ft_isalpha

Teste les caractères alphabétiques.

```
int main(void)
{
    printf("%d\n", ft_isalpha('A'));
    printf("%d\n", ft_isalpha('1'));
    printf("%d\n", ft_isalpha(-1));
    return (0);
}
```

ft_isdigit

Teste les caractères numériques.

```
int main(void)
{
    printf("%d\n", ft_isdigit('A'));
    printf("%d\n", ft_isdigit('1'));
    printf("%d\n", ft_isdigit(-1));
    return (0);
}
```

ft_isalnum

Teste les caractères alphanumériques.

```
int main(void)
{
    printf("%d\n", ft_isalnum('A'));
    printf("%d\n", ft_isalnum('1'));
    printf("%d\n", ft_isalnum(-1));
    return (0);
}
```

ft_isascii

Teste les caractères ASCII.

```
int main(void)
{
    printf("%d\n", ft_isascii('A'));
    printf("%d\n", ft_isascii('1'));
    printf("%d\n", ft_isascii(-1));
    return (0);
}
```

ft_isprint

Teste les caractères imprimables.

```
int main(void)
{
    printf("%d\n", ft_isprint('A'));
    printf("%d\n", ft_isprint('1'));
    printf("%d\n", ft_isprint(-1));
    return (0);
}
```

ft_toupper

Convertit un caractère en majuscule.

```
int main(void)
{
    printf("%d\n", ft_toupper('A'));
    printf("%d\n", ft_toupper('1'));
    printf("%d\n", ft_toupper(-1));
    return (0);
}
```

ft_tolower

Convertit un caractère en minuscule.

```
int main(void)
{
    printf("%d\n", ft_tolower('A'));
    printf("%d\n", ft_tolower('1'));
    printf("%d\n", ft_tolower(-1));
    return (0);
}
```

ft_memset

Remplit une zone mémoire.

```
int main(void)
{
    char buf[10] = "abcdefghi";
    ft_memset(buf, 'X', 5);
    printf("%s\n", buf);
    return (0);
}
```

ft_bzero

Met une zone mémoire à zéro.

```
int main(void)
{
    char buf[10] = "abcdef";
    ft_bzero(buf, 3);
    for (int i = 0; i < 6; i++)
        printf("%d ", buf[i]);
    printf("\n");
    return (0);
}
```

ft_memcpy

Copie mémoire sans chevauchement.

```
int main(void)
{
    char src[10] = "abcdef";
    char dst[10];
    ft_memcpy(dst, src, 6);
    printf("%s\n", dst);
    return (0);
}
```

ft_memmove

Copie mémoire avec chevauchement.

```
int main(void)
{
    char str[20] = "123456789";
    ft_memmove(str+2, str, 5);
    printf("%s\n", str);
    return (0);
}
```

ft_memchr

Cherche un caractère dans une zone mémoire.

```
int main(void)
{
    char str[] = "abcdef";
    printf("%s\n", (char *)ft_memchr(str, 'c', 6));
    return (0);
}
```

ft_memcmp

Compare deux zones mémoire.

```
int main(void)
{
    printf("%d\n", ft_memcmp("abc", "abc", 3));
    printf("%d\n", ft_memcmp("abc", "abd", 3));
    return (0);
}
```

ft_strlen

Renvoie la longueur d'une chaîne.

```
int main(void){ printf("%zu\n", ft_strlen("Hello")); printf("%zu\n", ft_strlen("")); return(0); }
```

ft_strdup

Duplique une chaîne.

```
int main(void){ char *s = ft_strdup("test"); printf("%s\n", s); free(s); return(0); }
```

ft_strchr

Cherche un caractère dans une chaîne.

```
int main(void){ printf("%s\n", ft_strchr("Hello", 'l')); return(0); }
```

ft_strrchr

Cherche le dernier caractère dans une chaîne.

```
int main(void){ printf("%s\n", ft_strrchr("Hello", 'l')); return(0); }
```

ft_strncmp

Compare deux chaînes.

```
int main(void){ printf("%d\n", ft_strncmp("abc","abd",3)); return(0); }
```

ft_strnstr

Cherche une sous-chaîne dans une autre.

```
int main(void){ printf("%s\n", ft_strnstr("hello world","world",11)); return(0); }
```

ft_strlcpy

Copie de chaîne limitée.

```
int main(void){ char dst[10]; printf("%zu\n", ft_strlcpy(dst,"Hello",10)); printf("%s\n", dst); retu
```

ft_strlcat

Concaténation limitée.

```
int main(void){ char dst[20]="Hi "; printf("%zu\n", ft_strlcat(dst,"there",20)); printf("%s\n", dst)
```

ft_substr

Sous-chaîne d'une chaîne.

```
int main(void){ char *s = ft_substr("HelloWorld",5,3); printf("%s\n", s); free(s); return(0); }
```

ft_strjoin

Concatène deux chaînes.

```
int main(void){ char *s = ft_strjoin("Hello ","World"); printf("%s\n", s); free(s); return(0); }
```

ft_strtrim

Supprime les caractères donnés du début et de la fin.

```
int main(void){ char *s = ft_strtrim("  **Hello**  "," *"); printf("%s\n", s); free(s); return(0); }
```

ft_split

Découpe une chaîne selon un séparateur.

```
int main(void){ char **t = ft_split("Hello world test",' '); for (int i=0;t[i];i++){printf("%s\n",t[
```

ft_strmapi

Applique une fonction à chaque caractère.

```
int main(void){ char f(unsigned int i,char c){return(c+1);} char *s=ft_strmapi("abc",f);printf("%s\n
```

ft_striteri

Applique une fonction sur chaque caractère (in-place).

```
int main(void){ void f(unsigned int i,char *c){*c=*c+1;} char s[]="abc";ft_striteri(s,f);printf("%s\
```

ft_atoi

Convertit une chaîne en entier.

```
int main(void)
{
    printf("%d\n", ft_atoi("42"));
    printf("%d\n", ft_atoi("   -42"));
    printf("%d\n", ft_atoi("abc42"));
    return (0);
}
```

ft_itoa

Convertit un int en chaîne.

```
int main(void)
{
    char *s = ft_itoa(42);
    printf("%s\n", s);
    free(s);
    s = ft_itoa(-42);
    printf("%s\n", s);
    free(s);
    return (0);
}
```

ft_putchar_fd

Affiche un caractère sur fd.

```
int main(void){ ft_putchar_fd('A',1); return(0); }
```

ft_putstr_fd

Affiche une chaîne sur fd.

```
int main(void){ ft_putstr_fd("Hello\n",1); return(0); }
```

ft_putendl_fd

Affiche une chaîne suivie d'un \n sur fd.

```
int main(void){ ft_putendl_fd("World",1); return(0); }
```

ft_putnbr_fd

Affiche un nombre sur fd.

```
int main(void){ ft_putnbr_fd(12345,1); ft_putchar_fd('\n',1); return(0); }
```