Introduction au langage C – Labo –

## Chapitre 1 : Découvertes

* Langage de moins en moins utilisé. Cependant, il possède de nombreux dérivés comme le C++
* Aucune notion d’objet, d’héritage ou de polymorphisme
* Toujours inclure :



* Autres librairies :

Une image contenant texte

Description générée automatiquement

## Chapitre 2 : Notions fondamentales

* Déclarer ses fonctions en début de programme (composé du type de retour, du nom de la fonction et du type des paramètres) appelé prototype ou signature
* Main = programme principal
* Scanf 🡪 utiliser & devant les variables élémentaires (ce symbole permet d’envoyer l’adresse de la variable)
* Terminer le main avec :



### 2.1 Les types élémentaires en C

Une image contenant texte

Description générée automatiquement

Une image contenant texte

Description générée automatiquementUne image contenant texte

Description générée automatiquementUne image contenant texte

Description générée automatiquement

* Pas de type booléen. **0** est toujours **FAUX**, toute autre valeur **VRAI**.

### 2.2 Printf et scanf

#### 2.2.1 Printf

Une image contenant texte

Description générée automatiquement

Une image contenant texte

Description générée automatiquement

#### 2.2.2 Scanf

* Toujours finir le format scanf par « **%\*c** »
* Le symbole « **&** » doit être utilisé pour toutes les variables élémentaires

### Autres informations :

* Mettre les prototypes/signatures dans des fichiers séparés (fichier en **.h**)
* **Un seul dossier par projet !**

## Chapitre 3 : Les chaines de caractère

* Représenté sous la forme de tableaux de caractère
* Requière l’inclusion de la bibliothèque standard :



* Toujours avoir le caractère ‘\0’ à la fin de la chaine de caractère
* Il faut toujours faire une précaution visant à empêcher l’utilisateur de dépasser le nombre de caractère prévu dans sa chaine

### 3.1 Fonctions classiques

Une image contenant table

Description générée automatiquement

Une image contenant texte

Description générée automatiquement

* Pour modifier un caractère, c’est simple, on fait comme dans un tableau classique Java

## Chapitre 4 : Les structures

* Il y a deux façons de déclarer une structure :

Une image contenant texte

Description générée automatiquement Une image contenant texte

Description générée automatiquement

* Après avoir déclarer une structure, (ici étudiant) devient un nouveau type et donc on est capable de faire :  donc il ne faut plus utiliser ‘struct’ devant après la deuxième méthode
* Pour accéder à une variable en particulier, il suffit de faire :  ce qui équivaut à nomDeLaVariable.nomDeLaDonnee
* Les structures sont déclarées en dehors de tout bloc
* Il est possible de partager ses structures entre ses fichiers de cette façon :  Une image contenant texte

  Description générée automatiquement
* Il est également possible de de copier tous les données d’une structure vers une autre : 
* Pour modifier une donnée dans une structure il faut faire : 

## Chapitre 5 : Les tableaux

* Exemples de création d’un tableau :

Une image contenant texte, orange

Description générée automatiquement

* Dimensions d’un tableau sont statiques donc on définit une constante en début de programme pour respecter le principe DRY (Do Not Repeat Yourself) :



* Pour parcourir un tableau 🡪 Boucle for (Attention, il faut bien initialiser le i avant la boucle et non dans la boucle comme en java)
* En C, le. length n’existe pas, il est donc conseiller de créer une variable qui va mémoriser le remplissage actuel du tableau
* Il est possible d’initialiser un tableau (lui transmettre des valeurs) dès sa création
* Une fonction C ne peut pas retourner un tableau comme résultat mais peut très bien prendre un tableau en paramètre (qui sera pris sous forme de pointeur) 🡪 Les modifications apportées au tableau dans la fonction sont appliquées même si la fonction ne renvoie pas de tableau

## Chapitre 6 : Les pointeurs

* Variable qui peut contenir l’adresse mémoire d’une autre variable
* Voici comme ça marche :

|  |  |
| --- | --- |
| int\* i ; | int j ; |
| i adresse | &j adresse |
| \*i int | j int |

* En utilisant \* devant un pointeur, on retrouve sa variable (inverse de &) 🡪 déréférencer un pointeur
* Raccourci d’écriture, lors de l’utilisation d’une structure avec un pointeur, « var🡪 » est équivalent à «(\*var) .»
* Il existe un pointeur non typé « void\* ». Tout pointeur peut être converti vers un type void\*. Un pointeur void\* peut être converti via un casting vers n’importe quel type.