Larbi Ben M'hidi -Oum El Bouaghi- University

Faculty of Exact Sciences and Natural and Life Sciences

Mathematics and Computer Science Department

Academic year: 2023-2024

Level: 1st year "Computer Science & Mathematics"

Module: Algorithmic and Data Structures 2

TP n°4
"Solution"

Pedagogic objective

- Manipulate pointers in C, pointer arithmetic and addressing the elements of an array;
- Apply the "malloc" and "free" functions for dynamic memory allocation and release.

Exercise n°1

```
#include<stdio.h>
main()
{
    int a,b,s;
    int *pa,*pb,*ps;
    pa = &a;
    pb = &b;
    ps = &s;
    printf("Give two integers:\n");
    scanf("%d%d",pa,pb);
    *ps = *pa + *pb;
    printf("The sum is %d.\n",s);
}
```

Exercise n°2

1.

```
#include <stdio.h>
main()
{
printf("The size of a character is %d byte(s)", sizeof(char));
printf("\n"); printf("The size of an integer short is %d
byte(s)", sizeof(short));
printf("\n"); printf("The size of a long integer is %d
byte(s)", sizeof(long));
printf("\n"); printf("The size of a real number is %d byte(s)",
sizeof(float));
printf("\n"); printf(" The size of the character string %s is
%d byte (s)","1234567",sizeof("1234567"));}
```

2. For the string: the number of characters in the string is 7, however the number of bytes to reserve is 8. The last byte is reserved for *the end of string character "\0".*

Exercise n°3

1.

```
#include <stdio.h>
main(){
int x,y; char a,b ; float f ;
int *pi1, *pi2;
char *pc1, *pc2 ;
float *pf1, *pf2;
x=10; y = 9; a='K'; b='S'; f=1.5;
pi1=& x; pi2=& y; pc1=& a; pc2=& b ;   pf1=& f; pf2=pf1 ;
printf(" Addresses in hexadecimal: pi1 = %X pi2 = %X \n", pi1,
pi2);
printf(" Addresses in decimal: pi1 = %d pi2 = %d \n", pi1, pi2);
printf("pc1= %d pc2= %d \n", pc1, pc2);
printf("pf1= %d pf2= %d \n", pf1, pf2);
// Increment
printf(" pi1 +1 = %d \n", pi1++);
printf(" pi2 +1 = %d \n", pi2++);
printf(" pc1 +1 = %d \n", pc1++);
printf(" pc2 +1 = %d \n", pc2++);
printf(" pif +1 = %d \n", pf1++);
printf(" pf2 +1 = %d \n", pf2++);
// Decrement
printf(" pi1 -1 = %d \n", pi1--);
printf(" pi2 -1 = %d \n", pi2--);
printf(" pc1 -1 = %d \n", pc1--);
printf(" pc2 -1 = %d \n", pc2--);
printf(" pif -1 = %d \n", pf1--);
printf(" pf2 -1 = %d \n", pf2--);
// Difference
printf(" pi1 - pi2 = %d \n", pi1 - pi2);
printf("pc1 - pc2= %d \n", pc1 - pc2);
printf("pf1 - pf2= %d \n", pf1 - pf2);
}
```

2. *Sum or addition is not allowed*

Adding an integer to a pointer. The result is a pointer of the same type as the starting pointer;
Subtracting an integer from a pointer. The result is a pointer of the same type as the starting pointer;
The difference of two pointers both pointing to objects of the same type. The result is an integer.
The sum of two pointers is not allowed.

Exercise n°4

```
#include <stdio.h>
#include <stdlib.h>
int main()
{
int *A,*B; int n,m,i;
printf("Give the dimension of A:");
scanf("%d",&n);
printf("Give the dimension of B:");
scanf("%d",&m);
A=(int*)malloc(n*sizeof(int));
for (i=0;i<n;i++)
{
printf("A[%d]=",i);
scanf("%d",&A[i]);
}
B=(int*)malloc(m*sizeof(int));
for (i=0;i<m;i++)
{
printf("B[%d]=",i);
scanf("%d",&B[i]);
}
A=(int*)realloc(A,(n+m)*sizeof(int));
for (i=n;i<n+m;i++)
A[i]=B[i-n];
printf("The array after fusion \n");
for (i=0;i<n+m;i++)
printf ("A[%d]=%d \n",i,A[i]);
free (A);
free (B);
return 0;
}
```

Exercise n°5

```
#include <stdio.h>
#include <stdlib.h>
main()
{
/* Declaration */
int *A;
int N,i;
int X;
int k=0;
int *P1;
printf("Give the dimension of the array : ");
scanf("%d", &N );
A=(int*)malloc(N*sizeof(int));
for (i=0; i<N; i++)
{
printf("A[%d] : ", i);
scanf("%d", &*(A+i));
}
printf("Introduce element X to eliminate from the array : ");
scanf("%d", &X );
P1=A;
for (i=0; i<N; i++)
if (A[i] != X)
{
P1[k] = A[i];
k++;
}
/* Display the result */
for (i=0; i<k; i++)
printf("%d ", A[i]);
printf("\n");
free (A);
}
```

<u>Exercise n°6</u>

```
#include <stdio.h>
#include <stdlib.h>
main()
{
/* Declaration */
int **U; /* unitary matrix  */
int N; /* dimension  */
int I, J;
printf("Dimension of the square matrix: ");
scanf("%d", &N);
/* Square matrix memory reservation */
U = (int**) malloc (N * sizeof(int*));
for (I = 0; I < N; I++)
U[I] = (int*) malloc (N * sizeof(int));
/* Construction of the unitary square matrix */
for (I=0; I<N; I++)
for (J=0; J<N; J++) if (I==J)
U[I][J]=1;
else
U[I][J]=0;
/* Display the result */
printf("Unitary dimension matrix %d :\n", N);
for (I=0; I<N; I++)
{
for (J=0; J<N; J++)
printf("%7d", U[I][J]); printf("\n");
}
for (I=0; I<N; I++)
free(U[I]); free(U);
}
```