

### EXERCICE 1)

```
1) float f = 10.0;
2) float *pf;
3) printf("Valeur de f : %.1f\n", f);
4) pf=&f;
5) printf("Valeur de pf : %.1f\n", *pf);
6) *pf = 999.5;
7) printf("Valeur de pf après modification : %.1f\n", *pf);
   printf("Valeur de f après modification : %.1f\n", f);
printf
```

### CODE COMPLET :

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

```
int main() {
```

```
    float f = 10.0;
    float *pf;
    printf("Valeur de f : %.1f\n", f);
    pf=&f;
    printf("Valeur de pf : %.1f\n", *pf);
    *pf = 999.5;
    printf("Valeur de pf après modification : %.1f\n", *pf);
    printf("Valeur de f après modification : %.1f\n", f);
```

```
}
```

### TRACE D'EXECUTION :

### EXERCICE 2)

```
1) float Min3Float(float a, float b, float c) {
```

```
    if(a < b && a < c) {
        return a;
    } else if(b < a && b < c) {
        return b;
    } else {
        return c;
    }
}
```

```
}
```

```
2) a) float x1;
```

```
    float x2;
```

```
    float x3;
```

```
    float res;
```

```
    b) printf("Nombre 1 : \n");
```

```
    scanf("%f", &x1);
```

```
    printf("Nombre 2 : \n");
```

```

scanf("%f", &x2);

printf("Nombre 3 : \n");
scanf("%f", &x3);
c) res = Min3Float(x1,x2,x3);
printf("Valeur de res : %f", res);
CODE COMPLET : #include <stdio.h>

float Min3Float(float a, float b, float c) {

```

```

    if(a < b && a < c) {
        return a;
    } else if(b < a && b < c) {
        return b;
    } else {
        return c;
    }
}

int main() {
    float x1;
    float x2;
    float x3;
    float res;

    printf("Nombre 1 : \n");
    scanf("%f", &x1);

    printf("Nombre 2 : \n");
    scanf("%f", &x2);

    printf("Nombre 3 : \n");
    scanf("%f", &x3);

    res = Min3Float(x1,x2,x3);
    printf("Valeur de res : %f", res);
}

```

TRACE D'EXECUTION :

### EXERCICE 3)

- 1) a) printf("AVANT PERMUTATION : \nValeur de a : %d\nValeur de b : %d\n\n",a,b);
- b) int temp;
- temp=a;
- a=b;
- b=temp;
- c) printf("APRES PERMUTATION : \nValeur de a : %d\nValeur de b : %d\n\n",a,b);

2) a) int x = 10;  
 int y = 999;  
 b) printf("x=%d et y=%d\n\n", x,y);  
 c) int res = permute(x,y);  
 d) printf("x=%d et y=%d\n\n", x,y);

CODE COMPLET : #include <stdio.h>

```
int permute(int a, int b) {
    printf("AVANT PERMUTATION : \nValeur de a : %d\nValeur de b : %d\n\n",a,b);
    int temp;
    temp=a;
    a=b;
    b=temp;
    printf("APRES PERMUTATION : \nValeur de a : %d\nValeur de b : %d\n\n",a,b);
}
```

```
int main() {
    int x = 10;
    int y = 999;
    printf("x=%d et y=%d\n\n", x,y);
    int res = permute(x,y);
    printf("x=%d et y=%d\n\n", x,y);
}
```

TRACE D'EXECUTION :

EXERCICE 4)

1)2)3)4)5)6)

#include <stdio.h>

```
int main() {
    unsigned int x;
    unsigned int masque;
    unsigned int res1;
    unsigned int res2;
    unsigned int res3;

    x = 0x0E2C;

    printf("La valeur en décimal de 0x0E2C est : %d\n", x);
    printf("La valeur en binaire de 0x0E2C est : ");
    for (int i = sizeof(x) * 8 - 1; i >= 0; i--) {
        printf("%d", (x >> i) & 1);
    }
}
```

```

    }
    printf("\n");
    masque = 0b1111111100000000;
    res1 = x & masque;
    printf("Le résultat en décimal est : %d\n", res1);

    printf("Le résultat en hexadécimal est : %x\n", res1);

    res2 = res1 >> 8;

    printf("Les 8 bits de poids fort de resi en décimal sont : %d\n", res2);

    printf("Les 8 bits de poids fort de resi en hexadécimal sont : %x\n", res2);
    x = 1600;
    res3 = x >> 2;

    printf("Le résultat du décalage à droite de 2 bits de x est : %d\n", res3);

    printf("Vérification : %d / 4 = %d\n", x, x / 4);

}

7) #include <stdio.h>

int main() {
    unsigned short var = 35;
    unsigned short masque = 0b1000000000000000;
    for(int i=0;i<16;i++) {
        if(var & masque) {
            printf("1");
        } else {
            printf("0");
        }
        masque = masque >> 1;
    }
}

```

TRACE D'EXECUTION :

## EXERCICE 5)

CODE COMPLET :

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

```
#define SIZE 100
```

```
int main() {
    int tab[SIZE];
    for (int i = 0; i < SIZE; i++) {
        tab[i] = i;
    }

    printf("Contenu initial du tableau :\n");
    for (int i = 0; i < SIZE; i++) {
        printf("tab[%d] = %d\n", i, tab[i]);
    }

    for (int i = 0; i < SIZE; i += 2) {
        tab[i] = 0;
    }

    printf("\nContenu après mise à zéro des éléments multiples de 2 :\n");
    for (int i = 0; i < SIZE; i++) {
        printf("tab[%d] = %d\n", i, tab[i]);
    }

    for (int i = 0; i < SIZE; i += 3) {
        if (tab[i] % 3 == 0) {
            tab[i] = 0;
        }
    }

    printf("\nContenu après mise à zéro des éléments multiples de 3 :\n");
    for (int i = 0; i < SIZE; i++) {
        printf("tab[%d] = %d\n", i, tab[i]);
    }

    for (int i = 0; i < SIZE; i += 5) {
        if (tab[i] % 5 == 0) {
            tab[i] = 0;
        }
    }

    printf("\nContenu après mise à zéro des éléments multiples de 5 :\n");
    for (int i = 0; i < SIZE; i++) {
        printf("tab[%d] = %d\n", i, tab[i]);
    }
}
```

```
    }  
    return 0;  
}
```

BONUS :

```
void fonctionBonus(int tab[], int step) {  
    for (int i = 0; i < sizeof(tab)/sizeof(tab[0]); i += step) {  
        tab[i] = 0;  
    }  
}
```