Nom : _____

Prénom : _____

Numéro étudiant : | | | | | | | | |

Les pointeurs

7 mars 2018

A lire absolument :

1. L'objectif n'est pas d'apprendre la correction par cœur, mais de comprendre les mécanismes mis en œuvre. Cela vous permettra de vous adapter face à un problème nouveau.
2. En particulier, vous devez être capable de refaire l'intégralité du sujet, seul, sans aucune aide ni support.
3. Votre travail sera corrigé automatiquement par l'outil de correction automatique CAT. Cela implique que vous devez respecter scrupuleusement les consignes de chaque exercice. Faites très attention aux messages qu'il vous est demandé d'afficher. Un espace en trop, un saut de ligne en moins et l'exercice risque d'échouer.
4. L'enseignant voit votre activité sur le site, ainsi que l'historique de vos dépôts. Pensez à déposer votre travail régulièrement afin qu'il puisse vous apporter des conseils personnalisés.
5. Si l'enseignant vous demande de rendre votre travail sur papier, vous devez répondre directement sur le sujet en respectant absolument la zone prévue à cet effet. Tout ce qui se trouve en dehors de la zone sera ignoré.
6. Si le sujet contient un QCM, vous devez colorier les cases avec un stylo bleu ou noir. Les autres couleurs seront ignorées.
7. Chaque feuille est identifiée de manière unique. Vous pouvez donc rendre votre sujet avec les feuilles mélangées, mais il est préférable de les trier car cela vous permet de vérifier que vous n'en avez pas oublié une.
8. Si vous faites face à un problème, un bug, une erreur ou que vous souhaitez participer à l'amélioration de la plateforme, envoyez un mail à l'adresse suivante : support-cat@liste.parisnanterre.fr

Ne rien écrire dans cette zone

La très grande majorité de ce TD est à réaliser avec un papier et un stylo. En effet, l'objectif n'est pas uniquement de constater ce qui se passe dans la mémoire, mais bien de comprendre et d'anticiper ce qui se passe.

Questions de cours sur les pointeurs 1

Remplissez les trous :

1. La mémoire est une succession d'_____.
2. Un octet contient _____ bits. Chaque bit peut valoir ___ ou ___.
3. Chaque octet possède une _____.
4. Le premier octet de la mémoire à pour adresse ___.
5. Lors de la _____ d'une variable un espace mémoire est réservé (alloué) automatiquement.
6. La taille de cet espace dépends de son _____. Cette taille, en _____, est donnée par l'opérateur _____.

Questions de cours sur les pointeurs 2

Recopiez le code suivant dans votre compilateur et executez le. Que constatez vous ?

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

int main(void)
{
    printf("sizeof(void)_________=_%d\n", sizeof(void));
    printf("sizeof(char)_________=_%d\n", sizeof(char));
    printf("sizeof(short)________=_%d\n", sizeof(short));
    printf("sizeof(int)__________=_%d\n", sizeof(int));
    printf("sizeof(long)_________=_%d\n", sizeof(long));
    printf("sizeof(long_long)____=_%d\n", sizeof(long long));
    printf("sizeof(float)________=_%d\n", sizeof(float));
    printf("sizeof(double)_______=_%d\n", sizeof(double));
    return EXIT_SUCCESS;
}
```

Questions de cours sur les pointeurs 3

Remplissez les trous :

1. Un pointeur est une variable qui contient l'_____ d'une zone dans la _____.
2. Un pointeur doit toujours être initialisé à la valeur _____.
3. La valeur d'un pointeur qui ne pointe vers aucune variable doit être _____.
4. L'opérateur & placé devant le nom d'une variable permet d'obtenir son _____.
5. L'adresse d'une variable correspond à l'adresse du premier de ses _____.
6. L'opérateur * placé devant le nom d'un pointeur permet d'obtenir la _____ située à l'adresse pointée par ce dernier.

Les pointeurs, séquence d'instructions 1

Dans cet exercice, vous ne devez pas recopier le code dans votre IDE et encore moins l'exécuter. Vous devez déterminer la solution par vous même. Un papier et un stylo étant l'idéal.

Considérons le code suivant :

```
int i , j , *ip ;
ip = &i ;
i = 22 ;
j = *ip ;
*ip = 17 ;
```

Que valent i et j à la fin de cette séquence d'instruction ?

Modification de valeurs passées en argument

```
1 void incremente_et_decremente(int * p, int * q);
```

Écrire une fonction qui accepte en argument deux pointeurs d'entiers. Incrémentez la valeurs pointée par le premier, et décrémentez la valeur pointée par le second.

Echange du contenu de deux variables

```
1 void swap(int * a, int * b);
```

Définir une fonction swap

Les pointeurs, séquence d'instructions 2

Dans cet exercice, vous ne devez pas recopier le code dans votre IDE et encore moins l'exécuter. Vous devez déterminer la solution par vous même. Un papier et un stylo étant l'idéal.

Considérons le code suivant :

```c
#include<stdio.h>

void main()
{
    int i = 4;
    int j = 10;
    int* p;
    int* q;
    p = &i;
    q = &j;
    printf("i = %d, j = %d, p = %d, q = %d\n", i, j, *p, *q);
    *p = *p + *q;
    printf("i = %d, j = %d, p = %d, q = %d\n", i, j, *p, *q);
    p = &j;
    printf("i = %d, j = %d, p = %d, q = %d\n", i, j, *p, *q);
    *q = *q + *p;
    printf("i = %d, j = %d, p = %d, q = %d\n", i, j, *p, *q);
    q = &i;
    printf("i = %d, j = %d, p = %d, q = %d\n", i, j, *p, *q);
    i = 4;
    printf("i = %d, j = %d, p = %d, q = %d\n", i, j, *p, *q);
    *q = *q + 1;
    printf("i = %d, j = %d, p = %d, q = %d\n", i, j, *p, *q);
}
```

Déterminez, sans l'exécuter, ce qu'affiche ce programme.

Ne rien écrire dans cette zone

Pointeurs et cases d'un tableau

Cet exercice n'est pas corrigé automatiquement. Il sert à vous faire comprendre le lien entre pointeur et tableau.

1. Déclarer trois pointeurs p, q, et r pointant respectivement vers des types char, int et double.
2. Écrire une boucle pour afficher les valeurs des adresses
   — p, $p+1 \ldots p+9$.
   — q, $q+1 \ldots q+9$.
   — r, $r+1 \ldots r+9$
3. De combien augmentent les adresses à chaque tour de boucle ?
4. Utilisez l'opérateur "sizeof" pour déterminer la taille en octets des types char, int et double. Qu'en concluez vous ?
5. Déclarer maintenant trois tableaux de taille 10, un tableau de char, un tableau d'int et un tableau de double. Initialiser les pointeurs p, q, et r à la première case du tableau correspondant, et parcourez les tableaux dans une boucle. A chaque tour de boucle :
   — Incrémenterez les pointeurs de 1.
   — Afficher les valeurs de chacun des 3 pointeurs.
   — Affichez l'adresse de la case courante de chaque tableau.
   Observez bien les résultats.

Les pointeurs, séquence d'instructions 3

Dans cet exercice, vous ne devez pas recopier le code dans votre IDE et encore moins l'exécuter. Vous devez déterminer la solution par vous même. Un papier et un stylo étant l'idéal.

Considérons le code suivant :

```
int tab[] = {5, 15, 34, 54, 14, 2, 52, 72};
int *p = &tab[1], *q = &tab[5];
```

1. Quelle est la valeur de $*(p+3)$?
2. Quelle est la valeur de $*(q-3)$?
3. Quelle est la valeur de $q-p$?
4. La condition $p < q$ est-elle vraie ou fausse ?
5. La condition $*p < *q$ est-elle vraie ou fausse ?

Ne rien écrire dans cette zone

Les pointeurs, séquence d'instructions 4

Dans cet exercice, vous ne devez pas recopier le code dans votre IDE et encore moins l'exécuter. Vous devez déterminer la solution par vous même. Un papier et un stylo étant l'idéal.

Considérons le code suivant :

```
1  #define N 10
2
3  int tab[N] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10};
4  int *p = &tab[0], *q = &tab[N-1], temp;
5  while (p < q)
6  {
7      temp = *p;
8      *(p++) = *q;
9      *(q--) = temp;
10 }
```

Déterminez quelles seront les valeurs des cases du tableau *tab* après l'exécution de ces lignes de code.

Les pointeurs, séquence d'instructions 5

Complétez le tableau en indiquant les valeurs des différentes variables au terme de chaque instruction du programme suivant (on peut aussi indiquer sur quoi pointent les pointeurs) :

Programme	a	b	c	p1	*p1	p2	*p2
$inta, b, c, *p1, *p2;$							
$a = 1, b = 2, c = 3;$							
$p1 = \&a, p2 = \&c;$							
$*p1 = (*p2) + +;$							
$p1 = p2;$							
$p2 = \&b;$							
$*p1 - = *p2;$							
$+ + *p2;$							
$*p1 * = *p2;$							
$a = + + *p2 * *p1;$							
$p1 = \&a;$							
$*p2 = *p1/ = *p2;$							

Ne rien écrire dans cette zone

Pointeurs. Réécriture 1.

Réécrivez les opérations suivantes sans utiliser les opérateurs & et * :

1. *tab=0 ;
2. *(tab+5)=10 ;
3. pt = &tab[0]
4. a=*(&toto) ;
5. a=*(&(*(&toto))) ;
6. *tab=*(&(*(tab+5))) ;

Pointeurs. Réécriture 2.

Réécrivez les opérations suivantes sans utiliser le sucre syntaxique :

1. tab[0]=0 ;
2. tab[4]=tab[3] ;
3. tab[tab[0]]=tab[1]*tab[2] ;

Pointeurs. Fonction mystère

Déterminez, à la main, ce que fait la fonction mystere, et ce qui est affiché par le programme suivant :

```c
#include <stdio.h>

void mystere(int *a, int *b)
{
    *a = *a - *b;
    *b = *a + *b;
    *a = *b - *a;
}
int main()
{
    int e1=5,e2=7;
    mystere(&e2, &e1);
    printf("%d %d",e1,e2);
    return 0;
}
```

Ne rien écrire dans cette zone