Libftprintf.a

Fichiers de rendu : *.c | */*.c | *.h | */*.h | Makefile (all, clean, clean, re, bonus)

Fonctions externes autorisées :

1. **Malloc, free et write**

2. **Va_arg, va_copy, va_end et va_start**

Entête : <stdio.h> | <stdarg.h>

Les macros va_arg, va_copy, va_end et va_start offrent un moyen portable d'accéder aux arguments d'une fonction lorsque la fonction accepte un nombre variable d'arguments.

Ces macros considèrent que la fonction accepte un nombre fixe d'arguments obligatoires, suivi d'un nombre variable d'arguments facultatifs.
- Les arguments obligatoires sont déclarés à la fin de la fonction en tant que paramètres ordinaires et sont accessibles via les noms des paramètres.
- Les arguments facultatifs sont accessibles via les marrons contenues dans STDARG.H (conforme à la norme ANSI C89.

Attention : Quand ils sont compilés à l'aide de /clr (compilation pour le Common Language Runtime), les programmes qui utilisent ces macros peuvent générer des résultats inattendus en raison des différences entre les systèmes de type natif et les systèmes de type CLR.

a. Va_arg : retourne la valeur actuelle

- Va_arg récupère une valeur de type à partir de l'emplacement donné par arg_ptr, et incrémente arg_ptr pour pointers vers l'arguments suivant dans la liste en utilisant la taille du type pour déterminer où commence l'argument suivant.

- Va_arg peut être utilisé un nombre quelconque de fois dans la fonction pour récupérer des arguments de la liste.

Syntaxe :

```
type va_arg (
        va_list arg_ptr,
        type
);
```

arg_ptr => pointeur désignant la liste d'arguments
type => type d'argument à récupérer

b. Va_copy : ne retourne par de valeur

- Va_copy effectue une copie d'une liste d'arguments dans son état actuel.

- Le paramètre src doit déjà être initialisé avec va_start, elle a peut-être été mise à jour avec des appels de va_arg, mais elle ne doit pas avoir été réinitialisée avec va_end.

- L'argument suivant extrait par va_arg de dest est le même que l'argument suivant extrait de src.

Syntaxe :

```
type va_copy (
        va_list dest,
        va_list src
);
```

dest => pointeur vers la liste d'arguments à initialiser à partir de src
src => pointeur vers la liste d'arguments initialisée à copier vers dest

c. Va_end : ne retourne par de valeur

- Une fois tous par arguments récupérés, va_end réinitialise le pointeur à la valeur NULL.
- Va_end doit être appelé sur chaque liste d'arguments initialisée avec va_start ou va_copy avant le retour de la fonction.

Syntaxe :

```
type va_start (
        va_list arg_ptr,
);
```

d. Va_start : ne retourne par de valeur

- Définit arg_ptr sur le premier argument facultatif de la liste d'argument passé à la fonction.
- L'argument arg_ptr doit être de type va_list.
- L'argument prev_param est le nom du paramètre requis qui précède immédiatement le premier argument facultatif dans la liste d'arguments (si prev_param est déclarée avec la classe de stockage Register, le comportement de la macro n'est pas défini).
- Attention : va_start doit être utilisé avant que va_arg soit utilisé pour la premier fois.

Syntaxe :

```
type va_arg (
        va_list arg_ptr,        prev_param => paramètre qui précède
        prev_param              le premier argument facultatif
);
```

Ou :

```
type va_arg (
        arg_ptr,
);
```

05/02/21-09/02/21 Melody Boy

Les conversions a gérer :

1. **%c : Caractère (avec printf, spécifie un caractère codé sur un octet)**
   -> ft_putchar.c

2. **%s : String (avec printf, spécifie un chaîne de caractère codés sur un octet ou multi-octets)**
   -> ft_putstr.c

3. **%p : Type de pointeur (affiche l'argument en tant qu'adresse en chiffres hexadécimaux)**
   -> ft_putpointer.c

4. **%d : Integer (entier décimal signé de type int)**
   -> ft_putnbr_signed.c

5. **%i : Integer (entier décimal signé de type long)**
   -> ft_putnbr_long_signed.c

6. **%u : Integer (entier décimal non signé)**
   -> ft_putnbr_unsigned.c

7. **%x : Integer (entier hexadécimal non signé : utilise "abcdef")**
   -> ft_putnbr_hex_signed.c

8. **%X : Integer (entier hexadécimal non signé : utilise "ABCDEF")**
   -> ft_putnbr_hex_unsigned.c

05/02/21-09/02/21 Melody Boy

Flags et taille de champ minimale :

1. **- : Aligne à gauche le résultat selon la largeur de champ donné (par default, c'est à droit)**
   -> ft_aligned_left.c

2. **0 : Si la longueur est préfixée par 0, des zéros non significatifs sont ajoutés jusqu'a ce que la largeur minimale soit atteinte**
   - Si "0" et "-" apparaissent, "0" est ignoré.
   - Si "0" et spécifié par un format d'entier (d, i, u, x et X)et q'une spécification de précision est également présente (ex : %04.d) "0" est ignoré.
     -> ft_zero_len.c

3. *** | .* : La longer n'est pas spécifiée dans la chaîne de formatage, mais sous la forme d'un argument de valeur entière supplémentaire précédant l'argument à mettre en forme**
   -> ft_len_entier.c

4. **. : Spécification de précision**
   -> ft_spe_precision.c

a. Spécification de largeur :

Dans une spécification de conversion, le champ facultatif de spécification de largeur apparaît après n'importe quel caractère d'indicateur.

L'argument width est un entier décimal non négatif qui contrôle le nombre de minimal de caractères qui sont générés :

- Si le nombre de caractères dans la valeur de sortie est inférieur à la largeur spécifié, des espaces sont ajoutés à gauche ou à droite des valeurs, selon si l'indicateur d'alignement à gauche ("-") est spécifié ou non, jusqu'à ce qui la largeur minimale soit atteinte.

- Si width est préfixé par 0, des zéros non significatifs sont ajoutés aux conversions en entier ou en nombres à virgule flottante jusqu'à ce que la largeur minimale soit atteinte, sauf en cas de conversion en valeur infinie ou NaN.

La spécification de largeur ne provoque jamais la troncature d'une valeur. Si le nombre de caractères dans la valeur de sortie est supérieur à la largeur spécifiée, ou si width n'est pas fourni, tous les caractères de la valeur sont générés, selon la spécification de précision.

Si la spécification de la largeur est une astérisque ("*"), un argument int issu de la liste d'arguments fournit la valeur. L'argument width doit précéder la valeur mise en forme dans la liste des arguments (ex : printf("%0*d", 5, 3); /* 00003 is output */).

Une valeur width manquante ou petite dans une spécification de conversion n'entraîne pas la troncation d'une valeur de sortie. Si le résultat d'un conversion est plus grande que la valeur width, le champ peut-être développé pour contenir le résultat de la conversion.

b. Spécification de précision :

Dans une spécification de conversion, le troisième champ facultatif concerne la spécification de précision. Il se compose d'un point (".") suivi d'un entier décimal non négatif qui, selon le type de conversion, spécifie le nombre de caractères de chaîne, le nombre de décimales ou le nombre de chiffres significatifs à générer.

Contrairement à la spécification de largeur, la spécification de précision peut entraîner la troncation de la valeur de sortie ou l'arrondie d'une valeur à virgule flottante. Si vous spécifiez 0 comme précision et que la valeur à convertir est 0, vous n'obtenez aucune sortie de caractères (ex : printf("%.0d", 0); /* No characters output */).

Si la spécification de précision est une astérisque, un argument int issu de la liste d'arguments fournit la valeur. Dans la liste d'arguments, l'argument précision doit précéder la valeur mise en forme (ex : printf("%.*f", 3, 3.14159265); /* 3.142 output */).

Le caractère type détermine soit l'interprétation de précision, soit la précision par défaut quand précision est omis :

Type	Signification	Default
c	La précision n'a aucun effet	Le caractère est imprimé@
d, l, u, x et X	La précision indique le nombre minimal de chiffres à imprimer. Si le nombre de chiffre dans l'argument est inférieur à précision, la valeur de sortie est remplie à gauche de zéros. La valeur n'est pas tronquée lorsque le nombre de chiffres dépasse la précision.	La précision par default s'élève à 1.
s	La précision indique le nombre maximal de caractères à imprimer. Les caractères au-delà de la précision ne sont pas imprimés.	Les caractères sont imprimés jusqu'à ce qu'un caractère nul soit trouvé.

Mise en pratique : Integer et Long (tous les flags)

Natation for Integer (cela fonctionne également pour le Long : spécifié par %i)	Sortie	Valeur de Retour	Note
printf("%d", 0);	0	1	Affiche 0 et retourne la taille de 0
printf("%d", 6);	6	6	La valeur de retour est égale à la taille de nombre retourné et celle du \0
printf("%d", -6);	-6	2	//
printf("%d", 12);	12	2	//
printf("%d", -12);	-12	3	//
With the Width Option	Résultats	Valeur de Retour	Note
printf("%5d", 0);	_ _ _ _ 0	5	La fonction affiche le nombre précédé par la with moins la taille du nombre et la valeur de retour est égale à la taille de nombre retourné
printf("%5d", 12);	_ _ _ 12	5	//
printf("%5d", -12);	_ _ -12	5	//
printf("%-5d", -12);	-12 _ _	5	Si un Minus se trouve avant le Width les espaces sont rajoutés à la fin
printf("%5d", 12345);	12345	5	Quand la with est inférieure à la taille du nombre, le nombre seul est affiché et sa taille
printf("%-5d", -12345);	-12345	6	//
With the Zero-Fill Option	Résultats		Note
printf("%05d", 0);	0 0 0 0 0	5	La fonction affiche le nombre précédé par le Zero-Fill moins la taille du nombre et la valeur de retour est égale à la taille de nombre retourné et celle du \0
printf("%05", 4);	0 0 0 0 4	5	//
printf("%05", -4);	- 0 0 0 4	5	Lorsque le nombre est négatif, le moins est positionné avant les Zero-Fill et compte pour 1 dans la taille du nombre
printf("%-05d", 4);	Warning : Flag '0' is ignored when flag '-' is present		
printf("%05", 12345);	12345	5	Quand la Zero-Fill est inférieure à la taille du nombre, le nombre seul est affiché et sa taille
printf("%05d", -12345);	-12345	6	//
With the Minus Option	Résultats	Valeur de Retour	Note
printf("%-d", 12);	12	2	Le nombre et sa longueur sont retourné
printf("%-5d", 4);	4 _ _ _	5	Les Widths sont placés le nombre
printf("%-5", -4	-4 _ _	5	//
printf("%-05d", -);	Warning	Warning	Le flag Zero-Fill est ignoré quand le flag Minus est présent
With the Asterisk Option	Résultats	Valeur de Retour	Note
printf("%*d", 5, 0);	_ _ _ _ 0	5	Fonctionne comme le flag Width
printf("%*d", 5, 7);	_ _ _ _7	5	//
printf("%*d", 5, -7);	_ _ _ -7	5	//
printf("%*d", 5, 123456);	123456	5	//
printf("%*d", -5, 10);	-10 _	5	//
printf("%*d", -5, -12345);	-12345	6	//
printf("%0*d", 5, 5);	0 0 0 0 5	5	Lorsque le flan '0' se trouve avant le flan '*', le résultat se comporte comme avec le flan '0' seul
With Int/Long Min	Résultats	Valeur de Retour	Note
printf("%d", -2147483648);	Warning : Format specifies type 'int' but the argument has type 'long' (use %ld)		-2147483647 works (returns 12)
With Int/Long Max	Résultats		Note
printf("%d", 2147483647)	2147483647	10	Warning for 2147483648 : Format specifies type 'int' but the argument has type 'long' (use %ld)

Mise en pratique : Unsigned Int

Natation for Integer (cela fonctionne également pour le Long : spécifié par %i)	Sortie	Valeur de Retour	Note
printf(“%d”, 0);	0	1	Affiche 0 et retourne sa taille
printf(“%d”, 6);	6	1	La valeur de retour est égale à la taille de nombre retourné
printf(“%d”, 12);	12	2	//
With the Width Option	Résultats	Valeur de Retour	Note
printf(“%5d”, 0);	_ _ _ _ 0	5	La fonction affiche le nombre précédé par la with moins la taille du nombre et la valeur de retour est égale à la taille de nombre retourné
printf(“%5d", 12);	_ _ _ 12	5	//
printf(“%5d”, 12345);	12345	5	Quand la with est inférieure à la taille du nombre, le nombre seul est affiché et sa taille
With the Zero-Fill Option	Résultats		Note
printf(“%05d”, 0);	0 0 0 0 0	5	La fonction affiche le nombre précédé par le Zero-Fill moins la taille du nombre et la valeur de retour est égale à la taille de nombre retourné
printf(“%05”, 4);	0 0 0 0 4	5	//
printf(“%05", 12345);	12345	5	Quand la Zero-Fill est inférieure à la taille du nombre, le nombre seul est affiché et sa taille est retourné
With the Minus Option	Résultats	Valeur de Retour	Note
printf(“%-d”, 12);	12	2	Le nombre et sa longueur sont retourné
printf(“%-5d”, 4);	4 _ _ _	5	Les Widths sont placés le nombre
With the Asterisk Option	Résultats	Valeur de Retour	Note
printf(“%*d”, 5, 0);	_ _ _ _ 0	5	Fonctionne comme le flag Width
printf(“%*d”, 5, 7);	_ _ _ _7	5	//
printf(“%*d”, 5, 123456);	123456	6	//
printf("%*d", -5, 10);	-10 _	5	//
With Int/Long Min	Résultats	Valeur de Retour	Note
printf(“%d”, 0);	0	1	Tous chiffres négatifs ne marche pas (unsigned)
With Int/Long Max	Résultats		Note
printf(“%d”, 2147483647)	2147483647	10	Warning for 2147483648 : Format specifies type ‘int’ but the argument has type ‘long’ (use %ld)

Mise en pratique : Hexadécimal (majuscule et minuscule)

Natation for Hexadémale (%x comme %X)	Résultats	Valeur de Retour	Note
printf(“%x”, 14);	e	1	La valeur est retourné mais cette fois-ci en hexadécimal avec des *minuscule, et la valeur de retour est égale à la longueur du nombre
printf(“%X”, 14);	E	1	La valeur est retourné mais cette fois-ci en hexadécimal avec des *majuscule, et la valeur de retour est égale à la longueur du nombre
printf(“%x”, 12345678);	bc614e	6	*minuscule
printf(“%X”, 12345678);	BC614E	6	*majuscule
With the Width Option	Résultats	Valeur de Retour	Note
printf(“%5x”, 14);	_ _ _ _ e	5	Si le nombre est inférieur au Width, le résultat est complété par des espaces au début et la valeur et soit égale à la Width si elle est supérieur à la taille du nombre soit, dans le cas contraire, elle est égale à la taille du nombre
printf(“%5X”, 14);	_ _ _ _E	5	//
printf(“%5x”, 12345678);	bc614e	6	Si la width est inférieure à la taille du nombre, le résultat n’est pas impacté, et la valeur de retour est la longueur du nombre
With the Zero-Fill Option	Résultats	Valeur de Retour	Note
printf(“%05x”, 0);	0 0 0 0 0	5	Si le nombre est inférieur au Zero-Fill, le résultat est complété par des zéros au début, et la valeur et soit égale à la Width si elle est supérieur à la taille du nombre soit, dans le cas contraire, elle est égale à la taille du nombre
printf(“%05x, 14);	0 0 0 0 e	5	//
printf(“%05d”, 12345678);	BC614E	6	Si le nombre est supérieur au Zero-Fill Option, le résultat n’est pas impacté, et la valeur de retour est la longueur du nombre
With the Minus Option	Résultats	Valeur de Retour	Note
printf(“%-5x”, 14);	e_ _ _ _	5	Fonctionne de la même façon que the Width option, mais rajoute des espaces avec le nombre que si ce dernière est inférieur au chiffre/nombre précisé avec le Minus, la valeur de retour est soit la Width soit la taille du nombre
printf(“%-5X”, 14);	E_ _ _	5	//
printf(“%-5x”, 12345678);	bc614e	6	Si le nombre est inférieur au Minus, le résultat n’est pas impacté et la valeur de retour est égale à la length du nombre
printf(“%-5X”, 12345678);	BC614E	6	//
With the Asterisk Option	Résultats	Valeur de Retour	Note
printf(“%*x”, 5, 14);	_ _ _ _ e	5	L’astérisque fonctionne de la même manière que le Width option, sauf que cette fois la Width sera spécifié après en tant qu’argument indépendant, et la valeur de retour est égale à la longueur du nombre si ce dernier est supérieur à la Width sinon la taille de la with est renvoyé
printf(“%*X”, 5, 14);	_ _ _ E	5	//
printf(“%*x”, 5, 12345678);	bc614e	6	Si le nombre est supérieur a l’astérisque, le résultat n’est pas impacté, la valeur de retour est la length du nombre
printf(“%*X, 5, -12345678);	BC614E	6	//

Mise en pratique : Character (flags Width, Zero_Fill et Minus)

Notation for Character	Résultat	Valeur de Retour	Note
printf(“%c”, ‘a’);	a	1	Le caractère est retourné, et la valeur de retour est de 1
With the Width Option	**Résultat**	**Valeur de Retour**	**Note**
printf(“%5c”, ‘a’);	_ _ _ _ a	5	Le caractère est précédé par la valeur de Width mois un caractère, et la valeur de retour est égale à la Width
With the Zero-Fill Option	**Résultat**	**Valeur de Retour**	**Note**
printf(“%05c”, ‘a’);	Warning : Flag ‘0’ results in undefined behaviour with ‘c’ conversion specifier Retourne 0000a		Malgré le warning la fonction compile est sort le résultat : 0000a (le caractère est précédé par le nombre de zéro spécifié moins la taille du caractère)
printf(“%0c”, ‘a’);	Warning : Flag ‘0’ results in undefined behaviour with ‘c’ conversion specifier Retourne : a		Malgré le warning la fonction compile est sort le résultat : a (quand l’option Zero-Fill est simplement de 0, le caractère est retourné seul)
With de Minus Option	**Résultat**	**Valeur de Retour**	**Note**
printf(“%-5c”, ‘a’);	c_ _ _ _	5	Le caractère est retourné suivit d’un espace autant de fois que le nombre spécifié après le minus moins le caractère, la valeur de retour reste la width
printf(“%-c”, ‘a’);	c	1	Le caractère est retourné, et la fonction retourne 1
With the Astericks Option	**Résultat**	**Valeur de Retour**	**Note**
printf(“%*c”, 5, ’a’);	_ _ _ _ a	5	Comme l’option Width, le caractère est précédé par la valeur attribué à l’astérisque (valeur spécifié indépendemment comme un argument) mois un caractère, la valeur de retour reste la width
With Character Max	**Résultat**		**Note**
printf(“%c”, 128);	?	1	Au-dessus des valeurs de la table ascii (de 0 a 127), la fonction retourne un point d’interrogation
printf("%c", 0);		1	Pour la valeur minimale de la table ascii, la fonction fonctionne correctemen, la valeur de retour est de 1

Mise en pratique : String (flags Width, Minus et Astérisks)

Natation for Strings	Résultat	Valeur de Retour	Note
printf(“%s”, “Bonjour”);	Bonjour	7	La string est rétournée, et la valeur de retour est la length de la string
With de Width Option	**Résultat**	**Valeur de Retour**	**Note**
printf(“%5s”, “Bonjour”);	Bonjour	7	La string est retournée même si la Width est inférieur à la taille de la string, et la valeur de retour est la length de la string
printf(“%10s”, “Bonjour”);	_ _ _ Bonjour	10	La string est précédée par la valeur de Width mois la taille de la string, la valeur de retour est la Width
With de Zero-Fill Option	**Résultat**	**Valeur de Retour**	**Note**
printf(“%05s”, “Bonjour”);	Warning : Flag ‘0’ results in undefined behaviour with ’s’ conversion specifier Retourne : Bonjour		Malgré le warning, la fonction retourne la string entière
printf(“%05s”, “Bonjour”);	Warning : Flag ‘0’ results in undefined behaviour with ’s’ conversion specifier Retourne : Bonjour		Malgré le warning, la fonction retourne la string entière
printf(“%010s”, “Bonjour”);	Warning : Flag ‘0’ results in undefined behaviour with ’s’ conversion specifier Retourne : 000Bonjour		Malgré le warning, la fonction retourne la string entière précédé par le nombre de Zero-Fill point la taille de la string
With de Minus Option	**Résultat**	**Valeur de Retour**	**Note**
printf(“%-5s”, “Bonjour”);	Bonjour	5	Retourne la string entière, et la valeur de retour est la length de la string
printf(“%-s”, “Bonjour”);	Bonjour	5	Retourne la string entière, et la valeur de retour est la length de la string
printf(“%-10s”, “Bonjour”);	Bonjour_ _ _	10	Retourne la string entière suivit du nombre de Minus moins la taille de la string, la valeur de retour est la Width
With de Astericks Option	**Résultat**		**Note**
printf(“%*s”, 5, “Bonjour”);	Bonjour	5	Retourne la string en entière, et la valeur de retour est la length de la string
printf(“%*s”, 10, “Bonjour”);	_ _ _ Bonjour	10	Retourne la string précédé par la valeur attribué à l’astérisque moins la taille de la string, la valeur de retour est la Width

Mise en pratique : Combinaisons

Combinaison	Résultat	Valeur de Retour
printf(“%5d%s”, 12345, “Hello”);	1234Hello	10
printf(“%5d %s”, 12345, “Hello”);	1234 Hello	11
printf(“%5d\t%s”, 12345, “Hello”);	1234\tHello	11
printf(“Hello my name is %s, I am %u.”, “Melody”, 19);	Hello my name is Melody, I am 19.	33
printf(“The equivalent to %d, in hex, is %X”, 12345678, 12345678);	The equivalent to 12345678, in hex, is BC614E	45
printf(“The equivalent to %d, in hex, is %x”, 12345678, 12345678);	The equivalent to 12345678, in hex, is bc614e	45
printf(“Test with Asterisk Option : %*d\n %s”, 5, 123, “ok”);	Test with Asterisk Option : _ _123 _ok	37