C par l'exemple
Exercices

—=—=—

Exercice 1 *Écrire une fonction qui affiche la taille des types prédéfinis.*

Exercice 2 *C ne précise pas si le type* `char` *est signé ou non. Que pensez-vous d'une boucle telle que:*

```
for(c = 0; c<128;++c) { ... }
```

sur une machine où le type est signé.

Exercice 3 *Écrire un* `main` *qui donne le maximum de 3 nombres.*

Écrire une fonction qui donne le maximum de 3 nombres.

Exercice 4 Questions (expressions):

1. *Quelle est la valeur logique de l'expression* `c = getchar()` *?*
2. ```
   char s[4] = "oui";
   char t[4];
   ```
   Quels sont les types, valeurs et effet de bord de l'expression $t[i] = s[i]$ *pour* i *allant de* 0 *à* 3 *?*

   Quelle est la valeur logique de cette expression ?
3. *Écrire une expression vraie si et seulement si* x *et* y *ont la même parité*
4. *Montrer qu'il existe en C une expression logique permettant de trouver le premier élément d'un tableau dont la valeur soit égale á une valeur donnée, sans sortir du tableau.*
5. *Le morceau de programme:*
   ```
   int A[10],B[10];
   int i =0;
   ...
   while(A[i] || B[i]++) i++;
   ```
   est censé parcourir simultanément 2 tableaux A *et* B *jusqu'à ce que* $A[i] = B[i] = 0$*, en incrémentant* $B[i]$ *de 1 à chaque fois. Est-il correct?*
6. *Même question pour le programme:*
   ```
   int t[100];
   int i=0,j=10;
   while(!t[i++] || !t[--j]);
   ```
   censé parcourir le tableau t *simultanément de la gauche vers la droite et de la droite vers la gauche et s'arrêter lorsque* $t[i]$ *et* $t[j]$ *sont non nuls.*
   ```

Exercice 5 *Écrire une fonction qui transforme minuscules en majuscules et inversement sur l'entrée standard.*

Exercice 6 *Écrire une version simplifiée de la commande* `wc`*:*

On compte le nombre de caractères, de mots, et de lignes sur l'entrée standard.

Exercice 7 *Écrire un programme de résolution d'une équation du deuxième degré.*

Exercice 8 *Écrire une fonction qui échange la valuer de ses arguments*

Exercice 9 *Donner la décomposition binaire d'un entier d'une part en utilisant les opérations de division et modulo, d'autre part en utilisant les opérations bit à bit.*

Exercice 10 *Écrire un ensemble de fonctions utilisant les opérateurs bit a bit :*

1. *valeur du bit p d'un entier x*
2. *lever le bit p de l'entier x*
3. *imprimer la décomposition en base 2 de l'entier x*
4. *valeur du groupe de n bits à la position p dans l'entier x*
5. *transformation d'un tableau de présence en une suite de bits*

Écrire un `main` *permettant de tester ces fonctions.*

Exercice 11 *Écrire la fonction puissance n-ième :*

1. *de façon naïve*
2. *selon l'algorithme logarithmique, avec ou sans opérateurs bit a bit.*

Exercice 12 *Écrire une fonction qui calcule le quotient et le reste dans la division de deux entiers :*

1. *la fonction retourne un des 2 résultats après avoir stocké l'autre à une adresse qui lui a été passée en argument,*
2. *la fonction retourne une structure,*
3. *la fonction retourne un pointeur sur une structure allouée sur la pile d'exécution,*
4. *la fonction retourne un pointeur sur une structure allouée dynamiquement,*
5. *la fonction calcule à une adresse de structure qui lui a été passée en argument.*

Exercice 13 *Écrire une fonction qui donne la somme des éléments positifs d'un tableau d'entiers qui lui est passé en paramètre. Écrire un* `main` *qui alloue dynamiquement un tableau et appelle cette fonction.*

Exercice 14 *Écrire une fonction qui donne le minimum et le maximum d'un tableau*

Exercice 15 *Donner la dimension du plus long plateau d'un tableau (un plateau de i à j du tableau a est tel que pour tout k:*

$$i \leq k \leq j \rightarrow a[i] = a[k]$$

la dimension est alors $j - i + 1$)

Exercice 16 *Trier un tableau par ordre croissant.*

Exercice 17 *Quels sont les résultats du programme suivant? expliquer.*

```
int t1[] = {0, 0, 1, 1, 1}, t2[] = {0, 0, 1, 1, 1};
main() {
   int *p1 = t1, *p2 = t2;
   while ( !*p1++ || !*p2++ );
   printf("%d %d", p1 - t1, p2 - t2);
}
```

Exercice 18 Pointeurs :

Soient les définitions suivantes:

```
char c[10], *s, **t;
int (*f)(), g(int);
struct st {
   char **st_ch;
   int (*st_pf)();
} *stp;
```

Les instructions suivantes sont elles valides?

1. `s = c++;`
2. `*++t = s;`
3. `*t[8] = c[2];`
4. `*t = *++s;`
5. `*t++ = *--s;`
6. `f++ = g;`

7. `f = g;`
8. `*f = g;`
9. `s = (char *)f;`
10. `f = (int (*)())s;`
11. `f = stp->st_pf;`
12. `**stp->st_ch = c[0];`
13. `**(stp->st_ch) = s;`
14. `stp = (struct st *) *t;`
15. `stp = (struct st *)c;`
16. `f = (int (*)())stp->st_ch;`

Exercice 19 Déclarations:
Comment déclarer:

1. *un pointeur sur un tableau de 3 entiers*
2. *un tableau de 5 tableaux de 3 pointeurs sur void*
3. *un pointeur sur un tableau de 5 pointeurs sur entiers*
4. *un tableau de 5 pointeurs sur fonction non prototypée retournant un pointeur sur un tableau de 5 entiers*
5. *un tableau de 4 pointeurs sur fonction ayant comme arguments un pointeur sur int et un int retournant un pointeur sur un tableau de 5 entiers*

On précisera bien la taille des objets alloués par ces déclarations.

Exercice 20 *Inversement à quels objets correspondent les types:*

1. `int *t[5][];`
2. `int (*t)[5][];`
3. `int (*f)(int,int *t[][5]);`
4. `int (*(*f)(int,void *))[5];`
5. `int (*(*)(int,void *))[5];`

Exercice 21 *Écrire différentes fonctions de manipulation de chaînes de caractères.*

1. *concaténation de deux chaînes*
2. *concaténation de deux chaînes limitée aux n premiers caractères.*
3. *comparaison de deux chaînes*
4. *recopie d'une chaîne dans une autre*

5. *recopie d'une chaîne dans une autre limitée aux n premiers caractères.*
6. *longueur d'une chaîne*
7. *Recherche de la première occurence d'un caractère dans une chaîne (on restitue un pointeur sur cette occurence).*
8. *Recherche de la dernière occurence d'un caractère dans une chaîne (on restitue un pointeur sur cette occurence).*

Exercice 22 *Convertir une chaîne de chiffres en son équivalent numérique entier en ignorant les éventuels blancs, tabulations placés en tête de ligne.*

Exercice 23 *Convertir une chaîne de chiffres en son équivalent numérique en virgule flottante et en double précision en ignorant les éventuels blancs, tabulations placés en tête de ligne.*

Exercice 24 *La structure point étant définie de la facon suivante :*

```
struct point {
   char nom[5];
   short x, y;
}
```

écrire une fonction permettant de saisir des points et une fonction permettant de les comparer.

Exercice 25 *Écrire une fonction qui calcule la somme d'une série de terme général u_n, le terme général étant calculé par une fonction. Appliquer cette fonction pour le calcul de la somme des n premiers entiers, des n premiers carrés, pour le calcul de e.*

Exercice 26 *Implémenter les opérations suivantes sur les arbres binaires:*

1. *lecture d'un arbre par ordre préfixe*
2. *écriture d'un arbre par ordre préfixe*
3. *écriture d'un arbre par ordre suffixe*
4. *écriture d'un arbre par ordre infixe*
5. *hauteur de l'arbre*
6. *...*

Exercice 27 *représenter les ensembles et les opérations ensemblistes (création, union, intersection, test vide, appartenance) en utilisant :*

1. *Les tableaux*
2. *Les listes chaînées*
3. *Les arbres binaires*

Exercice 28 *Exécuter le programme suivant:*

```
main() {
   int *p1,*p2,*f1(),*f2();
   p1=f1(4);
   p2=f2(*p1);
   printf("*p1=%d   *p2=%d\n",*p1,*p2);
}

int *f1(m) int m; {
   int n=5;
   n += m;
   return(&n);
}

int *f2(m) int m; {
   int p=8;
   p += m;
   return(&p);
}
```

Que se passe-t-il si les variables locales des fonctions f1 et f2 sont de la classe static?

Exercice 29

1. *Calculer la suite de Fibonnaci donnée par :*

```
FIB(x) = x si x < 2
      FIB(x-1)+FIB(x-2) sinon
```

2. *Calculer la fonction d'Ackermann donnée par :*

```
ACK(x,y) = y + 1 si x=0
        ACK(x-1,1) si x /= 0 et y = 0
        ACK(x-1,ACK(x,y-1)) si x /= 0 et y /= 0
```

3. *Donner le résultat de la fonction suivante :*

```
F(x) = x- 10 si x>100
        F(F(x+11)) sinon
```

Exercice 30 *Le programme suivant est-il correct? que calcule-t-il? Comment ?*

```
int i;
int f(int (*g)()) {
   int j;
   j=i;
   if(j==0) return 1;
   else {
      i--;
      return j*g(g);
   }
}

main() {
   printf("valeur?");
   scanf("%d",&i);
   printf("factoriel = %d\n",f(f));
}
```

Exercice 31 *Écrire une fonction hanoi à un paramètre entier n affichant la suite des déplacements de disques pour résoudre le problème dit des tours de hanoi à n disques .*

Le principe est simple: pour résoudre le problème des tours de Hanoi à n disques (déplacer n disques d'une potence A vers une potence B en utilisant une potence intermédiaire C, un disque quelconque devant toujours être sur un disque plus grand que lui), il suffit de résoudre:

1. *le problème avec les $n-1$ disques du dessus en les transportant de la potence A vers la potence C en utilisant la potence B comme intermédiaire;*
2. *déplacer le disque qui reste sur la potence A vers la potence B;*
3. *déplacer les $n-1$ disques de la potence C vers la potence B en utilisant la potence A comme intermédiaire.*

Exercice 32 *On considère le jeu suivant ("variante du baguenaudier"). On dispose d'une règlette à n cases (numerotées de 1 à n) et sur chacune des cases est posé un pion. On désire poser les pions en respectant les règles suivantes:*

1. *On peut à tout instant poser ou enlever le pion sur la case 1;*
2. *on peut poser ou enlever le pion $j(1 < j <= n)$ s'il y en a un sur la case $j-1$ et si aucune des cases $(1, ..., j-2)$ n'en contient.*

Écrire un ensemble de deux fonctions bag et debag à un paramètre entier m affichant la suite des mouvements de pions permettant respectivement de vider ou remplir les m premières cases de la règlette. Il s'agit là d'un exemple de récursion croisée.

Exercice 33 *Donner le premier élément commun à deux fichiers d'entiers triés.*

Exercice 34 *A partir de deux fichiers contenant des éléments triés, obtenir dans un troisième fichier la réunion triée de ces éléments (fusion de deux fichiers triés).*

Exercice 35 *Écrire une fonction pour insérer n caractères en début de fichier*

Exercice 36 *Écrire une fonction prototypée, qui fait la somme d'un nombre variable d'arguments*

Exercice 37 *Écrire une commande à arguments : comm [ch1 ch2 ... chn]*

- *pour chacune des chaînes en argument, on calule le nombre d'occurrences de chaque caractère y figurant*
- *sans argument : on réalise ce traitement sur une chaîne lue au clavier*

N.B. : envoyer la commande avec un argument permet de traiter une chaîne comportant des espaces ou tabulations (il suffit de la placer entre "), alors qu'un blanc délimite une chaîne lue au clavier

---

Exercice 38 *Écrire une fonction qui saisit au clavier des chaînes de longueur arbitraire et qui mémorise leurs adresses dans un tableau dont la fin est marquée par le pointeur NULL.*

- *La fin de la lecture est detectee par fin de fichier sur l'entree standard.*
- *Le nombre de chaines est limite par NMAX.*
- *La fonction retournera l'adresse du tableau.*

Exercice 39 *Écrire un mini-logiciel de manipulation de polynômes.*

1. *Dans une première version on utilisera le type polynôme défini ainsi :*

   ```
   #define NBCOEF 10
   typedef struct  {
             unsigned char degre;
             short coef[NBCOEF];
          } polynome;
   ```

   Dans une seconde version, on définira un type plus adapté à la représentation des polynômes "creux".

2. *Les opérations suivantes seront réalisées sur le type polynôme :*
    - *lecture sur l'entrée standard*
    - *impression sur la sortie standard*
    - *valeur en un point*
    - *addition*
    - *soustraction*
    - *produit*
    - *dérivée ni‘eme*
3. *Le main lancera l'exécution de ces diverses fonctions au gré des commandes frappées par l'utilisateur.*
4. *On utilisera les commandes make et ar.*

▽△▽△▽△▽