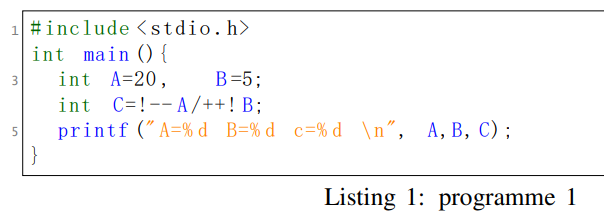
SR01 – Devoir 1

# **Exercice 1**

# Programme 1



Le programme n’a pas l’air de fonctionnner.  
mais basiquement je dirais qu’on fait

-Mettre A à 20 et B à 5

-Faire –A donc renvoie 19 et A = 19

- 19 est un valeur différente de 0 donc considérée comme true

-Donc son inverse !19 revient à !true donc à false

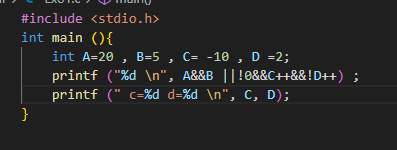
-Dans la même logique !18 donne false donc 0

-Donc ++ !B donne 1 mais n’augmente pas B car il augmente seulement l’inverse de B

-On fait donc 0/1 donc 0 donc C = 0

Donc A = 19, B = 5 et C = 0

# Programme 2



*On obtient :*

*1*

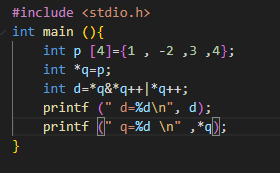
*c=-10 d=2*

Ce qui est logique car :

-On met A à 20, B à 5, C=-10, D=2

On fait A&&B or A et B sont à true car différents de 0 donc ca renvoie true donc vu qu’après c’est un || (OR), pas besoin de regarder ce que ca donne , ca donnera forcément true donc 1 (Ce qu’on voit bien). Le compilateur, dans un soucis de performance va donc arreter de lire ce qu’il y a après le || et c’est donc pour ca que le C++ et le !D++ ne sont pas executés

# Programme 3



*On obtient :*

*d=-1*

*q=3*

Voici pourquoi :

-On créer une liste p = [1, -2, 3, 4]

-On créer un pointer q vers p

-On va commencer à calculer d :

-\*q&\*q++ donne donc \*q&\*q, or un (ET) binaire sur sois meme reviens à sois meme donc \*q

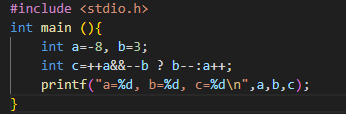
-On a donc \*q|\*(q+1) donc 1|-2 or :

- 1 en binaire = 0001

- -2 en binaire = 1110 donc avec l’opérateur | (OU) on a 1111 donc -1

Pour ce qui est de q , on l’a postincrémenté deux fois donc \*q = \*(qDuDebut +2) = p[2] = 3

# Programme 4



*On obtient :*

*a=-7, b=1, c=2*

On va d’abord chercher à savoir le résultat de la condition :

-On a a qui est préincrémenté et b qui est présoustrait. Donc 7&&2 ce qui donne donc 1&&1 donc 1 donc on effectue B—mais pas a++

Donc

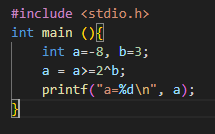
- La variable a a juste une préincrémentation donc a=-8+1=-7

-La variable b a une présoustraction et une postsoustraction donc b=3-1-1=1

-C vaut donc b— (après execution du --b) donc c=b(du départ)-1=3-1=2

# Programme 6

Ce programme ci non plus ne fonctionne pas  
je l’ai donc adapté à ma sauce



*On obtiens :*

*a=3*

Ceci s’explique assez facilement :

A=-8 or -8 >= 2 donne false donc 0

0^b = 0XOR3= 00XOR11 = 11 = 3

# **Exercice 2**