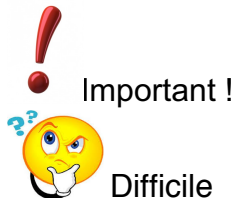


Programmation en C/C++

Série 2

Variables, Affectations Typage de données Entrées/ Sorties



Objectifs

Maîtriser la notion de variables, le typage de données, les entrées/sorties.

Notion de variables, typage des données

Notion de variables

Une **variable** est une zone mémoire qui va permettre de **stocker** des nombres entiers, des réels, caractères, chaînes de caractères...

• Une variable possède un **nom**, un **type** et une adresse mémoire (zone mémoire où l'on stocke l'information). Une variable qui contiendra des nombres entiers possèdera, par exemple, le type entier (*int*) ; une variable qui contiendra des nombres réels possèdera, par exemple, le type réel (*float*)...

Déclaration de variables, affectation

• Une variable doit toujours être **déclarée** avant d'être utilisée ; on doit préciser son **nom** et son **type**. On prendra l'habitude de déclarer les variables en début de programme.

(Exemple : *int mavar* : je déclare une variable nommée "mavar" de type entier).

Types usuels

char : pour stocker le code ascii d'un symbole (8 bits) de -128 à + 127

unsigned char : pour stocker le code ascii d'un symbole de 0 à +255

const : pour empêcher de modifier une variable ex: *const int a = 2;*

short int : pour stocker des entiers petits (16 bits), de -32 768 à 32 767

unsigned short int : pour stocker des entiers petits (16 bits), de 0 à 65 535

int : pour stocker un entier (32 bits), de -2 147 483 648 à 2 147 483 647

unsigned int : pour stocker un entier (32 bits), de 0 à 4 294 967 295

long int : pour stocker des entiers grands (32/64 bits),

float : pour stocker des nombres réels (décimaux) (32 bits),

double : pour stocker des nombres réels plus grands ou avec plus de précision (64 bits),

long double : pour stocker des nombres réels plus grands ou avec plus de précision (80 bits),

Exemple de déclarations de variables

<i>int u;</i>	on déclare une variable nommée <i>u</i> de type entier.
<i>int u, w;</i>	on déclare deux variables nommées <i>u</i> et <i>w</i> de type entier.
<i>float valeur ;</i>	on déclare une variable nommée <i>valeur</i> de type réel.

Affectation

On peut **affecter** des valeurs à une variable (exemple : *a=7* : j'affecte la valeur 7 à la variable *a*).




Attention ! L'affectation se lit de la droite vers la gauche.

L'opérateur égal ne signifie pas "égal" mais "j'affecte".

t = 9; on affecte la valeur 9 à la variable *t*.

Autres exemples d'affectations

- *bal = 3 * (7+8) / 5;* on affecte $3 * (7+8) / 5$ à la variable *bal*
- *float cout = 15.24;* on déclare une variable *cout* de type réel et on lui affecte la valeur réelle 15.24.
- *Compteur = Compteur + 1;* on prend le contenu de la variable *Compteur*, on lui ajoute 1 et on réaffecte le résultat obtenu à la variable *Compteur*.

 **Attention !** En C, le nombre décimal 15,24 s'écrit 15.24 (pas de virgule mais un point avant la partie décimale).

Affectations multiples

- $x = y = 1;$
Cette ligne de code signifie : 1 est affecté à y et le contenu de y est affecté dans x donc à l'arrivée, x vaut 1 et y vaut 1.
- $A = 5 + (B=10);$ on affecte 10 à B et on affecte $5 + 10$ à A. Attention : $(B=10)$ doit être placé entre parenthèses.

Entrées/sorties (I/O)

Affichage à l'écran (SORTIE)

Pour afficher des informations à l'écran on utilisera l'instruction *printf*.

Exemple

```
printf("Bonjour ! \n");
```

Il s'affichera "bonjour" à l'écran.

Opérateur \n

Après l'affichage, le curseur passera au début de la ligne suivante.

Opérateur \t

Après l'affichage, le curseur est décalé d'une tabulation.

Exemples avec \n et \t

```
printf("a \n");
```

Il s'affichera "a" et le curseur ira à la ligne.

```
printf("10 \t 12");
```

10 et 12 seront affichés, séparés d'une tabulation.

Autre exemple

```
int a = 5;
```

```
printf("La variable a vaut : %d ", a);
```

Il s'affichera " La variable a vaut : 5 " En fait, le contenu de la variable a sera affiché.

La variable nommée "a" a le type entier. Pour afficher le contenu d'une variable il faut mettre son format d'affichage dans la partie entre guillemets (%d, ici car c'est une variable de type entier) et mentionner son nom (a, ici) après la virgule.

Exercice avec une variable de type entier

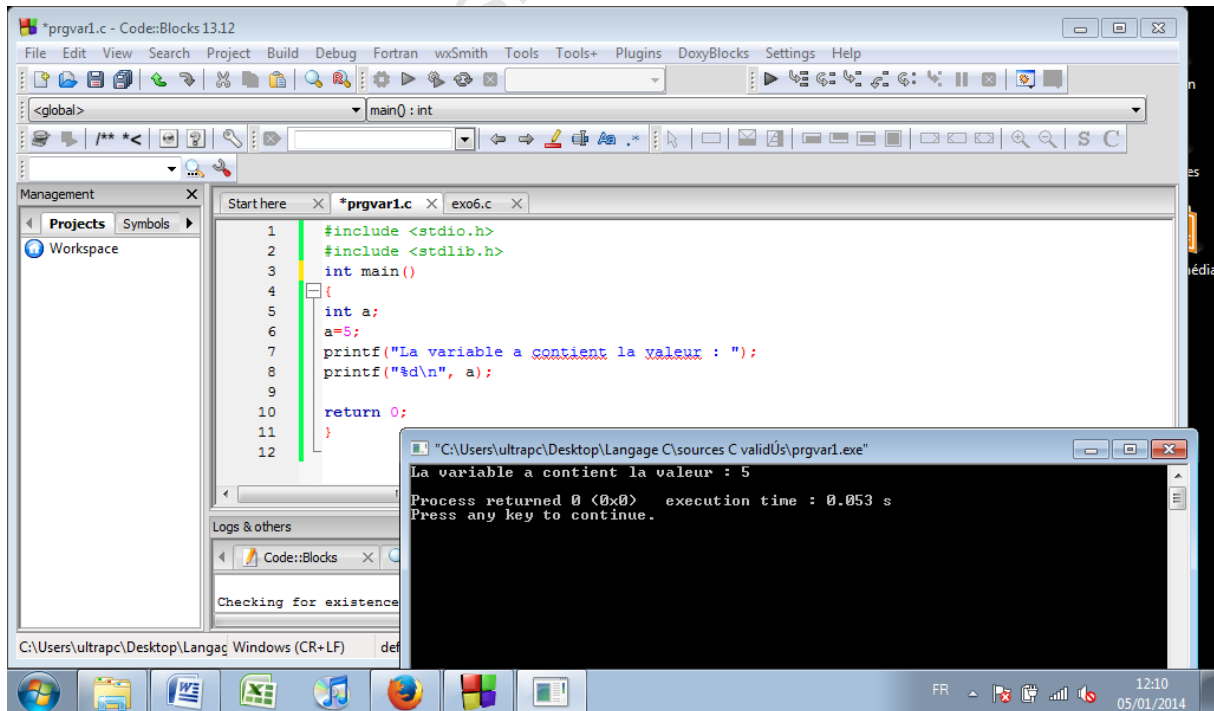
Sans regarder le corrigé, créez le programme C qui affecte la valeur 5 à une variable nommée *VARI* de type entier puis affiche son contenu.

Corrigé

```
#include <stdio.h>
#include <stdlib.h>
int main()
{
int VARI;
VARI=5;
printf("La variable a contient la valeur : ");
printf("%d\n", VARI);

return 0;
}
```

Voici le résultat à l'écran avec une variable nommée "a":



! Quelques formats d'affichage (ou de saisie)

Type	Format	Contenu
short	%hd	Entier court
int	%d	Entier
long	%ld	Entier long
unsigned	%u	Entier non signé
float	%f	Réel en décimal
double	%lf	Réel long en décimal
char	%c	caractère
char VAR[x]	%s	Chaîne de x caractères nommée VAR
pointeur (void*)	%p	adresse
Autres exemples		
float, double	%e	Réel en notation scientifique
int, long	%4d	Entier à 4 chiffres
float, double	%.2f	Réel avec 2 chiffres après la virgule

Exemple

```
float val1 = 10.54321;
```

```
printf("%.2f\n", val1);
```

Rq : Il s'affichera : 10.54

Conversion de type

Conversion implicite

```
int x;
```

```
x = 18.356;
```

x contiendra 18 : x étant déclaré en entier, 18.356 est converti implicitement en entier (18).

Conversion explicite

On peut convertir le type d'une variable (opérateur de cast).

Exemple

```
int V1=8;
```

```
printf("V1 vaut %.2f\n", (float) V1);
```

La variable V1 qui est de type entier au départ est convertie en réel (float) V1;

Il s'affichera à l'écran : " V1 vaut 8.00".

Exercice

Écrivez le programme source complet de l'exemple précédent ; vérifiez le résultat en exécutant le programme.

Saisie de données (ENTRÉE)

L'instruction **`scanf()`** permet de saisir des données au clavier et de les stocker dans des variables. On parle d'entrée car on saisit les données au clavier et elles sont stockées dans une variable. Il faut préciser le type de la variable et on place le symbole & devant le nom de la variable.

Syntaxe : `scanf("type de la variable",&nom_de_variable)`

Exemple : Saisie d'un nombre entier

```
int x;  
printf("Veuillez saisir un entier: ");  
scanf("%d", &x);
```



Ne pas placer d'espace ou de texte de part et d'autre des `%d` !



Pas de `\n` dans les `scanf` !

Rq : on déclare une variable `x` de type entier ; on affiche un message à l'écran pour que l'utilisateur sache ce qu'il doit faire (`printf`); on affecte ce qui est saisi au clavier (`scanf`) dans la variable `x` ; on place `%d` car la variable `x` est de type entier.

Exemple Saisie d'un nombre décimal

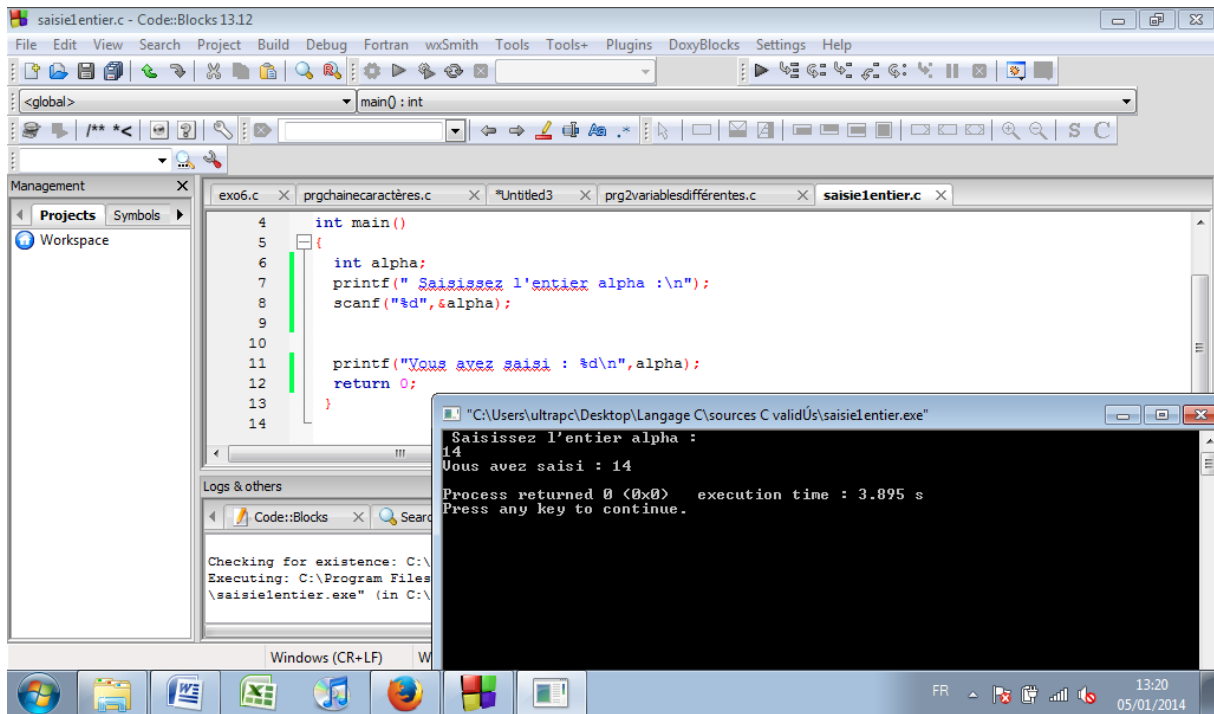
```
float z;  
printf("Veuillez saisir un nombre décimal ");  
scanf("%f", &z);
```

Exercice

Écrivez un programme source C avec une variable nommée *alpha* de type entier. Le programme demande de saisir un entier au clavier, il est stocké dans *alpha* et, à la fin, on affiche "vous avez saisi : " suivi de la valeur de la variable *alpha*.

Corrigé

Voici le résultat à l'écran :



Code source

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

```
int main()
{
    int alpha;
    printf(" Saisissez l'entier alpha :\n");
    scanf("%d",&alpha);
    printf("Vous avez saisi : %d\n",alpha);
    return 0;
}
```

Remarque 1

Lors de la saisie de données au clavier, une suite d'espaces ou tabulations ne compte que pour une espace (espace est un nom féminin en typographie !)

Exemple

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

```
int main()
{
    int A, B, C;
    scanf("%d %d %d", &A, &B, &C);
```

```
printf("Vous avez saisi : %d %d %d \n",A,B,C);
```

```
    return 0;  
}
```

!! Ne pas placer d'espace de part et d'autre des %d !

Dans ce programme, on demande de saisir 3 valeurs au clavier (*scanf*), sans afficher quoi que ce soit.

Les 3 saisies au clavier a), b) ou c) sont équivalentes

Saisie a)

10 100 1000

Saisie b)

10 100 1000

Saisie c)

10

 100

1000



Exemple de saisies successives pour le programme précédent

Remarque 2

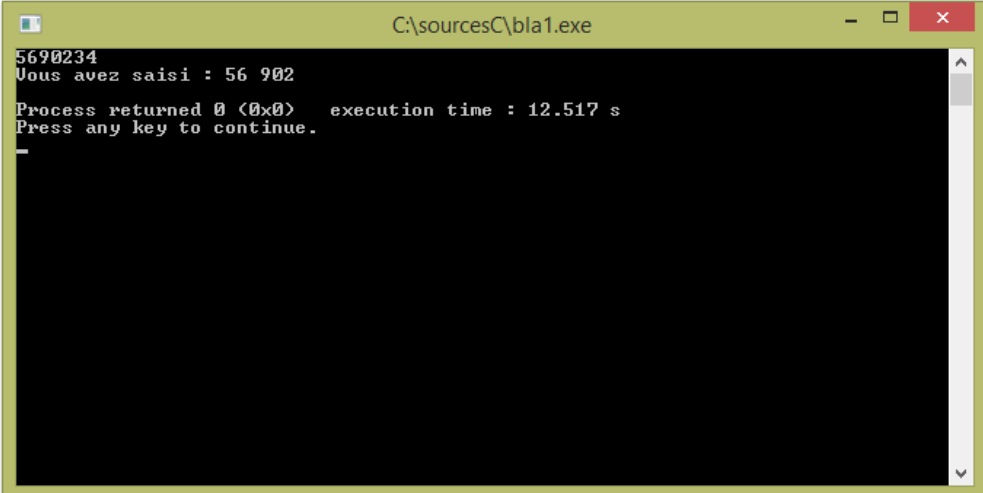
Soient les instructions :

```
int A,T;
```

```
scanf("%2d %3d", &A, &T);
```


Si, par exemple, le nombre 5690234 est saisi au clavier, les 2 variables auront pour valeur : A = 56 et T = 902 ; 34 sera gardé pour le *scanf* suivant !!

```
1  #include <stdio.h>
2  #include <stdlib.h>
3
4  int main()
5  {
6
7      int A,T;
8      scanf("%2d %3d", &A, &T);
9
10     printf("Vous avez saisi : %d %d \n",A,T);
11
12     return 0;
13 }
14
15
16
```



Saisie d'un caractère

On utilisera le type *char* (et le format *%c*).

Exemple

char *symbole* ;

symbole = 78;

// on aurait pu écrire *symbole* = 'N' car N correspond au code ASCII 78;

printf(" Votre symbole est : %c\n", *symbole*);

CODE ASCII

L'American Standard Code for Information Interchange (code américain normalisé pour l'échange d'information), plus connu sous l'acronyme ASCII est une norme de codage de caractères non accentués en informatique.

Table ASCII

0		32		64	@	96	`	128	Ç	160	Á	192	Ł	224	ó
1	☺	33	!	65	A	97	a	129	ü	161	í	193	ł	225	ô
2	☹	34	"	66	B	98	b	130	é	162	ó	194	Ł	226	õ
3	♥	35	#	67	C	99	c	131	â	163	ú	195	ł	227	ö
4	♦	36	\$	68	D	100	d	132	ä	164	ñ	196	Ł	228	õ
5	♣	37	%	69	E	101	e	133	à	165	Ñ	197	ł	229	ö
6	♠	38	&	70	F	102	f	134	å	166	œ	198	Ł	230	µ
		39	'	71	G	103	g	135	ç	167	œ	199	ł	231	ı
		40	<	72	H	104	h	136	ê	168	ç	200	Ł	232	ı
		41	>	73	I	105	i	137	ë	169	@	201	ł	233	ú
		42	*	74	J	106	j	138	è	170	ı	202	Ł	234	û
11	♂	43	+	75	K	107	k	139	ï	171	½	203	ł	235	ü
12	♀	44	,	76	L	108	l	140	î	172	¾	204	Ł	236	ý
13	♂	45	-	77	M	109	m	141	ì	173	ı	205	ł	237	ÿ
14	♂	46	.	78	N	110	n	142	ä	174	<<	206	Ł	238	-
15	*	47	/	79	O	111	o	143	å	175	>>	207	ł	239	'
16	▶	48	0	80	P	112	p	144	É	176	▤	208	Ł	240	-
17	◀	49	1	81	Q	113	q	145	æ	177	▥	209	ł	241	±
18	‡	50	2	82	R	114	r	146	Œ	178	▦	210	Ł	242	=
19	‡	51	3	83	S	115	s	147	ô	179		211	ł	243	¾
20	¶	52	4	84	T	116	t	148	ö	180	ı	212	Ł	244	¶
21	§	53	5	85	U	117	u	149	ò	181	Á	213	ł	245	§
22	—	54	6	86	V	118	v	150	û	182	Â	214	Ł	246	÷
23		55	7	87	W	119	w	151	ù	183	À	215	ł	247	°
24	↑	56	8	88	X	120	x	152	ÿ	184	©	216	Ł	248	°
25	↓	57	9	89	Y	121	y	153	õ	185	ı	217	ł	249	°
26	→	58	:	90	Z	122	z	154	Ü	186		218	Ł	250	°
27	←	59	;	91	[123	<	155	ø	187	ı	219	ł	251	°
28	└	60	<	92	\	124	ı	156	£	188	ı	220	Ł	252	°
29	↕	61	=	93]	125	>	157	Ø	189	ç	221	ł	253	°
30	▲	62	>	94	^	126	~	158	×	190	¥	222	Ł	254	■
31	▼	63	?	95	_	127	Δ	159	ƒ	191	ı	223	ł	255	■

Saisie d'une chaîne de caractère

On utilisera le type *char* (et le format %s) en précisant le nombre de caractères entre crochets.

Exemple

```
/*déclaration*/
char chaine[10];
printf("Saisissez votre chaine de 10 caractères \n");
scanf("%s", chaine);
```

Remarque

Le nombre de caractères contenus par une variable de type chaîne de caractères peut être connu grâce à la fonction *strlen()* (Il faudra inclure la librairie *string.h* en tête de programme (*#include <string.h>*)).

strlen() renvoie un nombre entier ; il faudra placer le nom de la variable entre les parenthèses.

! Opérateurs de calcul

+ : addition

- : soustraction

* : multiplication

/ : division

% : reste de la division entière (7%2 donnera 1 comme résultat car $7 = 3 * 2 + 1$)



L'opérateur d'incrémentation : ++

$a = a + 1$; peut s'écrire $a++$; ou $++a$;

Premier cas

Attention au placement de l'opérateur ++. Si ce dernier est placé **avant** la variable, le contenu de la variable est modifié et ensuite le résultat de cette nouvelle affectation est mis dans la variable.

```
int x = 2, y;
```

```
y = ++x;
```

Après ces 2 lignes :

x vaut 3 et y vaut 3.

Détails : On affecte 2 à x ; on incrémente x qui passe à 3 puis on affecte le contenu de x à y donc y vaut 3, également.

Deuxième cas

Attention au placement de l'opérateur ++. Si ce dernier est placé **après** la variable, le contenu de la variable est incrémenté après.

```
int x = 2, y;
```

```
y = x++;
```

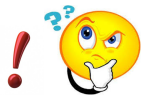
Après ces 2 lignes :

y vaut 2 et x vaut 3.

Détails : On affecte 2 à x ; on affecte le contenu de x à y donc y vaut 2 puis on incrémente x qui passe donc à 3.

Exemple

$a = a + 2$; peut s'écrire : $a += 2$;



L'opérateur de décrémentation : --

Pareillement, on pourra écrire $a--$; ou $-- a$; ce qui correspond à l'instruction : $a = a - 1$;



Autres opérateurs

+=

$a += b$ est équivalent à $a = a + b$;

-=

$a -= b + 2$ est équivalent à $a = a - (b + 2)$;

***=**

$A *= B + 2$;

est équivalent à : $A = A * (B + 2)$;

/=

$b /= a$;

est équivalent à $b = b / a$;

Autre exemple

$b *= ++a$; □

Cette instruction équivaut à :

$b = b * (++a)$

Soit encore :

$a = a + 1$;

$b = b * a$;

EXERCICES

Exercice 1

Énoncé

Écrire un programme qui affecte la lettre "c" à une variable nommée *Lettre* puis l'affiche.

Corrigé

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

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

```
int main()
{
    char Lettre;
    Lettre = 'c';
    printf("La variable Lettre contient : %c\n",Lettre);
    return 0;
}
```

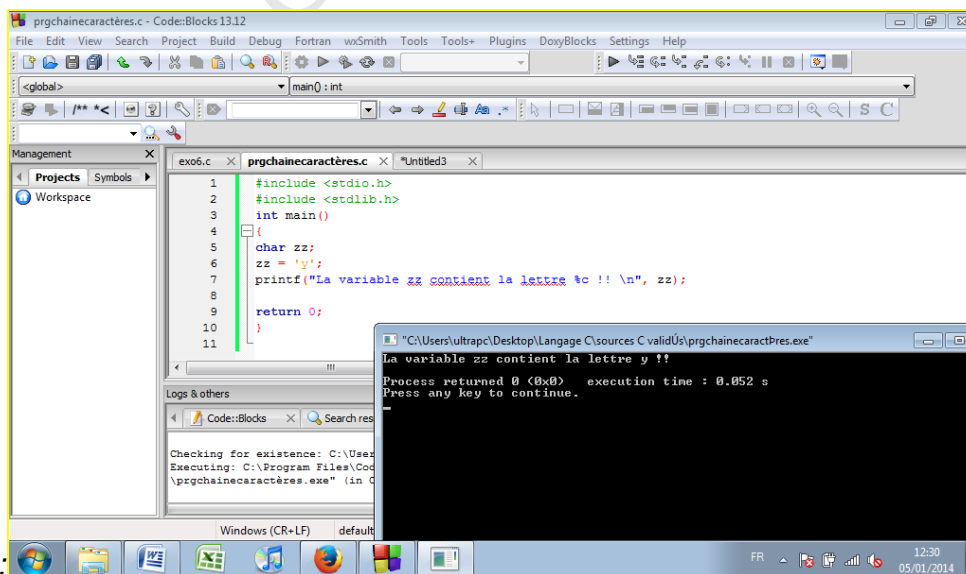
Exercice 2

Énoncé

Créez un programme source avec une variable nommée "zz" qui stockera la lettre 'y'. Après compilation, exécutez-le.

Corrigé

Voici le résultat à l'écran :



Ici, la variable `zz` a le type `char` (caractère). On lui affecte la lettre 'y' grâce à l'instruction `zz = 'y'`; Attention ! Pas de guillemets !

Exercice 3

Énoncé

Écrire le programme qui demande de saisir un caractère ou symbole au clavier et qui affiche ensuite son code ASCII. (Il suffit d'afficher l'entier correspondant au caractère saisi, via une conversion de type).

Corrigé

```
#include <stdio.h>
main()
{
    char lesymbole ;
    printf(" Saisissez votre symbole ?\n");
    scanf ("%c", &lesymbole);
    printf (" Votre symbole a pour code ASCII : %d\n", (int)lesymbole );
    return 0;
}
```

Exercice 4

Énoncé

Écrire le programme inverse qui demande de saisir un code ASCII et qui affiche ensuite le caractère ou symbole correspondant.

Corrigé

```
#include <stdio.h>
int main ()
{
    char symbole ;
    printf("Saisissez le code ASCII de votre symbole\n");
    scanf("%d", &symbole);
    printf(" Votre symbole ASCII de numero %d est : %c\n", symbole, symbole);
    return 0;
}
```

Exercice 5

Énoncé

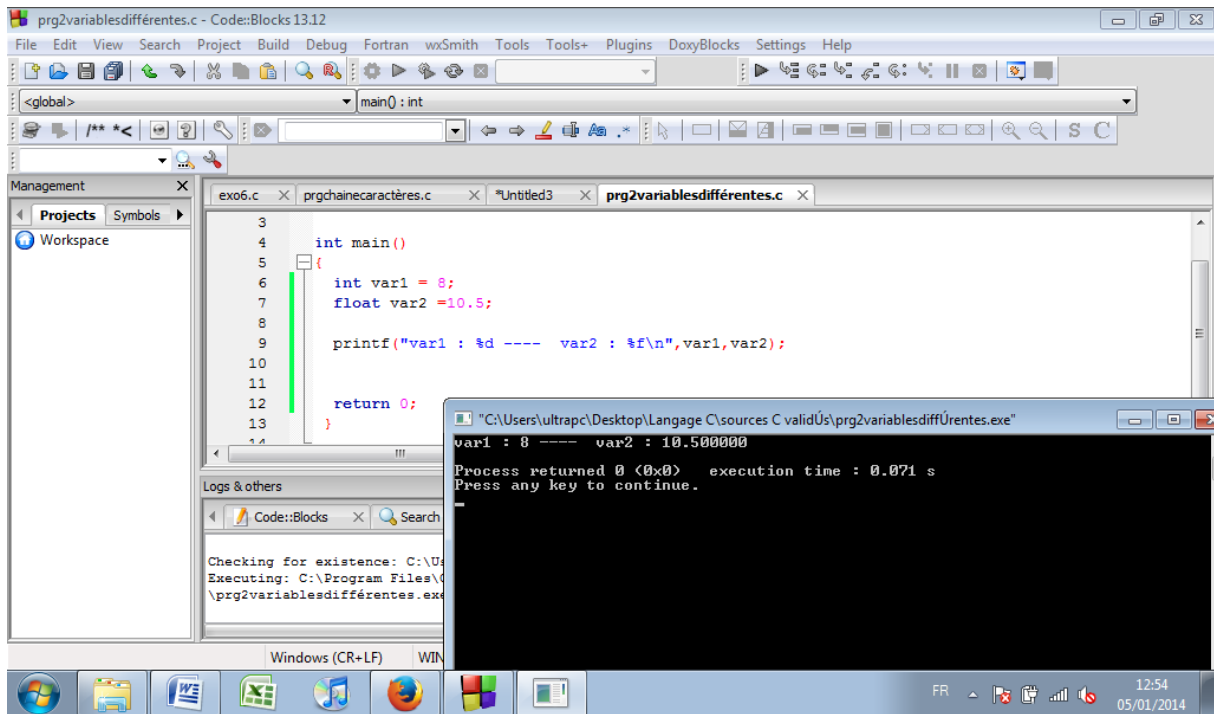
Créez un programme C qui affiche le contenu d'une variable `var1` qui contient la valeur entière 8 et d'une variable `var2` qui contient la valeur réelle 10,5.

Contrainte : vous ne devrez utiliser qu'une seule instruction `printf` !

Après compilation, exécutez le programme.

Corrigé

Voici le résultat à l'écran :



Exercice 6

Énoncé

Soit la fonction mathématique f définie par $f(x) = 3x^2 - 1$;

Écrire le programme qui demande la valeur x (nombre réel) et affiche la valeur de $f(x)$ associée (nombre réel). Par exemple, si on saisit 0.0 il s'affichera -1 ($3 * 0^2 - 1$) ; si on saisit 1.0 il s'affichera 2 ($3 * 1^2 - 1$).

Corrigé

```

#include <stdio.h>
int main()
{
    // fonction f(x)=3X^2-1
    float x;
    printf("saisir x\n ");
    scanf("%f",&x);
    printf("f(%f) = %f\n", x,(3*x*x-1));
    return 0;
}

```

Exercice 7

Énoncé

Stocker les réels 12.5 et 25.5 dans 2 variables nommées *FF* et *GG* puis afficher la somme des contenus de ces 2 variables sans utiliser d'autre variable.

Corrigé

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

```
int main()
{
    float FF = 12.5;
    float GG =25.5;
    printf("La somme des 2 variables vaut : %f\n", (FF+GG));
    return 0;
}
```

Exercice 8

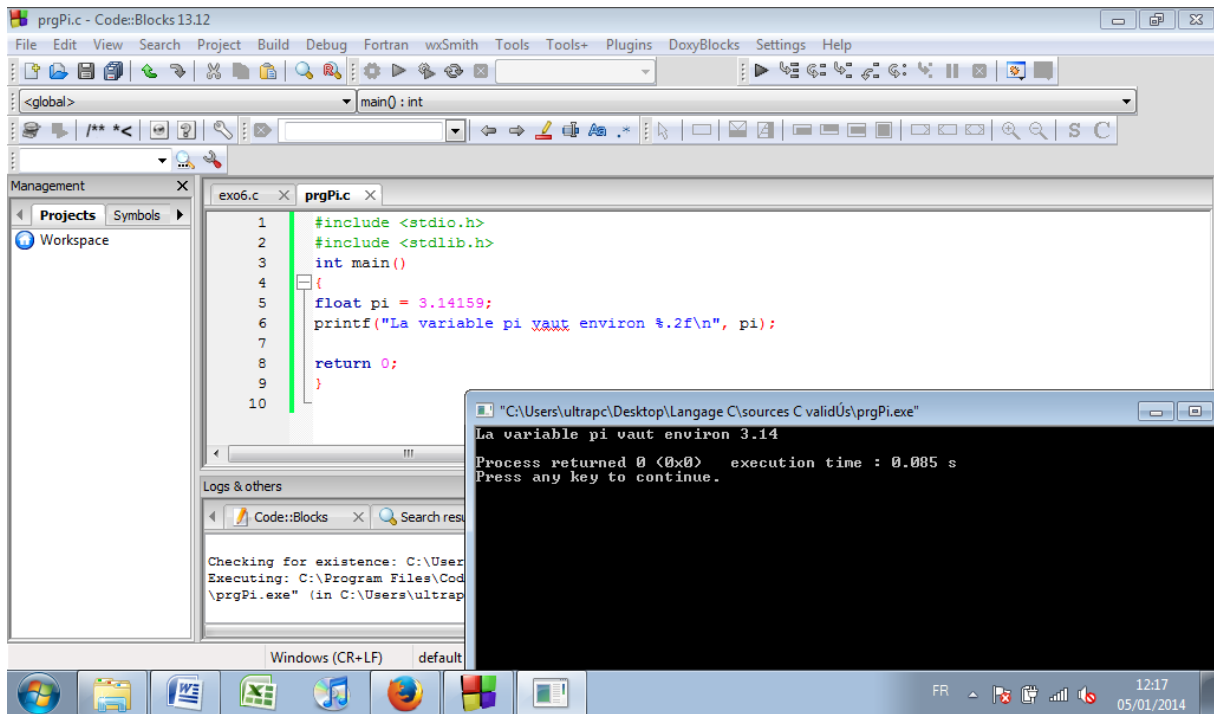
Énoncé

Créez le programme C qui affecte la valeur 3.14159 à une variable nommée *Pi* puis affiche cette dernière avec 2 chiffres après la virgule.

Corrigé

```
#include <stdio.h>
#include <stdlib.h>
int main()
{
    float pi = 3.14159;
    printf("La variable pi vaut environ %.2f\n", pi);
    return 0;
}
```

Voici le résultat à l'écran :



Exercice 9

Énoncé

Écrire un programme qui demande de saisir une somme (réel) en euros au clavier puis affiche "Vous avez xx euros", où xx est la somme en euros avec 2 décimales uniquement.

Corrigé

```

#include <stdio.h>
int main ()
{
    float somme;
    printf("Somme en euros ? \n");
    scanf("%f", &somme);
    printf("Vous avez : %.2f euros", somme);
    return 0;
}

```

Exercice 10

Énoncé

Écrire le programme qui demande de saisir son prénom au clavier.

Exemple : si la personne a saisi "Arthur", il s'affichera :

"Vous avez saisi Arthur. Est-ce bien cela ?"

Corrigé

```

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

```

```

int main()
{
    char prenom[10];
    printf("Saisissez votre prenom \n");
    scanf("%s", prenom);
    printf("Vous avez saisi : %s. Est-ce bien cela ? \n",prenom);
    return 0;
}

```

Exercice 11

Énoncé

Compléter le programme précédent pour qu'il affiche, en plus, le nombre de caractères que possède le prénom. Utilisez la fonction `strlen()`. Il faudra inclure la librairie `string.h` en tête de programme.

Corrigé

```

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
int main()
{

    char prenom[10];

    printf("Saisissez votre prenom \n");
    scanf("%s", prenom);

    printf("Vous avez saisi : %s. Est-ce bien cela ? \n",prenom);
    printf("Vous avez saisi %d caracteres", strlen(prenom));
    return 0;
}

```

Rq : `strlen` doit être utilisé après la saisie du prénom.

Exercice 12

Énoncé

1 yen japonais = 0,0078255407 euro. (1 euro vaut environ 130 yens)
 Écrire le programme qui demande de saisir une somme en euros et qui affiche le message suivant "Vous avez xx yens". Où xx est la somme saisie en euros.
 Afficher la somme en yens avec seulement 2 chiffres après la virgule !
 Réfléchissez bien car il y a plusieurs petits pièges.

Corrigé

```

#include <stdio.h>□□
int main ()□□
{□
    double somme;□□□
    printf("Somme en euros ? \n") ;□□
    scanf("%lf", &somme) ;□□
}

```

```
printf("Vous avez : %.2lf yens", (somme/0.0078255407) );  
return 0;  
}
```

Exercice 13

Énoncé

Qu'affiche exactement le programme ci-après ?

```
#include <stdio.h>  
int main ()  
{  
    int a;  
    int b = 2 + (a=3);  
    printf("a: %d ; b : %d\n", a, b) ;  
    b *= ++a;  
    printf("a: %d ; b : %d\n", a, b) ;  
    return 0;  
}
```

Corrigé

```
a: 3 ; b : 5  
a: 4 ; b : 20  
MBP-de-G:~ G$
```

Capture de l'affichage à l'écran

Exercice 14

Énoncé

Qu'affiche exactement le programme ci-après ?

```
#include <stdio.h>  
int main ()  
{  
    float a;  
    float b = 97 + (a=3.0);  
    printf("a: %f ; b : %f\n", a, b) ;  
    b /= ++a;  
    printf("a: %f ; b : %f\n", a, b) ;  
    return 0;  
}
```

Corrigé

```
a: 3.000000 ; b : 100.000000
a: 4.000000 ; b : 25.000000
MBP-de-G:~ G$
```

Capture de l'affichage à l'écran

Exercice 15

Énoncé

Écrire un programme qui demande le rayon R d'un cercle (réel) puis affiche la circonférence (périmètre) de ce dernier.

Rappel : circonférence = $2 \times \pi \times R$; on utilisera $\pi = 3,14156$

Corrigé

```
#include <stdio.h>
#include <stdlib.h>
int main()
{
    double PI = 3.14156;
    double rayon, circonference;
    printf("%lf", PI);
    printf("Tapez le rayon du cercle\n");
    scanf("%lf", &rayon);
    circonference = 2.0*PI*rayon;
    printf("La circonference de ce cercle vaut : %lf\n", circonference);
    return 0;
}
```

Exercice 16

Énoncé

Soit le code source C suivant :

```
int A1=4 ;
int A2 ;
A2 = (++A1 + 2);
A1 = A2++ + 1;
```

Que contiennent les variables A1 et A2 après ces 4 lignes de programme ?

Corrigé

A1 vaut 8 ; A2 vaut 8.

Exercice 17

Énoncé

Écrivez le programme source complet de l'exercice précédent ; vérifiez le résultat.

Corrigé

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

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

```
int main()
```

```
{
```

```
int A1=4;
```

```
int A2;
```

```
A2 = (++A1 + 2);
```

```
A1 = A2++ + 1;
```

```
printf("A1 vaut : %d\n",A1);
```

```
printf("A2 vaut : %d\n",A2);
```

```
return 0;
```

```
}
```

Exercice 18

Énoncé

Créer un programme de votre choix qui utilise une fois l'opérateur d'incrémentation et une fois l'opérateur de décrémentation (++ , une fois devant et -- une fois derrière).

Exercice 19

Énoncé

Créer un autre programme qui utilise une fois l'opérateur d'incrémentation de type +=, ou -= ou *= ou /= avec également un autre opérateur de d'incrémentation ++ ou de décrémentation --.