REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE
MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE LA RECHERCHE SCIENTIFIQUE

Université de M'sila
Faculté des Mathématiques et de l'Informatique
Département d'informatique

جامعة المسيلة
كلية الرياضيات والإعلام الآلي
قسم الإعلام الآلي

Chapter 5:
Arrays and strings

Algorithms and data structure 1

Presented by: Dr. Benazi Makhlouf
Academic year : 2023/2024

Contents of chapter 05:

1. Introduction

- In programming, data is organized in the form of constants and variables. There are different types of data, which can be divided into two parts:
    1. Simple types: Such as integers, floats, characters, and Booleans.
    2. Composite types: Arrays, and structures or records.
- Suppose we want to enter the grades of 1000 students. It would be unreasonable to use 1000 variables to store the grades and write 1000 input instructions (read) in the program.
- It is better to use a single variable that can hold all the grade values and use a loop to input them.
- The structure that can store multiple values at the same time is called an array.

2. Arrays

Array: A complex data structure, consisting of a finite set of homogeneous elements (of the same type), accessible through indexes indicating their location.

An array can be seen as a group of variables of the same type with the same name.

Index: When data is stored in an array, each element is identified by an index, which in C is a non-negative integer (≥ 0). The index starts from 0 to N - 1 (where N is the size of the array).

One-dimensional array (vector): In this type of array, we use a single index to access its elements. The array is represented in memory as a sequence of adjacent cells. Once the array is created, a new cell cannot be added or removed (static).

indice	0	1	2	3	4	5	6	7
valeur	15	7	-3	0	9	2	0	-3

Declaration

Algorithm

```
        array_name : array[Size] of type_Elements
```

In C language

```
        type_Elements array_name [Size] ;
```

Example

Algorithm

```
    const N=100
    Grade : array[N] of real
    arr : array[50] of integer
    arr2 : array[20] of integer
```

In C language

```
    const int N=100 ;
    float Grade[N] ;
    int arr[50], arr2[20];
```

Initialization

- In C, it is possible to specify initial values for all elements of the array within curly braces **{** and **}** when declaring the array. The values must be of the same type and separated by commas '**,**'.
- You can specify the number of elements between the **square brackets** ‘**[ ]**', or leave them **empty**, in this case the size will be calculated.

Example

```
int arr[] = {15, 7, -2, 0, 9, 2, 0, -3};
```

indice	0	1	2	3	4	5	6	7
valeur	15	7	2-	0	9	2	0	3-

Use

- The array cannot be treated as a single block like tab*10; instead, each element must be accessed separately. To access a single element of the array, we use the name of the array with an index inside **square brackets [** and **]**.

  Note: Attempting to access an element that does not exist (if the index is greater than or equal to the size of the array or negative) will cause the program to terminate.

Example:

```
arr[5-3]←arr[arr[3]+1]   ⇔   arr[2]←arr[0+1]   ⇔   arr[2]←7
```

Indice	0	1	2	3	4	5	6	7
valeur	15	7	7	0	9	2	0	-3

The reading of an array

```
read(tab[0])

read(tab[1])

...

read(tab[N-1])
```

Algorithm	C
for i←0 **to** N-1 **do**	**for**(i = 0; i < N; i++)**{**
write("nb", i, "⇒")	printf("nb %d ⇒", i);
read(tab[i])	scanf("%d", &tab[i]);
End For	**}**

The display of an array

Algorithm	C

`for i←0 to N-1 do`
` write(tab[i])`
`End For`
	`for(i = 0; i < N; i++){`
	` printf("%d\t", tab[i]);`
	`}`

Example

Write a program that receives the grades of **N** students, where N is determined by the user, and then calculates the number of students who have not passed the course. (Average below 10).

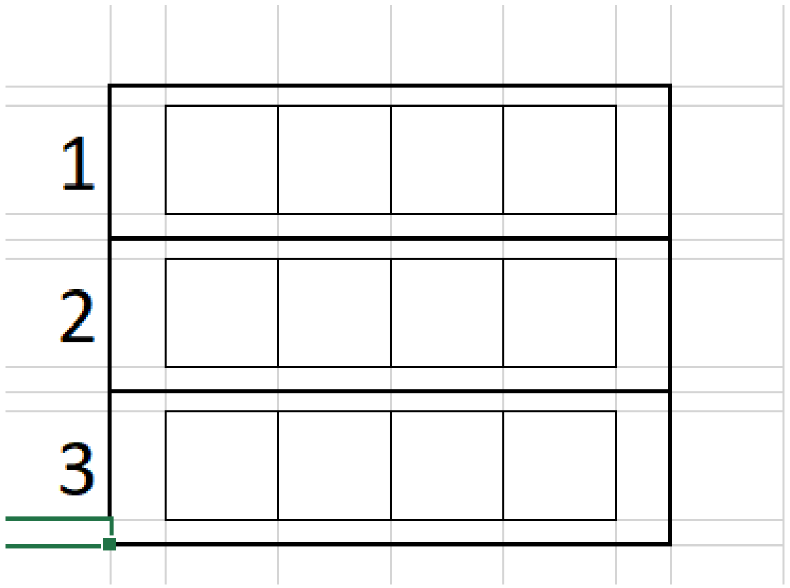
```
Algorithm nb_failed
Const MAX=200
var grade : array[MAX] of real
 i, aj, N : integer
begin
do
 write(" enter number of students (<", MAX, ")")
 read(N)
while N>MAX

for i←0 to N-1 do
 write(" grade ", i, "⇒")
 read(grade[i])
end for

aj ←0
for i←0 to N-1 do
 if grade[i]<10 then
    aj←aj+1
 end if
end for
write("The number of students who failed is", aj)
end
```

C

```
#include<stdio.h>
#define MAX 200
int main(){
 float grade[MAX] ;
 int i, N, aj=0; // aj càd nb ajournés
 // retrieve number of students
 do{
   printf(" enter number of students (<%d)",MAX) ;
   scanf("%f",&N) ;
 }while (N>MAX) ;
 // Fill in the table
 for(i = 0; i < N; i++){
   printf(" grade %d ⇒", i);
   scanf("%d", &grade[i]);
 }
 // calculate number of adjourned
 for(i = 0; i < N; i++)
   if(grade[i]<10) aj++ ;
 // result display.
 printf(" The number of students who failed is %d", aj);
}
```

3. Multidimensional arrays

A two-dimensional array (also called a matrix) is an array of arrays. The elements are accessible through two indices, the first specifying the row number and the second specifying the element number in that row (column).



Numéro de colonne

N ligne	0	1	2	3	4
0	15	7	-3	0	9
1	6	12	4	33	85
2	2	-8	17	28	-52
3	14	42	36	49	-12

Declaration

Algorithm

```
matrix_name : array[Lignes][Colonnes] of typeElements
```

C

```
typeElements matrix_name [Lignes][Colonnes] ;
```

Example

Algorithm

```
const L=100
const C=100
M : array[L][C] of real
mat1 : array[50][30] of integer
mat2 : array[30][20] of integer
```

C

```
const int L=100 , C=200 ;
float M[L][C] ;
int mat1[50][30],mat2[30][20];
```

Initialisation

```
int mat[][] = {{15, 7, -3 ,0 ,9},
               {6, 12, 4,33,85},
               {2, -8, 17 ,28,-52},
               {14, 42, 36, 49, -12}};
```

numéro de colonne

n ligne

	0	1	2	3	4
0	15	7	3-	0	9
1	6	12	4	33	85
2	2	8-	17	28	52-
3	14	42	36	49	12-

Use

To access a single element of the matrix, we use the name of the matrix with two sets of square brackets [ ] specifying the row number and the column number, respectively.

```
syntax    mat[ligne][ colonne]

example   mat[1][3]←mat[1][3]+2
```

Numéro de colonne

		0	1	2	3	4
N ligne	0	15	7	3-	0	9
	1	6	12	4	35	85
	2	2	8-	17	28	52-
	3	14	42	36	49	12-

Reading a matrix

```
for j←0 to C-1 do
  read(M[0][j])
end for
for j←0 to C-1 do
  read(M[1][j])
end for
…
for j←0 to C-1 do
  read(M[L-1][j])
end for
```

```
for i←0 to L-1 do
  for j←0 to C-1 do
    write("M[", i, ",", j, "]⇒")
    read(M[i][j])
  end for
end for
```

```
for(i = 0; i < L; i++)
 for(j = 0; j < C; j++){
  printf("M[%d, %d] ⇒", i,j);
  scanf("%d", &M[i][j]);
 }
```

Displaying a matrix

Algorithm

```
for i←0 to L-1 do
  for j←0 to C-1 do
    write(M[i][j])
  end for
end for
```

C

```
for(i = 0; i < L; i++){
  for(j = 0; j < C; j++)
    printf("%d\t", M[i][j]);
  printf("\n") ;
}
```

Example

Write a program that reads hourly temperatures for 30 days in the form of a matrix (30 by 24), then displays them on the screen. After that, it shows the highest temperature recorded and when it was recorded.

```
Algorithm nb_ajourned
Const Jr =30   Hr=24
var  T : array[Jr][Hr] of real
  maxT : real
  i, j, maxJr, maxHr : integer
début
for i←0 to Jr-1 do
   for j←0 to Hr-1 do
     write("T[",  i+1,  ",",  j,
 "]⇒")
     read(T[i][j])
   end for
end for
for i←0 to Jr-1 do
  for j←0 to Hr-1 do
    write(M[i][j])
  end for
end for
maxT←T[0][0]
maxJr←0
maxHr←0
for i←0 to Jr-1 do
   for j←0 to Hr-1 do
     if (T[i][j]>maxT) then
       maxT←T[i][j]
       maxJr←i
       maxHr←j
     end if
  end for
end for
write("The maximum temperature
       is ", maxT, " and was
       recorded on day ",
       maxJr+1, " at ", maxHr )
end
```

```c
#include<stdio.h>
#define Jr 30 // nb lignes
#define Hr 24 // nb colonnes
int main(){
float T[Jr][Hr] ,maxT;
int i, j,maxJr,maxHr;
// Fill in temperatures
for(i = 0; i < Jr; i++)
 for(j = 0; j < Hr; j++){
  printf("T[%d, %d] ⇒", i+1,j);
  scanf("%d", &T[i][j]);
 }
// display all temperatures
for(i = 0; i < Jr; i++){
 for(j = 0; j < Hr; j++)
   printf("%d\t", M[i][j]);
 printf("\n") ;
}
// search for maximum temperature
maxT=T[0][0];
maxJr=0 ;
maxHr=0 ;
for(i = 0; i < Jr; i++)
 for(j = 0; j < Hr; j++)
  if (T[i][j]>maxT){
   maxT=T[i][j];
   maxJr=i;
   maxHr=j;
  }
// display of results
printf(" The maximum temperature is
     %d and was recorded on day %d
     at %d", maxT, maxJr+1, maxHr) ;
}
```

4. Strings

- A string of characters is an ordered set of characters.
- They are always enclosed in double quotes « " »
  such as "computer science", "Good luck\n", "1", "3.14".
- In C, character arrays (char []) are used to create strings.
- When reading a string from the keyboard, each character is placed in a memory area and the character '\0' is added at the end of the text to indicate its termination.
- The character '\0' is called "**null**" and it has the ASCII code **0**.
- There is a constant declared in the « stdio.h » library called **NULL** in uppercase.

```
#define NULL 0
NULL ⇔ '\0' ⇔ 0
```

Declaration & Initialization

Algorithm

```
   var str : string
   var str :array [30] of characters
```

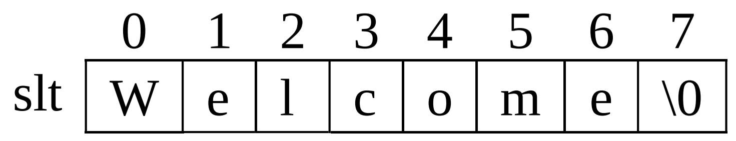
in C

```
   char str[30] ;
```

Initialization in C

```
      char slt[] = {'W', 'e', 'l', 'c', 'o', 'm', 'e', '\0'};
```

This instruction creates an array of 8 characters (7 positions for the word 'Welcome' and one slot containing the character '\0'). However, there is a simpler and faster way to create and initialize a string:

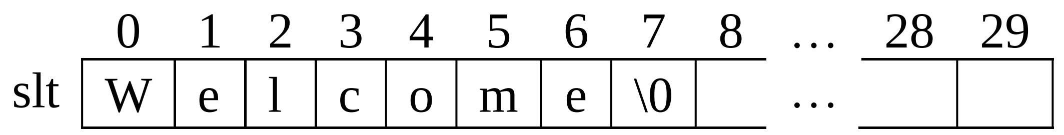


```
      char slt[] = "Welcome";
```

This leads to the same result, which is the creation of an array of 8 characters, ending with the character '\0'.

The size of the array can also be specified:

```
      char slt[30] = "Welcome";
```



Assignment

Since strings are arrays, a string cannot be assigned to a variable directly after its declaration. The following operation is incorrect.

```
char slt[30];
~~slt = "Welcome";~~// erreur : assignment to an array.
```

To assign a string to a variable or copy a variable to another variable, we use the strcpy() function.

```
strcpy(slt , "Welcome" );
```

Display & Reading

Display

```
    write(str)
```

In C

```
   printf("%s",str) ;
   Puts(str);
```

Reading

```
    read(str)
```

In C

```
   scanf("%s",str) ;
```

or

```
   #include <string.h>
   …
   gets(slt) ; // for text containing spaces
```

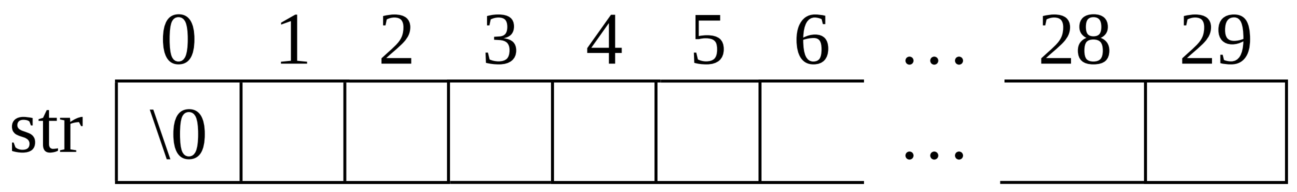
Some functions specific to strings in C

`strcpy(dest, src);`	Copies the string from src to the destination (dest)
`strcat(s1, s2);`	Concatenates (appends) s2 to s1.
`strlen(s);`	Returns the length of the string s (excluding '\0')
`strcmp(s1, s2);`	Returns 0 if s1 and s2 are identical ; less than 0 if s1<s2 ; greater than 0 if s1>s2.

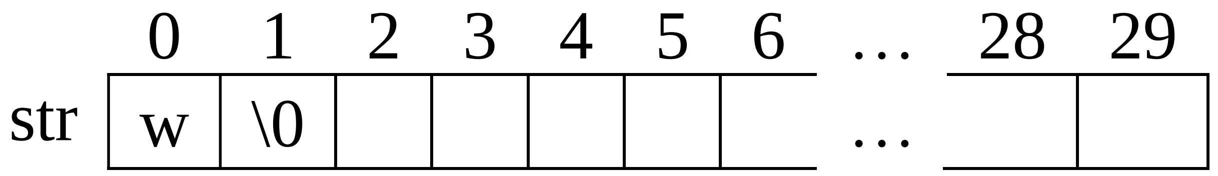
Examples

Example 1 : Empty string str="" ; which has a length of 0.

The content of the cells after '\0' doesn't matter; the string ends at the first '\0'. Therefore, any string can be converted to an empty string by placing str[0]='\0' ;



Example 2 : A string containing a single character is different from the character type. Thus, "w"≠'w' because "w" is a string.



Examples

Example 3 : Write a program that takes input text and converts uppercase letters to lowercase and lowercase letters to uppercase.

```
algorithm inverse
var txt :array[200] of charactere
    i : integer
begin
 write((" enter text ")
 read(txt)
 i←0
 while txt[i]≠'\0' do
   if txt[i]>='A' and txt[i]<='Z' then
    txt[i]=txt[i]+'a'-'A'
   else
     if (txt[i]>='a' and txt[i]<='z') then
        txt[i]=txt[i]-('a'-'A');
     end if
   end if
 end while
 write(txt)
end.
```

Examples

```
#include<stdio.h>
#include<string.h>
int main(){
 char txt[200] ;
 int i ;
 printf(" enter text \n") ;
 gets(txt)
 for(i=0 ;txt[i] !=’\0’ ;i++)
  if (txt[i]>=’A’&&txt[i]<=’Z’)
   txt[i]+=’a’-’A’ ;
  else
   if (txt[i]>=’a’&&txt[i]<=’z’)
     txt[i]-=’a’-’A’ ;

 printf("%s",txt) ;
 return 0 ;
}
```

End Chapter 05