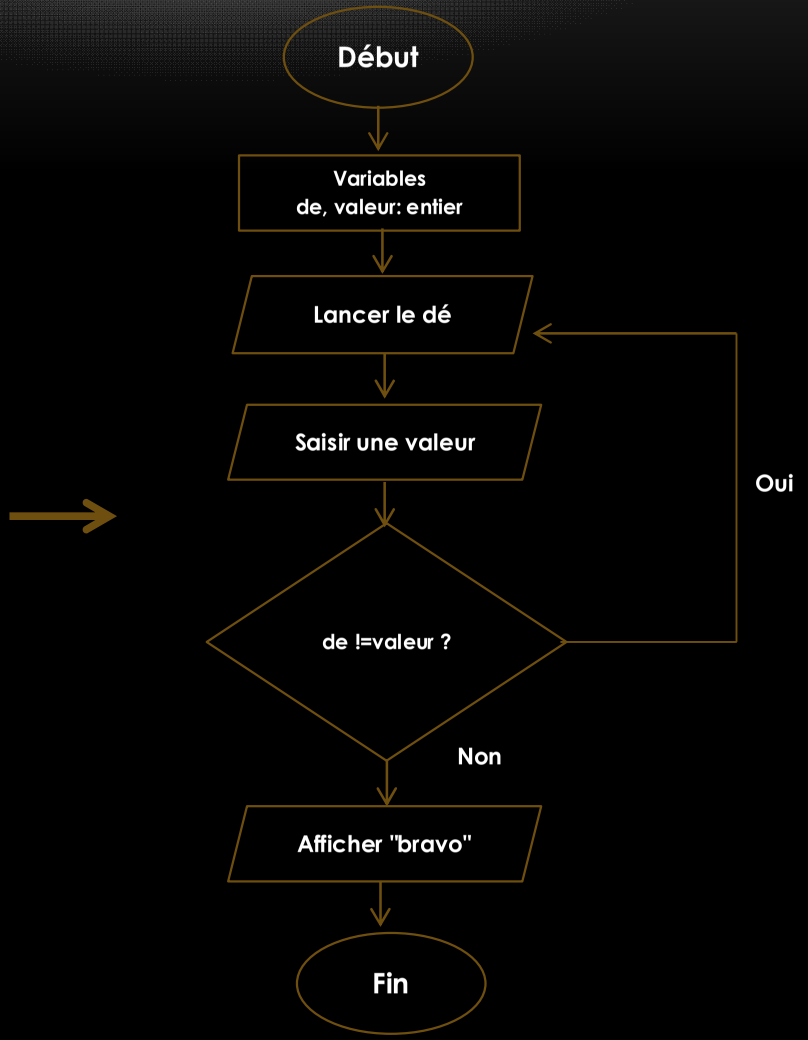


Représente les Tests

LE FORMALISME

L'ORGANIGRAMME

La représentation graphique du jeu de dé.



LE FORMALISME

L'ALGORITHME SOUS FORME DE TEXTE

Ecrire un algorithme sous forme de texte est sans doute la manière la plus simple de faire comprendre à un non informaticien ce que votre code est censé faire.

La syntaxe utilisée est simple, précise et concise.

Reprenons notre exemple sur le jeu de dé.

```
/* Commentaires : jeu de dé */
/* Nom du programme  */
ALGORITHME jeu_de_de
/* Déclarations des variables, constantes, types, etc */
VARIABLES
        de:entier;
        valeur:entier;
/* Déclarations de fonctions*/
/* Début du programme */
DEBUT
        de←aléatoire(6);
        valeur←0;
        Tantque valeur<>de Faire
                Ecrire ("saisir une valeur ");
                Lire (valeur);
        FinTantQue
        Ecrire ("Bravo");
FIN
```

LE FORMALISME

L'ALGORITHME SOUS FORME DE TEXTE

Ce pseudocode algorithmique, est décomposé en plusieurs parties.

- Le nom de l'algorithme situé après le mot "ALGORITHME",
- Une zone de déclaration des données utilisées par le programme. Cette zone commence par le mot "VARIABLES",
- Les instructions du programme sont encadrées par les mots "DEBUT" et "FIN",
- Chaque instructions est terminée par un point-virgule " ; ",
- Les commentaires peuvent être encadrés par séquences de caractères "/*" et "*/" s'il s'agit de plusieurs lignes ou par "//" s'il s'agit d'une seule ligne.
- L'affectation permet de donner une valeur à une variable : valeur <- 0 « reçoit ». Si valeur avait une valeur auparavant, cette valeur disparaît. Généralement, le formalisme est la suivante :

  <id_variable> <- <expression>,

- Les opérations entrées/sorties permettent de récupérer une valeur venant de l'extérieur (Lire) ou de transmettre une valeur à l'extérieur (Ecrire).

LE FORMALISME

LE PROGRAMME EN C

Les lignes en haut qui commence par # sont appelées directives de préprocesseur (un programme qui se lance au début de la compilation).

Elles ajoutent avec le mot clé include des fichiers qui existe déjà pour la compilation. Ces fichiers sont appelés des bibliothèques, librairies en anglais).

Exemple, stdio.h (pour standard input output) est une bibliothèque qui vous permet entre autres d'interagir avec l'utilisateur.

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

int main(){
        // déclaration
        int de, valeur ;
        srand( time( NULL ) );
        de = rand() % 6 + 1 ;
        valeur = 0 ;
        while(valeur!=de){

                printf("veuillez saisir une
                valeur \n");
                scanf("%d", &valeur);
        }
        printf("Bravo !");

        return 0;

}
```

LE FORMALISME

LE PROGRAMME EN C

- **main** est la fonction principale. Le programme, commence toujours avec la fonction main. Elle débute avec l'accolade ouvrante « { » et se termine avec l'accolade fermante « } ». Chaque ligne à l'intérieur de main est appelée instruction.
- printf affiche un message à l'écran. Exemple : printf("veuillez saisir une valeur ").
- antislash (\) puis une seconde lettre indique qu'on veut aller à la ligne (\n) ou faire une tabulation (\t), etc…
- scanf récupère ce que l'utilisateur entre dans la console sous demande de la fonction printf. Il prend en entrée, le format de la donnée (int⇔%d, float⇔%f, etc.) et le nom de la variable en question précédé du symbole « & » .
- return 0 indique qu'on est arrivé à la fin de la fonction main en renvoyant la valeur 0. Cela n'empêchera pas le programme de fonctionner mais c'est plus sérieux de le mettre.

LE FORMALISME

LE PROGRAMME EN C++

- Comme dans C, nous avons les directives de préprocesseur. Ici, la standard est iostream « Input Output Stream », ce qui veut dire « Flux d'entrée-sortie » et donc est diffèrent de stdio de C.
- using namespace est un espace de noms. Son rôle est d'éviter les problèmes d'appel de fonction de même noms se situant dans deux bibliothèque diffèrent. std correspond à la bibliothèque standard, livrée par défaut avec le langage C++ et dont iotream fait partie.
- cout (comme printf en C) permet d'afficher un message à l'écran. Les morceaux de texte sont séparés par les chevrons ouvrants (<<) et le mot clé endl est utilisé pour réaliser des retours à la ligne.
- cin (comme scanf en C) permet de faire entrer des informations dans le programme. Les chevrons fermants (>>) sont utilisés pour cela.

```
# include <iostream>
#include <cstdlib>
#include <time.h>

using namespace std ;
int main(){
                // déclaration
                int de, valeur ;
                srand( time( NULL ) );
                de = rand() % 6 + 1 ;
                valeur = 0 ;
                while(valeur!=de){
                                cout << "veuillez sairir une
                                valeur " << endl;
                                cin >> valeur ;
                }
                cout << "Bravo !" << endl;

                return 0;
}
```