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

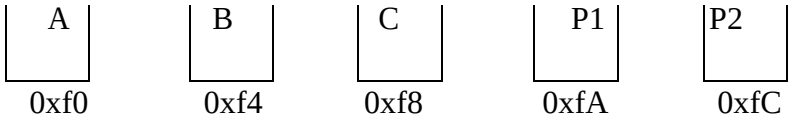
TD n°4

"Solution"

Pedagogic objective

- Understand the declaration, syntax and semantics of pointers, among others: indirect addressing, the content of a dynamic variable as well as arrays using pointers;
- Understand dynamic memory allocation and freeing.

Exercise n°1



	A	B	C	&A	&B	&C	P1	P2	*P1	*P2
A←1, B←2, C←3 P1←&A, P2←&B	1	2	3	0xf0	0xf4	0xf8	0xf0	0xf4	1	2
P2= &C	1	2	3	0xf0	0xf4	0xf8	0xf0	**0xf8**	1	**3**
***P1 = *P2**	**3**	2	3	0xf0	0xf4	0xf8	0xf0	0xf8	**3**	3
(*P2)++	3	2	**4**	0xf0	0xf4	0xf8	0xf0	0xf8	3	**4**
P1= P2	3	2	4	0xf0	0xf4	0xf8	**0xf8**	0xf8	**4**	4
P2= &B	3	2	4	0xf0	0xf4	0xf8	0xf8	**0xf4**	4	**2**
***P2 = *P1 - 2**P2**	3	**0**	4	0xf0	0xf4	0xf8	0xf8	0xf4	4	**0**
(*P2)--	3	**-1**	4	0xf0	0xf4	0xf8	0xf8	0xf4	4	**-1**
C = (P2 == &C)	3	-1	0	0xf0	0xf4	0xf8	0xf8	0xf4	**0**	-1
***P2= *P1 + A**	3	**3**	0	0xf0	0xf4	0xf8	0xf8	0xf4	0	**3**

Exercise n°2

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

a) *P+2 => the value 14

b) *(P+2) => the value 34

c) &P+1 => the address of the pointer behind the pointer P (rarely used)

d) &A[4]-3 => the address of element A[1]

e) A+3 => the address of element A[3]

f) &A[7]-P => the value (index) 7

g) P+(*P-10) => the address of element A[2]

h) *(P+*(P+8)-A[7]) => the value 23

Exercise n°3

```
Algorithm Exercise1 ;
Type Student  = Record
    N_Registration: integer;
    Last_name: string (20); First_name: string (20);
    Res: real;
EndRecord
Variables pi: * int; // pointer to an integer.
          pr:*real; // pointer to real.
          pc: * character; // pointer to character.
          pe: * Student; // pointer to a student record.
          i: int; r: real; c: character; e: Student;
Begin
pi ←&i ; pr ← &r ; pc ← &c ; pe ← &e ;
Write ("enter an integer"); Read (*pi);
Write ("enter a real"); Read (*pr);
Write ("enter a character"); Read (*pc);
Write ("enter student's information :");
Write ("enter the registration number"); Read ((*pe). N_Registration );
Write ("enter the name"); Read ((*pe). Last_name);
Write ("enter the first name"); Read ((*pe). First_name);
Write ("enter the result"); Read ((*pe).Res);
End
```

Exercise n°4

Algorithm invert_Array;

Variables tab = Array [1..50] integer;

N, HELP: integer;

P1: * integer; P2: *integer; /*helper pointers*/

Begin

/* Data entry */

Repeat

Write ("enter the dimension of the array: ");

Read (N);

Until (N>=1) and (N<=50);

For (P1=tab) to (tab+N) ***do***

// Equivalent to (For i=1 to N do)

// Just to show indirect (pointer) addressing manipulation

Read (*P1);

// we did not use the address operator &, however, we used the pointer P1 which contains an address to tab[i]; (indirect addressing)

EndFor

// Display the array

For (P1=tab) o (tab+N) ***do***

Write (*P1);

EndFor

//Reverse array elements

P1 ← tab; P2 ← tab+N;

While (P1<P2) ***do***

HELP ← *P1 ;

*P1 ← *P2;

*P2 ← HELP;

P1 ← P1+1;

P2 ← P2-1;

EndWhile

// Display result

```
For (P1=tab) to (tab+N) Do
Write (*P1);
EndFor
END
```

Exercise n°5

1. Data structures

```
Type Date=Record
    dd: integer;
    mm: integer;
    aa: integer;
EndRecord
Type Employer= Record
 Last_name: string(20);
 First_name: string (20);
 Date_B: Date;
 Nb_E: integer;
 Qualif: string (30);
EndRecord
Employer_list=Array [1…300] of * Employer;
```

2. Algorithm to enter data

```
Algorithm Employers;
Type Date=Record
    dd: integer;
    mm: integer;
    aa: integer;
EndRecord
Type Employer= Record
 Last_name: string(20);
 First_name: string (20);
 Date_B: Date;
 Nb_E: integer;
 Qualif: string (30);
```

```
EndRecord
Employer_list=Array [1..300] of *Employer; //Array of pointers to employers.
Variables i: integer, Bool: boolean; Char: character; L: Employer_list;
Begin
i ← 1 ; Bool ← true;
While (Bool = true) and (i<=300) do
 Allocate (L[i]);
// L[i] is a pointer to an employer, it is the memory allocation of a dynamic employer type
variable whose pointer is L[ i ].
Write (“"Employer n°", i);
Write ("Give the last name: ");
Read ((*L[ i ]).Last_name);
Write ("Give the first name: ");
Read ((*L[ i ]).First_name);
Write ("Give the date of birth: ");
Write ("Give the day: "); Read ((*L[ i ]). Date_B.dd);
Write ("Give the month: ");Read ((*L[ i ]).Date_B.mm);
Write ("Give the year: "); Read ((*L[ i ]). Date_B.yy);
Write ("Give the number of children: "); Read ((*L[ i ]).Nb_E);
Write ("Give the qualification: "); Read ((*L[ i ]).Qualif);
Repeat
Write (“are there other employers Y/yes/N/no?”);
Read (Char);
Until (Char = ‘Y’) or (Char = ‘N’)
If (Char = 'Y') then
    i ← i +1;
else
    Bool ← false;
Endwhile
END.
```

3. Algorithm for sorting employers

```
Algorithm Sort_Employers;
Employer_list=Array [1..300] of *Employer; //Array of pointers to employers.
Variables N, Pc: integer;
//  N number of employers entered in the previous algorithm =i
        L: Employer_list; P1, P2: *Employer; Aux: Employer;
Begin
For (P1=L)  to (L+N-1) do
    For (P2=P1+ 1) to (L+N) do
      If ((*P1). Date_B .yy) < ((*P2). Date_B .yy) then
         Aux ← (*P1);
        (*P1) ← (*P2);
        (*P2) ← Aux ;
       else
          If ((*P1).Date_B .yy)= ((*P2).Date_B .yy) then
              If ((*P1). Date_B .mm) < ((*P2). Date_B .mm) then
                  Aux ← (*P1);
                  (*P1) ← (*P2);
                  (*P2) ← Aux ;
               else
                 If ((*P1). Date_B .mm)= ((*P2). Date_B .mm) then
                      If ((*P1). Date_B .dd) < ((*P2). Date_B .dd) then
                          Aux ← (*P1);
                          (*P1) ← (*P2);
                          (*P2) ← Aux ;
                       End if
                    End if
                End if
            End if
        End if
  EndFor
EndFor
// Number of employers selected
```

```
Pc ← (N DIV 10);
P1 ← L;
For (P1 = L+Pc)  to (L+N) Do
    Free (P1) ;
EndFor
End
```