ISEN ALL IS DIGITAL! OUEST | yncréa

# TP n°1 - Algorithmique
## *Utilisation des entrées-sorties*

benoit.lardeux@isen-ouest.yncrea.fr

## Objectifs

- Compilation d'un programme en langage C / Utilisation du débuggeur
- Utilisation des entrées / sorties en langage C
- Création de fonctions
- Compilation séparée / Réalisation d'un makefile

## 1 Programme initial

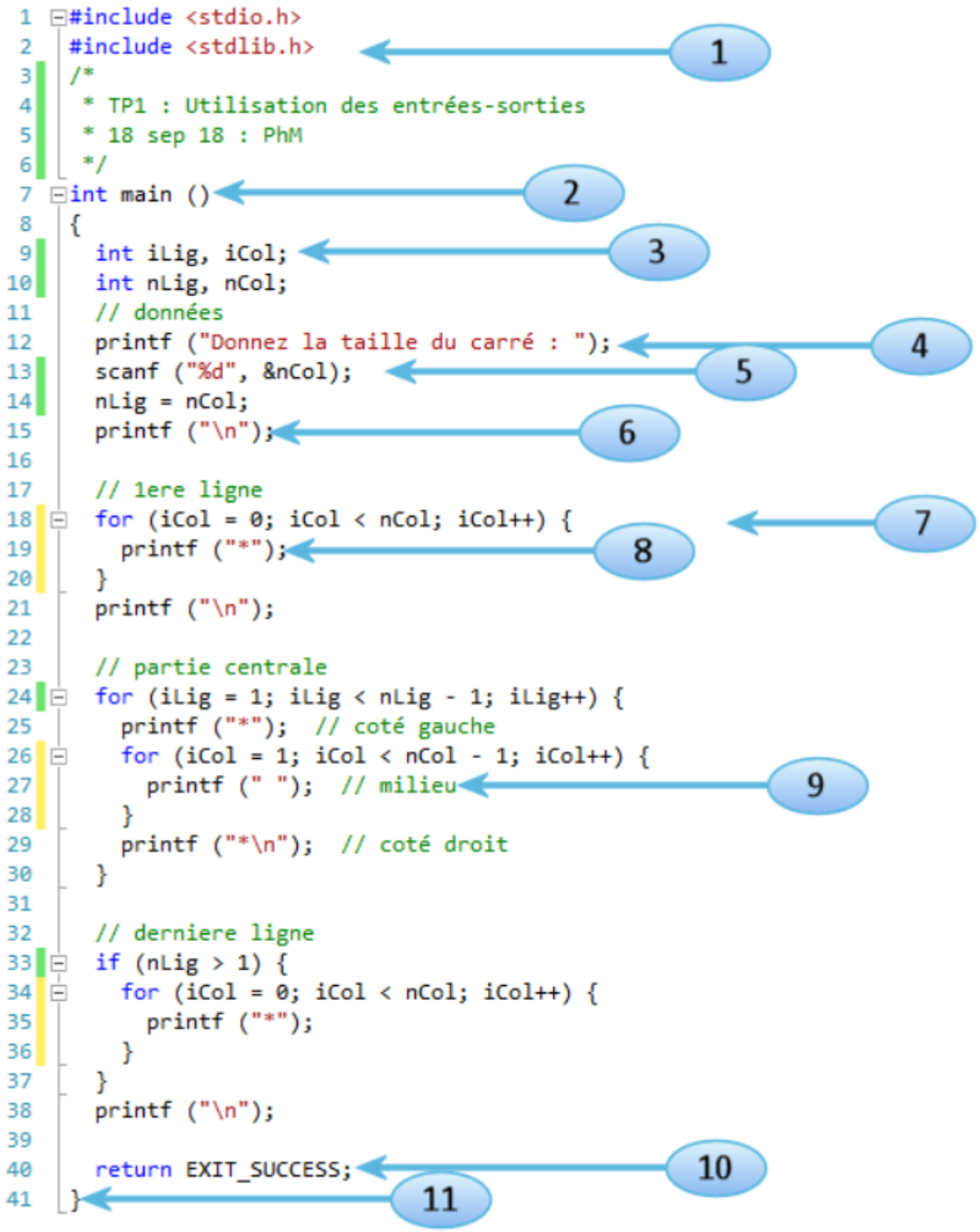
Dans la WSL, créer un répertoire *tp1* dans votre répertoire de travail *algo*. Vous recopierez ensuite le code décrit ci-dessous en Figure 1 (vous pouvez également télécharger sur l'ENT ce même programme) puis vous le nommerez *exo1.c* en prenant soin de la placer dans le répertoire *tp1*.

Quelques rappels :

- 1. Inclusion du fichier système *stdio.h* et *stdlib.h*. Le premier fichier décrit l'interface des fonctions *printf* et *scanf*, ces fonctions peuvent alors être appelées dans le programme que l'on écrit. Le deuxième fichier décrit (entre autres) la macro *EXIT_SUCCESS* utilisée en fin de programme

- 2. Fonction principale, *main* est la première fonction appelée lors de l'exécution. Ici, elle n'a pas de paramètres

- 3. Déclarations de variables entières
- 4. Affichage de texte à l'écran. Le texte est compris entre 2 guillemets
- 5. Lecture d'un entier à partir de l'entrée standard. On précise le format de lecture %d => entier, et on indique l'emplacement où ranger l'entier lu : *&nCol* , adresse de l'emplacement associé à l'entier *nCol* précédemment déclaré
- 6. *\n* est un caractère spécial : le saut de ligne. L'instruction *printf ("\n") ;* permet donc de passer à la ligne suivante dans l'affichage à l'écran
- 7. Pour *iCol* $\leftarrow$ *1* à *nCol* faire ===> pour (*iCol* $\leftarrow$ *1 ; tant que iCol* $\leq$ *nCol ; iCol* $\leftarrow$ *iCol +1 )*
- 8. Affichage du caractère "étoile". On aurait pu aussi écrire *printf ("%c",'*') ;*
- 9. Affichage du caractère "espace" ou "blanc"
- 10. Le return *(EXIT_SUCCESS) ;* est là pour signifier au système d'exploitation que le logiciel a été exécuté sans erreur. *EXIT_SUCCESS* est une constante qui vaut 0, elle est définie dans *stdlib.h*. On peut donc, si une erreur se produit, retourner au système une autre valeur que 0 par *return x* ; où x est une variable qui contiendrait une valeur entière autre que 0. On peut ainsi associer une certaine valeur à un certain type d'erreur. Il est conseillé de n'avoir qu'un seul point de sortie par fonction. Notez que la syntaxe *return x ;* est également correcte. Lorsque la fonction *main ()* aura terminé d'exécuter ses instructions, le programme se terminera




```c
#include <stdio.h>
#include <stdlib.h>
/*
 * TP1 : Utilisation des entrées-sorties
 * 18 sep 18 : PhM
 */
int main ()
{
  int iLig, iCol;
  int nLig, nCol;
  // données
  printf ("Donnez la taille du carré : ");
  scanf ("%d", &nCol);
  nLig = nCol;
  printf ("\n");

  // 1ere ligne
  for (iCol = 0; iCol < nCol; iCol++) {
    printf ("*");
  }
  printf ("\n");

  // partie centrale
  for (iLig = 1; iLig < nLig - 1; iLig++) {
    printf ("*");  // coté gauche
    for (iCol = 1; iCol < nCol - 1; iCol++) {
      printf (" ");  // milieu
    }
    printf ("*\n");  // coté droit
  }

  // derniere ligne
  if (nLig > 1) {
    for (iCol = 0; iCol < nCol; iCol++) {
      printf ("*");
    }
  }
  printf ("\n");

  return EXIT_SUCCESS;
}
```

- 11. Toute fonction commence par une accolade ouvrante, et finit par une accolade fermante

On vous demande de compiler puis d'exécuter le programme ci-dessus et d'interpréter le résultat. Vous pouvez utiliser le débuggeur pour observer ligne à ligne l'évolution du programme

(voir annexe).

## 2 Rectangle paramétrable

Dans cette partie, on vous demande de transformer le programme précédent pour dessiner un rectangle de L x H étoiles ou L et H seront demandés à l'utilisateur du programme au moyen de *scanf*.

Sous l'éditeur de texte, utilisez "File" "Save" (ou "Save As" pour attribuer un nouveau nom de fichier) et découpez le programme précédent en deux fichiers :

- *rect.c* contenant une fonction permettant de tracer le rectangle L*H
- *main.c* contenant le programme principal qui demande L et H, puis appelle la fonction présente dans *rect.c*

Créez un troisième fichier *rect.h* spécifiant les interfaces de rect.c, c'est à dire les éléments utilisables qui sont décrits en détail dans le fichier .c.

Ici, *rect.h* contiendra simplement : void rectangle ( int L, int H ) ; qui indique que l'on peut utiliser la procédure rectangle avec deux entiers.

Ajoutez *#include "rect.h"* à la suite de *#include <stdlib.h>* dans *main.c*

Ceci permet au compilateur de "connaître" la procédure rectangle, et de pouvoir compiler *main.c*. (Rappel : <fichier_système.h> "fichier_utilisateur.h" )

Compilez chacun des fichiers *.c* avec l'option *–c* pour la compilation séparée :
*gcc –c main.c*
*gcc –c rect.c*

Compilez l'ensemble pour produire un exécutable :
*gcc main.o rect.o –o exo1b.x*
(on peut aussi faire *gcc main.c rect.o –o exo1b.x* , mais on recompile alors *main.c*)

On vous demande d'écrire un *makefile* permettant d'exécuter automatiquement les commandes précédentes.

## 3 Croix de Saint-André

La croix de saint André est une croix en forme de X. Ce symbole a été utilisé par de nombreux pays européens (France, Espagne, Belgique, Russie, Écosse, Pays-Bas et Irlande).

On vous demande de copier *rect.c* dans un nouveau fichier que vous nommerez *rect2.c* et de modifier le programme pour afficher une croix de Saint André à l'intérieur du rectangle.

Exemples :

Attention, selon que l'on compile main.c avec l'un ou l'autre des fichiers rectangle, on obtient un exécutable différent.

```
Donnez la largeur : 12
Donnez la hauteur : 10

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

```
Donnez la largeur : 9
Donnez la hauteur : 11

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

## 4 Bornes maximum

En partant de l'exercice précédent, on souhaite borner l'affichage du dessin par deux constantes LMAX et HMAX dont on définira les valeurs dans un fichier d'en-tête par exemple *"limites.h"*. Ces constantes sont définies au moyen de : *#define constante valeur*.

Définir dans *limites.h*, deux constantes *LMAX* et *HMAX* avec des valeurs arbitraires. Inclure ensuite le fichier *limites.h* là où c'est utile par la directive *#include "limites.h"* ajoutée par exemple après l'inclusion de *stdio.h*.

Copier et modifiez le programme, au moyen de boucles tant que, pour que l'affichage du rectangle et de la croix ne s'effectue que sur la zone *LMAX x HMAX*. Exemple : avec *LMAX* et *HMAX* définies dans limites.h avec les valeurs respectives 20 et 5.

```
Donnez la largeur : 22
Donnez la hauteur : 11

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

## 5 Affichage dans un fichier

Reprendre l'exercice précédent et enregistrer vos résultats dans un fichier.
Vous pourrez utiliser la commande man pour obtenir les informations détaillées concernant les fonctions d'accès aux fichiers :
*fopen(), fprintf(), fscanf() et fclose()*

## Annexes

### Compilation et génération d'exécutable

Afin d'obtenir, à partir de code en langage C, un programme que l'on peut réellement exécuter sur un ordinateur, deux étapes principales sont nécessaires :

1- La compilation :

- Le code source écrit en C est d'abord envoyé à travers un préprocesseur qui effectue l'inclusion des fichiers d'en-tête ainsi que le remplacement des différentes macro-instructions déclarées
- Le code obtenu est ensuite effectivement compilé pour donner un code en assembleur
- Le code assembleur est ensuite assemblé afin de générer un fichier objet associé lisible par la machine

2- Édition de lien :

- La totalité des fichiers objets correspondants au programme complet (codes objets provenant des sources écrites par l'utilisateur et codes objets correspondants à des bibliothèques disponibles sur la machine) sont réunis pour générer un exécutable

Sous l'environnement Linux, le compilateur libre gcc (GNU Compiler Collection) permet d'effectuer l'ensemble de ces opérations en une ou plusieurs étapes au moyen de la commande *gcc*


| Ligne de commande | Opérations effectuées |
|---|---|
| `gcc fich.c` | Compile le fichier `fich.c` et génère un exécutable nommé par défaut a.out |
| `gcc fich.c -o fich.x` | Compile le fichier `fich.c` et génère un exécutable nommé `fich.x` |
| `gcc -c fich.c` | Compile le fichier `fich.c` et produit un code objet nommé `fich.o` |
| `gcc fich.o` | Edition de liens du fichier objet `fich.o` et génération d'un exécutable nommé par défaut a.out |

De nombreuses autres options existent, tapez *man gcc* pour davantage d'informations

## Exécution du programme

Une fois la compilation et l'édition de liens réalisées avec succès, vous pouvez lancer votre exécutable directement dans votre terminal en tapant la commande :

*./{nom executable}*