### Quiz

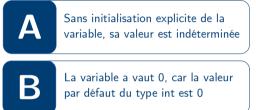
## Programmation impérative C

2025-2026

**Variables** 

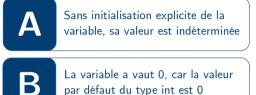
#### Quelle est la valeur initiale de a ?

int a;



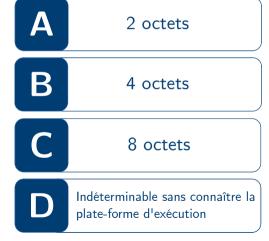
#### Quelle est la valeur initiale de a ?

int a;



### Quelle est la taille (en octets) de a ?

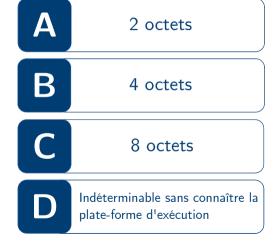
int a = 10;



#QDLE#Q#ABCD\*#30#

#### Quelle est la taille (en octets) de a ?

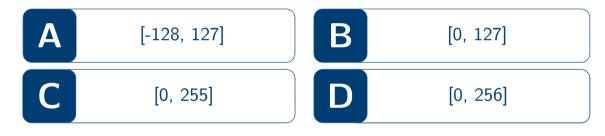
int a = 10;



# Quelle est la plage d'entiers que l'on peut coder avec une variable de type *char* ?



## Quelle est la plage d'entiers que l'on peut coder avec une variable de type *char* ?



# Quel type de données est le plus approprié pour stocker la valeur -1200 ?



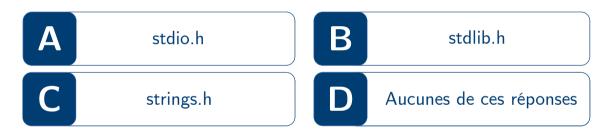
## Quel type de données est le plus approprié pour stocker la valeur -1200 ?



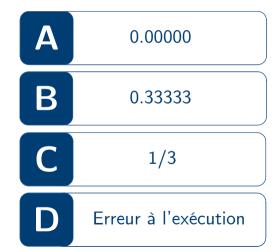
# Quelle est la bibliothèque qui contient le prototypage de la fonction *printf* ?



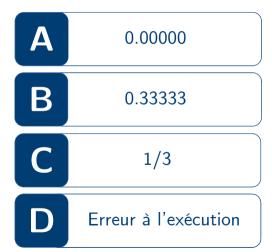
## Quelle est la bibliothèque qui contient le prototypage de la fonction *printf* ?



```
double result = 1 / 3;
printf("%lf\n", result);
```



```
double result = 1 / 3;
printf("%lf\n", result);
```



```
int main(){
                                              equal
  float a = 0.1:
  if (a == 0.1)
       printf("equal\n");
                                            not equal
  else
       printf("not equal\n");
                                         On ne sait pas, ça
  return 0:
                                         dépend du système
```

```
int main(){
                                              equal
  float a = 0.1:
  if (a == 0.1)
       printf("equal\n");
                                            not equal
  else
       printf("not equal\n");
                                         On ne sait pas, ça
  return 0:
                                         dépend du système
```

```
int main() {
                                              OK
  int a = 2-3:
  if (a)
       printf("OK\n");
                                              KO
  else
        printf("K0\n");
                                      Erreur à la compilation
  return 0:
                                      Erreur de segmentation
```

#QDLE#Q#A\*BCD#40#

```
int main() {
                                              OK
  int a = 2-3:
  if (a)
       printf("OK\n");
                                              KO
  else
        printf("K0\n");
                                      Erreur à la compilation
  return 0:
                                      Erreur de segmentation
```

```
int main() {
  printf("Hello World! %d\n", x);
                                                   Hello World! x:
  return 0:
                                               Hello World! suivi par une
                                                    valeur aléatoire
                                                     Hello World!
                                               Erreur à la compilation
```

#QDLE#Q#ABCD\*#40#

```
int main() {
  printf("Hello World! %d\n", x);
                                                    Hello World! x:
  return 0:
                                               Hello World! suivi par une
                                                    valeur aléatoire
                                                     Hello World!
                                               Erreur à la compilation
```

**Fonctions** 

# Est-il possible de définir des valeurs par défaut pour les paramètres d'une fonction ?

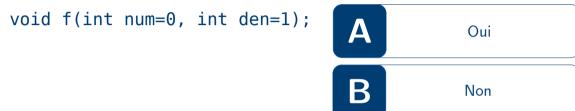
void f(int num=0, int den=1);

A Oui

B Non

# Est-il possible de définir des valeurs par défaut pour les paramètres d'une fonction ?

void f(int num=0, int den=1);



```
int test(int c) {
  printf("c\n");
  return c:
int main() {
  test(2+3);
  return 0;
```

5 Erreur de segmentation Erreur à la compilation

#QDLE#Q#AB\*CD#45#

5

```
int test(int c) {
  printf("c\n");
  return c:
int main() {
  test(2+3);
                                         Erreur de segmentation
  return 0;
                                         Erreur à la compilation
```

```
int func(int a) {
  a = 0:
  return a:
int main() {
   int a = 1:
   func(a):
                                         Erreur de segmentation
   printf("%d\n",a);
   return 0:
                                         Erreur à la compilation
```

#QDLE#Q#AB\*CD#45#

```
int func(int a) {
  a = 0:
  return a:
int main() {
   int a = 1:
   func(a);
                                         Erreur de segmentation
   printf("%d\n",a);
   return 0:
                                         Erreur à la compilation
```

**Boucles et Tableaux** 

## La boucle suivante est-elle syntaxiquement correcte ?

```
for(int i=0; i<10; i++) {
    printf("%d - ", i);
}</pre>

    Oui, cette syntaxe n'est valable
    qu'en C++ (j'imagine)

Oui, cette syntaxe (reprise de
    C++) existe depuis la version C
    ISO99
```

## La boucle suivante est-elle syntaxiquement correcte ?

#### Est-ce que ce code compile sans erreur ?

```
int main() {
   int i;
   {
     int i=0;
   }
   return 0;
}
```

```
Dépend de la norme C mise en
 œuvre par les compilateurs
           Oui
           Non
Aucunes de ces réponses
```

#### Est-ce que ce code compile sans erreur ?

```
int main() {
    int i;
    {
       int i=0;
    }
    return 0;
}
```



```
int main () {
                                               5 5
 int i = 0, j = 10;
 for(int i=0; i<=j; i++, j--)
 printf("%d %d", i, j);
 return 0:
                                               0.10
                                               0 4
```

#QDLE#Q#ABCD\*#80#

```
int main () {
                                              5 5
 int i = 0, j = 10;
 for(int i=0; i<=j; i++, j--)
 printf("%d %d", i, j);
 return 0:
                                              0.10
```

```
int a = 3;
                                                3 - 7 - 10
int b = 7;
int r = a++ + b++;
printf("%d-%d-%d", a, b, r);
                                                3 - 7 - 12
                                                4-8-10
                                                4-8-12
```

#QDLE#Q#ABC\*D#35#

```
int a = 3;
                                            3-7-10
int b = 7;
int r = a++ + b++;
printf("%d-%d-%d", a, b, r);
                                            3-7-12
                                            4-8-10
                                            4-8-12
```

```
int main() {
  a = 5:
  printf("%d\n", a++);
  return 0;
```

Erreur à la compilation

```
int main() {
  a = 5;
  printf("%d\n", a++);
  return 0:
                                     Erreur à la compilation
```

```
int a = 3;
                                                3 - 7 - 10
int b = 7;
int r = ++a + ++b;
printf("%d-%d-%d", a, b, r);
                                                3 - 7 - 12
                                                4-8-10
                                                4-8-12
```

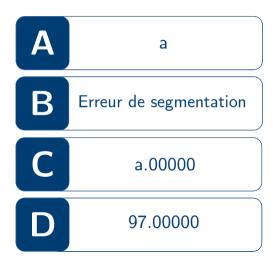
#QDLE#Q#ABCD\*#30#

```
int a = 3;
                                            3-7-10
int b = 7;
int r = ++a + ++b;
printf("%d-%d-%d", a, b, r);
                                            3-7-12
                                            4-8-10
                                            4-8-12
```

```
int main() {
    float y = 'a';
    printf("%f", y);
    return 0;
}
```

```
Erreur de segmentation
       a.00000
      97.00000
```

```
int main() {
   float y = 'a';
   printf("%f", y);
   return 0;
```



```
int main() {
  int a = 1;
  int c = 3;
  int b = a << c;
  printf("%d\n", b);
  return 0:
                                       -724794438
                                    Erreur de segmentation
```

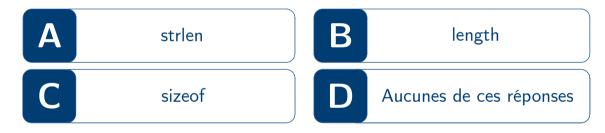
#QDLE#Q#A\*BCD#35#

```
int main() {
  int a = 1;
  int c = 3;
  int b = a << c;
  printf("%d\n", b);
  return 0:
                                       -724794438
                                    Erreur de segmentation
```

# Quelle est la fonction qui permet de connaître la taille d'un tableau générique d'entiers ?



# Quelle est la fonction qui permet de connaître la taille d'un tableau générique d'entiers ?



### Prenons un tableau d'entiers *tab* de taille *I* non nulle. Que renvoie cette fonction ?

```
Elle renvoie l'indice de la
int f(int tab[], int l, int val){
                                                     première occurrence de val dans
  for (int i=0: i<1: i++) {
                                                             tab, sinon -1
      if (tab[i]==val)
          printf("Trouvé !\n");
                                                         Elle renvoie toujours 0
          return i:
  return -1:
                                                         Elle renvoie toujours -1
```

Erreur de segmentation

#QDLE#Q#AB\*CD#70#

### Prenons un tableau d'entiers *tab* de taille *I* non nulle.

### Que renvoie cette fonction?

```
Elle renvoie l'indice de la
int f(int tab[], int l, int val){
                                                      première occurrence de val dans
  for (int i=0; i<1; i++) {
                                                              tab, sinon -1
      if (tab[i]==val)
          printf("Trouvé !\n");
                                                          Elle renvoie toujours 0
          return i:
  return -1:
                                                          Elle renvoie toujours -1
                                                          Erreur de segmentation
```

Chaînes de caractères

#### Laquelle de ces déclarations est incorrecte ?

A char[] str = "Hello world!";

B char \*str = "Hello world!";

C char str[25] = "Hello world!";

C char str[25] = "Hello world!";

#### Laquelle de ces déclarations est incorrecte ?

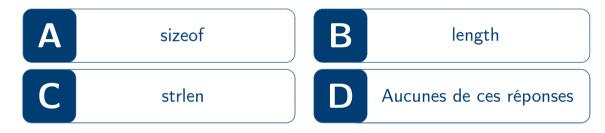
char[] str = "Hello world!";

char str[25] = "Hello world!";

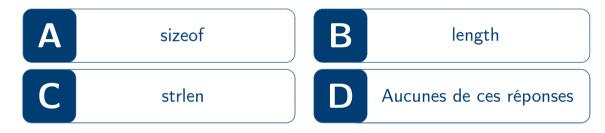
B char \*str = "Hello world!";

char str[] = "Hello world!";

# Quelle est la fonction qui permet de connaître la taille d'une chaîne de caractères ?



# Quelle est la fonction qui permet de connaître la taille d'une chaîne de caractères ?



```
int test(int c) {
  printf("c\n");
  return c:
int main() {
  test(2+3);
  return 0;
```

5 Erreur de segmentation Erreur à la compilation

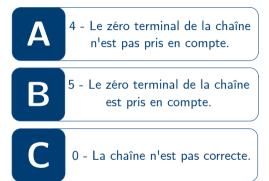
#QDLE#Q#AB\*CD#45#

5

```
int test(int c) {
  printf("c\n");
  return c:
int main() {
  test(2+3);
                                         Erreur de segmentation
  return 0;
                                         Erreur à la compilation
```

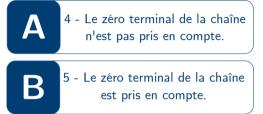
#### Quelle est la valeur de la variable len?

```
int len = strlen("toto");
```



#### Quelle est la valeur de la variable len?

```
int len = strlen("toto");
```



0 - La chaîne n'est pas correcte.

```
int main() {
  char s[] = \{'t', 'o', '\setminus 0', 't', 'o', '\setminus 0'\};
                                                                            toto
  printf("%s\n", s);
  return 0;
                                                                          to\0to
                                                                             to
                                                                 Erreur à la compilation
```

#QDLE#Q#ABC\*D#45#

```
int main() {
  char s[] = \{'t', 'o', '\setminus 0', 't', 'o', '\setminus 0'\};
                                                                           toto
  printf("%s\n", s);
  return 0;
                                                                          to\0to
                                                                             to
                                                                Erreur à la compilation
```

```
int main () {
                                                toto
   char* toto = "toto":
   toto[2] = 97;
   printf("%s\n", toto);
                                               to97to
   return 0:
                                                toao
                                         Erreur de segmentation
```

#QDLE#Q#ABCD\*#45#

```
int main () {
                                              toto
   char* toto = "toto":
   toto[2] = 97;
   printf("%s\n", toto);
                                             to97to
   return 0:
                                              toao
```

Erreur de segmentation

### Que fait cette fonction pour une chaîne de caractères a non vide ?

```
int func(char* a) {
                                            Une erreur de
  char*b=a:
                                            segmentation
  while(*++a)
                                         Une différence entre
  return a-b;
```

C Elle calcule la taille de la chaîne de caractères a

Elle renvoie un pointeur sur le début de la chaîne a

### Que fait cette fonction pour une chaîne de caractères a non vide ?

```
int func(char* a) {
 char*b = a:
 while(*++a)
  return a-b;
```

Une erreur de segmentation Une différence entre caractères Elle calcule la taille de la chaîne de caractères a Elle renvoie un pointeur sur le début de la chaîne a

# À quoi correspond la fonction streat proposée par <string.h> ?

Elle permet d'imprimer un emoji en forme de chat dans la console

Elle permet de concaténer le contenu d'une chaîne de caractères à une autre

Elle découpe une chaîne en souschaînes selon un caractère de séparation

Elle n'existe pas en réalité

# À quoi correspond la fonction streat proposée par <string.h> ?

Elle permet d'imprimer un emoji en forme de chat dans la console

Elle découpe une chaîne en souschaînes selon un caractère de séparation Elle permet de concaténer le contenu d'une chaîne de caractères à une autre

Elle n'existe pas en réalité

```
int main(){
   int a = 1:
   if ((char)a == '1')
     printf("%d\n", a);
   else
     printf("FAIL\n");
   return 0;
```

```
Erreur à la compilation
 La valeur ASCII de 1
         FAIL
```

```
int main(){
                                        Erreur à la compilation
   int a = 1:
   if ((char)a == '1')
      printf("%d\n", a);
   else
     printf("FAIL\n");
   return 0;
                                         La valeur ASCII de 1
                                                FAIL
```

### **Pointeurs**

### À quoi correspond la ligne de code suivante ?

int\* a;

A Elle multiplie les contenus des variables int et a

type pointeur sur entier.

Elle déclare une variable (opérateur \*) a de type int.

Elle déclare un pointeur générique sur 4 octets

### À quoi correspond la ligne de code suivante ?

int\* a;

Elle multiplie les contenus des variables int et a

B Elle déclare une variable a de type pointeur sur entier.

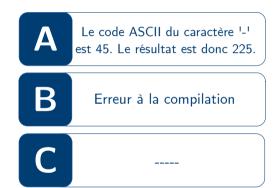
Elle déclare une variable (opérateur \*) a de type int.

Elle déclare un pointeur générique sur 4 octets

```
char* result = "-" * 5;
printf("%s\n", result);
```

```
Le code ASCII du caractère '-'
est 45. Le résultat est donc 225.
   Erreur à la compilation
```

```
char* result = "-" * 5;
printf("%s\n", result);
```



### Qu'affiche ce code sur un système classique **64bits** ?

```
int main() {
                                                   T(p) = 4
  int x = 20000:
                                                   T(q) = 4
  double y = 26;
  int* p = &x:
                                                   T(p) = 4
  double* q = &y;
                                                   T(q) = 8
  printf("T(p)=%d\n", sizeof(p));
```

printf("T(g)=%d\n", sizeof(g));

T(p) = 8T(a) = 8

Erreur de segmentation

return 0;

# Qu'affiche ce code sur un système classique 64bits ?

```
int main() {
                                                    T(p) = 4
  int x = 20000:
                                                    T(q) = 4
  double y = 26;
  int* p = &x:
                                                    T(p) = 4
  double* q = &v:
                                                    T(q) = 8
  printf("T(p)=%d\n", sizeof(p));
  printf("T(g)=%d\n", sizeof(g));
                                                    T(p) = 8
  return 0;
                                                     T(a) = 8
                                               Erreur de segmentation
```

```
int main() {
                                               1 1
  char* str = "v";
  char tab[1]:
  tab[0] = 'y';
                                               1 2
  printf("%d ",strlen(str));
  printf("%d" ,strlen(tab));
  return 0:
                                               22
                                           1 valeur indéfinie
```

#QDLE#Q#ABCD\*#70#

```
int main() {
  char* str = "v";
  char tab[1]:
  tab[0] = 'y';
                                              1 2
  printf("%d ",strlen(str));
  printf("%d" ,strlen(tab));
  return 0:
                                              22
                                          1 valeur indéfinie
```

### Que risque très certainement d'afficher le code suivant?

```
int main(){
   int a:
   int* ptr a;
   *ptr a = 1;
                                            valeur indéfinie
   printf("%d\n", a);
```

Erreur à la compilation Erreur de segmentation

# Que risque très certainement d'afficher le code suivant ?

```
int main(){
   int a:
   int* ptr a;
   *ptr a = 1;
                                            valeur indéfinie
   printf("%d\n", a);
```

Erreur à la compilation

Erreur de segmentation