Un pointeur est une variable (un emplacement mémoire) d'un type spécial qui pourra contenir l'adresse d'une autre variable.

La syntaxe est la suivante: type \*identificateur;

**Exemple:** déclaration d'un pointeur vers un entier:

 int\* p; **ou encore:** int \*p;

* On accède à la valeur stockée à l'adresse contenue dans un pointeur grâce à l'opérateur unaire dit de référencement ou d'indirection: \*

**Illustration d'un exemple: (colonne 2 est la solution que doit donner l’etudiant)**

|  |  |
| --- | --- |
|  int x=10; | **&x=62 et x=10** |
|  int\* px;  |  |
|  px=&x;  | On écrit l'adresse de x à l'emplacement réservé pour le pointeur px.  |
|  printf("%d",\*px); | On affiche la valeur de x=10  |
|  \*px=20; | On obtient x = 20 |

1. Un pointeur peut pointer vers un autre pointeur.
2. On peut aussi incrémenter un pointeur et augmenter sa valeur de la taille de l'objet (voir exercice 2);
3. Un pointeur peut pointer vers un tableau

 Soit la déclaration: int Tab[10]={5,8,4,3,9,6,5,4,3,8};

* le nom d'un tableau est un pointeur sur son premier élément. On peut déclarer un pointeur et l'initialiser avec le nom du tableau.

 **int \*pTab;**

 **pTab = Tab;**

|  |  |
| --- | --- |
| printf("%d",\*Tab);  | Affiche **5**  |
| printf("%d",pTab[0]); | **Affiche 5.** |
| pTab[0]++;printf("%d",Tab[0]); | **Affiche 6.** |

**Exercice 1:** Dites ce qu'on réserve en mémoire:

|  |  |
| --- | --- |
| **Déclaration** | **Réservation en mémoire (solution)** |
| int t1[3];  | 3 entiers |
| int \*p1;  | 1 pointeur d'entiers |
| int \*\*p2;  | un pointeur de pointeur d'entiers. |
| int \*t3[3];  | 3 pointeurs d' entiers. |
| int (\*p3)[3];  | un pointeur de tableau de 3 entiers. |

1. Il est conseillé d'initialiser tout pointeur avant son utilisation effective avec la valeur NULL (constante prédéfinie qui vaut 0) ce qui, par convention, indiqué que le pointeur ne pointe sur rien.

**Exercice 2:**

Remplir le tableau ci-dessus et expliquer les valeurs prises par les différentes variables et pointeurs utilisés dans ce programme après chaque printf.

.

#include <stdio.h>

#include <stdlib.h>

**main()**

{system("color 3F");

**int A = 10;**

printf("\n adresse de la variable A = **%p**\n", &A); /\*%p: hexadecimal\*/

**int \* ptrA = &A;**

printf(" \n\t contenu de la variable A= %d son adresse**=%d** " ,A, **ptrA**);

printf("\n\t contenu de \*ptrA=%d",**\*ptrA**);

**int B=20;**

**int \* ptrB = &B;**

printf(" \n\t contenu variable B= %d son adresse**=%p** ",B , **ptrB**);

**\*ptrA = 15**;

printf(" \n\t contenu variable A= %d son adresse=%p \n",A**, ptrA**);

system("pause");

return 0;}

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **A** | **ptrA**  | **\*ptrA** | **B** | **ptrB** | **\*ptrB** |
|  **Initialisation** |  |  |  |  |  |  |
| printf(" ADRESSE A = **%p**\n", &A); |  |  |  |  |  |  |
| printf(" \n\t variable A= %d adresse**=%p** " ,A, **ptrA**); |  |  |  |  |  |  |
| printf("\n\t contenu de \*ptrA=%d",\*ptrA); |  |  |  |  |  |  |
| printf(" \n\t variable B= %d adresse**=%p** ",B , **ptrB**); |  |  |  |  |  |  |
| printf(" \n\t variable A= %d adresse=%p \n",A**, ptrA**); |  |  |  |  |  |  |

**Exercice 3 :**

Soit P un **pointeur** qui 'pointe' sur un tableau A:

**int A[] = {12, 23, 34, 45, 56, 67, 78, 89, 90};**

**int \*P;**

**P = A;**

Quelles valeurs ou adresses fournissent ces expressions (ne regardez pas la solution)

|  |  |  |
| --- | --- | --- |
| **a)  \*P+2** |  | **14** |
| **b)  \*(P+2)** |  | **34** |
| **c)   &P+1** |  | l'adresse du pointeur derrière le pointeur P  |
| **d)   &A[4]-3** |  | l'adresse de la composante A[1] |
| **e)    A+3** |  | l'adresse de la composante A[3]  |
| **f)     &A[7]-P** |  | la valeur (indice) 7 |
| **g)    P+(\*P-10)** |  | l'adresse de la composante A[2] |
| **h)    \*(P+\*(P+8)-A[7])** |  | la valeur 23  |