Devoir de SR01

Exercice1

Pour chacun de ces programmes, exécuter le programme et donner une explication du resultat obtenu:

programme1:

```
# include < stdio .h>
int main (){
int A=20 ,
B =5;
int C=!-- A /++! B;
printf (" A=% d B=% d c=% d \ n", A,B, C);
}
```

Résultat obtenu: A=19 B=5 C=0

L'expression '--A' décrémente A avant l'opération, et '++!B' incrémente la négation de B (qui est 0). La division de 0 par 1 donne 0, le résultat est stocké dans C.

programme2:

```
# include < stdio .h>
int main (){
int A=20 ,
B=5 ,
C= -10 ,
D =2;
printf ("% d \ n", A&& B ||! 0&& C ++&&! D ++) ;
printf (" c=% d d=% d \ n", C, D);
}
```

Résultat obtenu: 1

-9 2

L'expression 'A&&B' retourne 1, '!0 && C++' est équivalente à '1 && C&&!D puis C++, !D++' retourne 0 avant d'incrémenté C et de la négationde D. Lerésulatat final est 1 car && estprioritair sur ||. on affiche lesvaleur de C et D.

programme3:

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

```
int p [4]={1 , -2 ,3 ,4};
int * q=p;
printf (" c=% d\ n", *++ q** q ++) ;
printf (" c=% d \ n" ,* q);
}
```

Résultat obtenu:-2
3

L'expression '*++q**q++' signifie (*(++q) * *(q++)), est équivalente à 'q[1]*q[0] puis *q = *q[2]', q pointe maintenant sur le troisième élément on affiche q[2].

programme4:

```
# include < stdio .h>
int main (){
int p [4]={1 , -2 ,3 ,4};
int * q=p;
int d=* q&* q ++|* q ++;
printf (" d=% d\ n", d);
printf (" q=% d \ n" ,* q);
}
```

Résultat obtenu: d =
q = 3

L'expression '*q & *q ++ | *q++' signifie ' *q & *(q++) | *(q++). L'utilisation de l'opérateur de post-incrémentation plusieurs fois dans la meme instruction n'est pas conseillé et embigue, cependant comme & est prioritair sur | alors on a, '1 & 1' retourne 1 puis '1 | -2' bit à bit retourne -1, q pointe sur 3.

programme5:

```
# include < stdio .h>
int main (){
int a=-8 , b =3;
int c= ++a&&-- b ? b --: a ++;
printf (" a=% d b=% d c=% d\ n",a,b, c);
}
```

Résultat obtenu: -7 1 1

a++ && - -b est vrai donc B– est exécuté et A++ non.

programme6:

```
# include < stdio .h>
int main (){
int
a=-8 , b =3;
a
>
```

```
>=2^b;
printf (" a=% d\ n",a);
}
```

Résultat obtenu: -8

‘> >=’ n’est pas très conventionnelle.