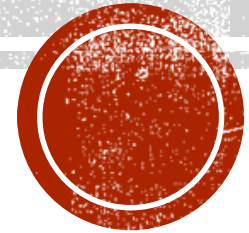
ARRAY OF ONE DIMENSION/ VECTOR

2020-2021



CHAPTER HEADLINES

- Part 1
  - Introduction
  - Definition
  - Declaration
  - Storage
  - Initialization
  - Automatic reservation
- Part 2
  - Access to components
  - Displaying the content of an array
  - Assignment with values from keyboard
- Part 3
  - Exercises

CHAPTER HEADLINES

- **Part 1**
  - **Introduction**
  - Definition
  - Declaration
  - Storage
  - Initialization
  - Automatic reservation
- Part 2
  - Access to components
  - Displaying the content of an array
  - Assignment with values from keyboard
- Part 3
  - Exercises

INTRODUCTION

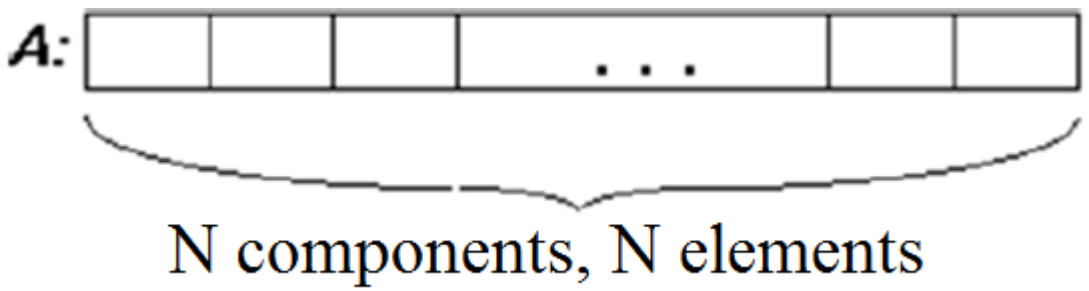
Arrays are certainly the most popular structured variables. They are available in all programming languages and are used to solve a multitude of problems.

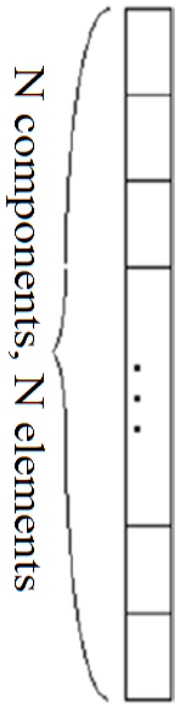
CHAPTER HEADLINES

- **Part 1**
  - Introduction
  - **Definition**
  - Declaration
  - Storage
  - Initialization
  - Automatic reservation
- Part 2
  - Access to components
  - Displaying the content of an array
  - Assignment with values from keyboard
- Part 3
  - Exercises

DEFINITION

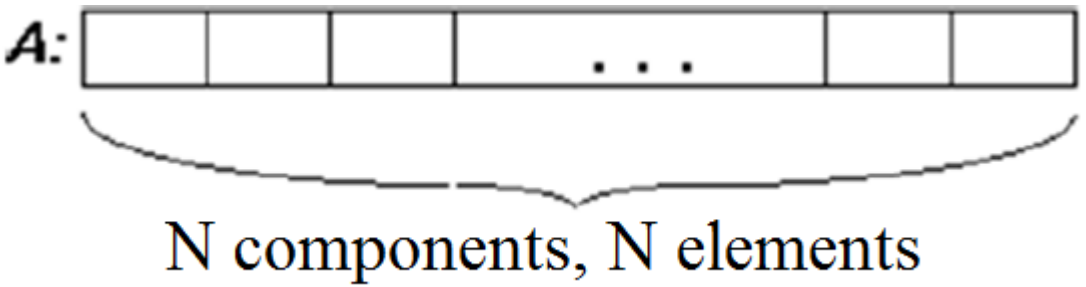
- A (uni-dimensional) array A is a structured variable formed of N variables of the same type, which are called the components of the array (or the elements of the array).

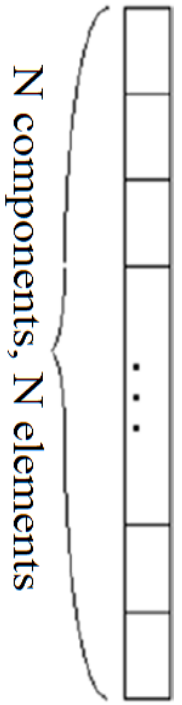




DEFINITION

- The number of components/elements is called the dimension/size of the array.
- Array A is called also a vector of dimension N





CHAPTER HEADLINES

- **Part 1**
  - Introduction
  - Definition
  - **Declaration**
  - Storage
  - Initialization
  - Automatic reservation
- Part 2
  - Access to components
  - Displaying the content of an array
  - Assignment with values from keyboard
- Part 3
  - Exercises

DECLARATION

- <Type> <ArrayName> [<Dimension >] ;
  - Example:

    int A[25]; float B[100]; int C[10] ; char D[30] ;

DECLARATION

- **<Type>** **<ArrayName>** **[<Dimension >]** **;**
  - Example:

    int A[25]; float B[100]; int C[10] ; char D[30] ;

- **Type :** could be any predefined type: int, float, double, short, char, bool, long, … it determines the type of the components/elements of the array

DECLARATION

- <Type> <ArrayName> [<Dimension >] ;
  - Example:

    int A[25]; float B[100]; int C[10] ; char D[30] ;

- **Type :** could be any predefined type: int, float, double, short, char, bool, long, …
- **ArrayName** is the name of the array (same conditions of the name of variables:
  - ✓Does not start with numbers, exp: int 1A[25]; ERROR
  - ✓Does not contain punctuations characters (space, comma, semi-colon, …) exp: int grades of students[30]; ERROR

DECLARATION

- **<Type> <ArrayName> [<Dimension >] ;**
  - Example:

    int A[25]; float B[100]; int C[10] ; char D[30] ;

- **Type :** could be any predefined type: int, float, double, short, char, bool, long, …
- **ArrayName** is the name of the array (same conditions of the name of variables:
  - ✓Does not start with numbers, exp: int 1A[25]; ERROR
  - ✓Does not contain punctuations characters (space, comma, semi-colon, …) exp: int grades of students[30]; ERROR
- **Dimension:**
  - ✓Integer number only, exp: int A[20.5]; ERROR,
  - ✓Positive number Only exp: float B [-15]; ERROR
  - ✓Constant value, int A[x]; ERROR

DECLARATION

- <Type> <ArrayName> [<Dimension >] ;
- **Dimension:**
  - ✓Integer number only,
  - ✓Positive number Only
  - ✓Constant value,

```
int main()
{
    int x;
    float A[x]; //ERROR
    cin>>x;
    …..
}
```

```
int main()
{
    int x =5;
    float A[x]; //OK
    …..
    …..
}
```

DECLARATION

- <Type> <ArrayName> [<Dimension >] ;
- **Dimension:**
  - ✓Integer number only,
  - ✓Positive number Only
  - ✓Constant value,

    Static reservation (opposite of dynamic reservation where the dimension of the array is precised during the running phase).

CHAPTER HEADLINES

- **Part 1**
  - Introduction
  - Definition
  - Declaration
  - **Storage**
  - Initialization
  - Automatic reservation
- Part 2
  - Access to components
  - Displaying the content of an array
  - Assignment with values from keyboard
- Part 3
  - Exercises

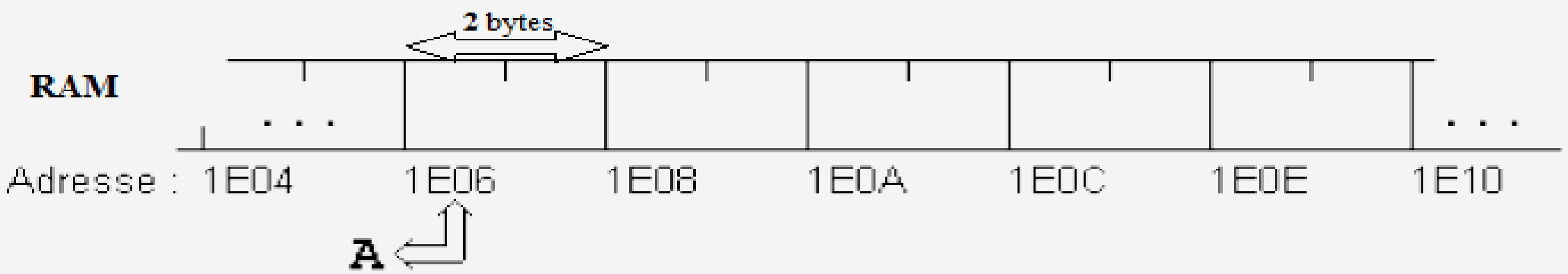
STORAGE

- In C, the name of an array is the representative of the address of the first element of the array. The addresses of the other components are calculated (automatically) relative to this address.

STORAGE

- Example 1:

short A[5];



1E06= 0001 1110 0000 0110
1E0A=0001 1110 0000 1010
1E0E=0001 1110 0000 1110
1E08=0001 1110 0000 1000
1E0C=0001 1110 0000 1100
1E10= 0001 1110 0001 0000

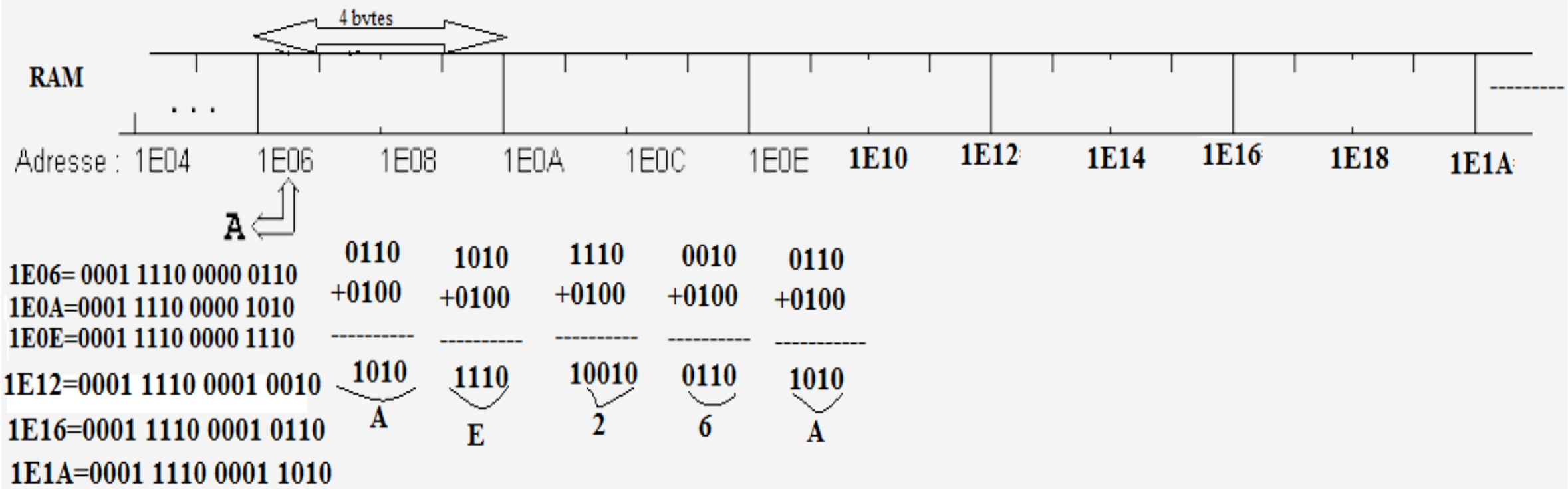
+ 0110 + 0010 = 1000
1010 + 0010 = 1100
1110 +0010 = 10000

A will reserve 5 * 2 bytes = 10 bytes in memory.

STORAGE

- Example 2:

int A[5];



A will reserve 5 * 4 bytes = 20 bytes in memory.

CHAPTER HEADLINES

- **Part 1**
  - Introduction
  - Definition
  - Declaration
  - Storage
  - **Initialization**
  - Automatic reservation
- Part 2
  - Access to components
  - Displaying the content of an array
  - Assignment with values from keyboard
- Part 3
  - Exercises

INITIALIZATION

- When declaring an array, we can initialize the components/elements of the table/array, by indicating the list of the respective values between braces.

INITIALIZATION

- When declaring an array, we can initialize the components/elements of the table/array, by indicating the list of the respective values (separated by comma) between braces.

```
int A[5]    = {10, 20, 30, 40, 50};
float B[4] = {-1.05, 3.33, 87e-5, -12.3E4};
int C[10]   = {1, 0, 0, 1, 1, 1, 0, 1, 0, 1};
```

INITIALIZATION

- Of course, you must ensure that the number of values in the list matches the size of the array (the number of values ≤ dimension of the array) .
  - Short T[3]={2,10,8}; OK
  - int C[3]={1, -6, 4, 20}; ERROR
  - float A[5]={1,6}; OK
- If the list does not contain enough values for all components/elements, the remaining components/elements are initialized with zero.

A

1	6	0	0	0

A

1
6
0
0
0

CHAPTER HEADLINES

- Part 1
  - Introduction
  - Definition
  - Declaration
  - Storage
  - Initialization
  - **Automatic reservation**
- Part 2
  - Access to components
  - Displaying the content of an array
  - Assignment with values from keyboard
- Part 3
  - Exercises

AUTOMATIC RESERVATION

- If the dimension is not specified explicitly during initialization, then the computer automatically reserves the necessary number of bytes.

✓ int A[ ] = {10 , 20 , 30 , 40 , 50 };

➔A is one dimensional array of 5 elements.

✓ float B [ ] = { -1.05 , 3.33 , 87e -5 , -12.3E4 };

➔ B is one dimensional array of 4 elements

✓ int C[ ] = {1 , 0 , 0 , 1 , 1 , 1 , 0 , 1 , 0 , 1 } ;

➔C is one dimensional array of 10 elements

AUTOMATIC RESERVATION

- If the dimension is not specified explicitly during initialization, then the computer automatically reserves the necessary number of bytes.

✓ int A[ ] = {10 , 20 , 30 , 40 , 50 };

➔A is one dimensional array of 5 elements.

✓ float B [ ] = { -1.05 , 3.33 , 87e -5 , -12.3E4 };

➔ B is one dimensional array of 4 elements

✓ int C[ ] = {1 , 0 , 0 , 1 , 1 , 1 , 0 , 1 , 0 , 1 } ;

➔C is one dimensional array of 10 elements

- int A[]; ERROR
- Float B[]; ERROR

CHAPTER HEADLINES

- Part 1
  - Introduction
  - Definition
  - Declaration
  - Storage
  - Initialization
  - Automatic reservation
- **Part 2**
  - **Access to components**
  - Displaying the content of an array
  - Assignment with values from keyboard
- Part 3
  - Exercises

ACCESS TO COMPONENTS

- By declaring an array by: int A[5] ;

we have defined an array A with 5 components/elements,

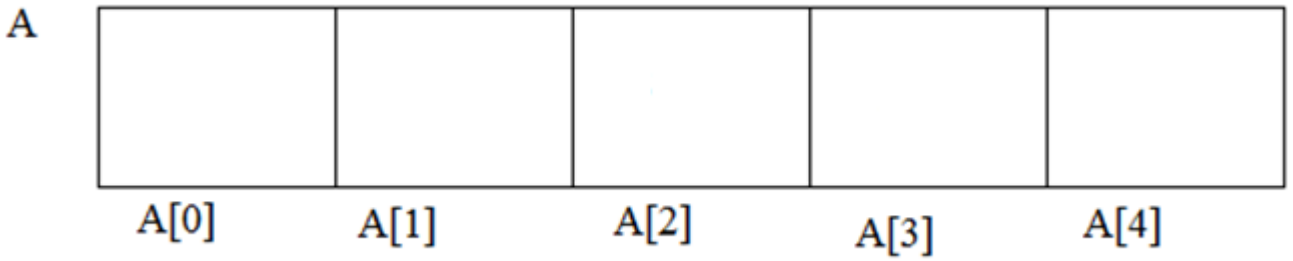
ACCESS TO COMPONENTS

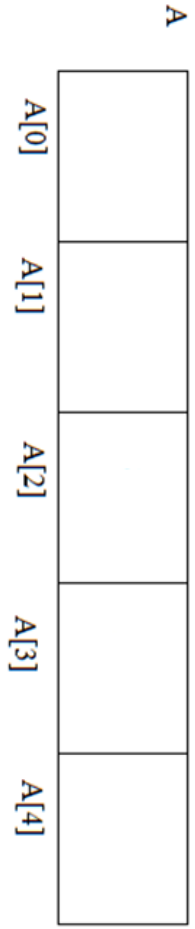
- By declaring an array by: int A[5] ;

we have defined an array A with 5 components/elements.

These elements are:

A[0] , A[1] , A[2], A[3], A[4]





ACCESS TO COMPONENTS

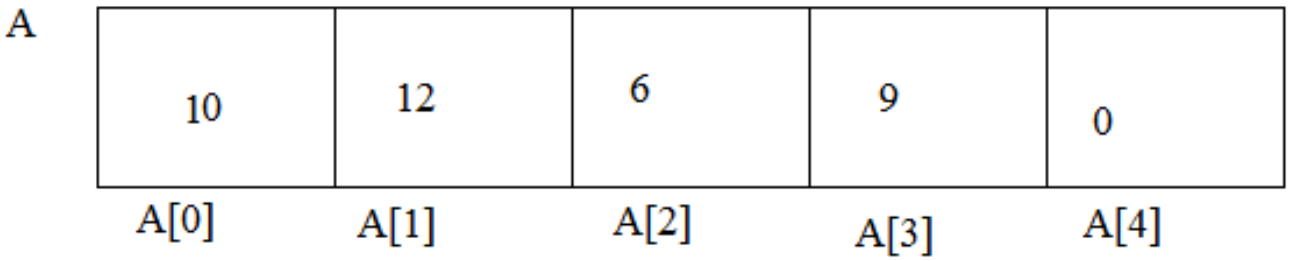
- By declaring an array by: int A[5] ;

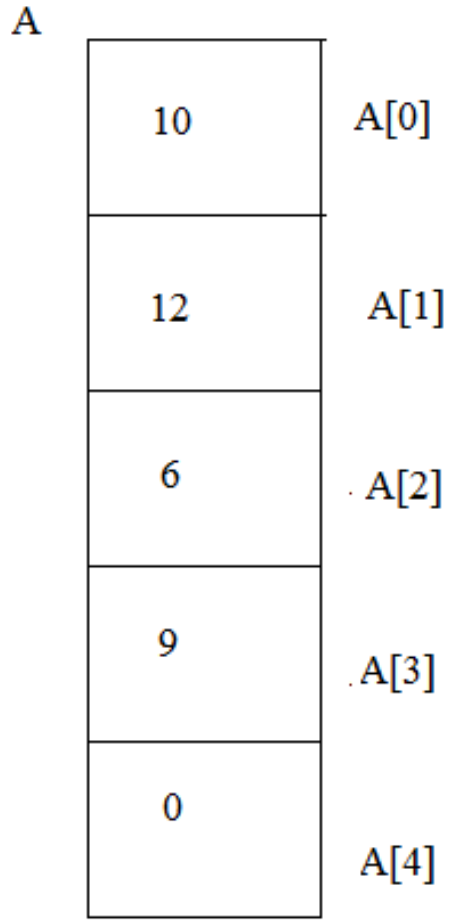
we have defined an array A with 5 components/elements, which can be accessed by:

A[0] , A[1] , A[2], A[3], A[4]

Example:

int A[5]={10,12,6,9};





ACCESS TO COMPONENTS

- Access to the element of rank *i* of an array A can be done by A[i],
- The value of *i* should be ≥ 0 and ≤N-1 (where N is the dimension/size of the array A)
  1. **Constante value**

Example :

int A[5]={10,12,6,9},i=1;

A[0]=-30; A[1]= A[0]+2*A[2];

ACCESS TO COMPONENTS

- Access to the element of rank i of an array A can be done by A[i],
- The value of i should be ≥ 0 and ≤N-1 (where N is the dimension/size of the array A)
  1. Constante value
  2. **Variable**

Example :

```
int A[5]={10,12,6,9},i=1;
A[i]=-30;
i=2;
A[i]=13;
```

ACCESS TO COMPONENTS

- Access to the element of rank i of an array A can be done by A[i],
- The value of i should be ≥ 0 and ≤N-1 (where N is the dimension/size of the array A)
  1. Constante value
  2. Variable
  3. **Expression**

Example :

```
int A[5]={10,12,6,9},i=1;
A[i+1]=-2;
A[2*i+1]=15;
```

ACCESS TO COMPONENTS

- Access to the element of rank i of an array A can be done by A[i],
- The value of *i* should be **integer** type (integer types: int, short, char, long), but not float, double, bool,....

Example:

```
int A[5]={10,12,6,9};
float i=1;
cout<<A[i]; //ERROR
```

CHAPTER HEADLINES

- Part 1
  - Introduction
  - Definition
  - Declaration
  - Storage
  - Initialization
  - Automatic reservation
- **Part 2**
  - Access to components
  - **Displaying the content of an array**
  - Assignment with values from keyboard
- Part 3
  - Exercises

DISPLAYING THE CONTENT OF AN ARRAY

DISPLAYING THE CONTENT OF AN ARRAY

- The for structure is particularly suitable for working with arrays.

```
#include<iostream>
using namespace std;
int main()
{
float A[5]={3.5,5,-7.1};
int i;
for (i=0;i<=4;i++)
cout<<A[i]<<",";
cout<<"\n end";
return 0;
}
```

```
3.5,5,-7.1,0,0,
 end
--------------------------------
Process exited after 1.76 seconds with return value 0
Press any key to continue . . .
```

DISPLAYING THE CONTENT OF AN ARRAY

- The for structure is particularly suitable for working with arrays.

```
#include<iostream>
using namespace std;
int main()
{
float A[5]={3.5,5,-7.1};
int i;
for (i=0;i<=4;i++)
cout<<"A["<<i<<"]="<<A[i]<<",";
cout<<"\n end";
return 0;
}
```

```
A[0]=3.5,A[1]=5,A[2]=-7.1,A[3]=0,A[4]=0,
 end
--------------------------------
Process exited after 0.03076 seconds with return value 0
Press any key to continue . . .
```

CHAPTER HEADLINES

- Part 1
  - Introduction
  - Definition
  - Declaration
  - Storage
  - Initialization
  - Automatic reservation
- **Part 2**
  - Access to components
  - Displaying the content of an array
  - **Assignment with values from keyboard**
- Part 3
  - Exercises

ASSIGNMENT WITH VALUES FROM KEYBOARD

ASSIGNMENT WITH VALUES FROM KEYBOARD

```
#include<iostream>
using namespace std;
int main()
{
float A[5];
int i;
//Assign the array A with values from the keyboard
for (i=0;i<=4;i++)
cin>>A[i];
//Display the content of the array A
for (i=0;i<=4;i++)
cout<<A[i]<<",";
return 0;
}
```

```
8
-4
17
13
2.5
8,-4,17,13,2.5,
--------------------------------
Process exited after 15.39 seconds with return value 0
Press any key to continue . . .
```

ASSIGNMENT WITH VALUES FROM KEYBOARD

```
#include<iostream>
using namespace std;
int main()
{
float A[5];
int i;
//Assign the array A with values from the keyboard
for (i=0;i<=4;i++)
{
cout<<"enter value of A["<<i<<"]:";
cin>>A[i];
}
//Display the content of the array A
for (i=0;i<=4;i++)
cout<<A[i]<<",";
return 0;
}
```

```
enter value of A[0]:10
enter value of A[1]:2
enter value of A[2]:16
enter value of A[3]:8
enter value of A[4]:13
10,2,16,8,13,
--------------------------------

Process exited after 11.3 seconds with return value 0
Press any key to continue . . .
```

CHAPTER HEADLINES

- Part 1
  - Introduction
  - Definition
  - Declaration
  - Storage
  - Initialization
  - Automatic reservation
- Part 2
  - Access to components
  - Displaying the content of an array
  - Assignment with values from keyboard
- **Part 3**
  - **Exercises**

EXERCISE 1:

Write a program that reads the dimension N of an integer array T (maximum dimension: 50 components), fills the array with values entered on the keyboard and displays the array. Calculate and then display the sum of the elements of the array.

EXERCISE 1: SOLUTION

```
int main()
{
int T[50], N,i,S;
do
{
    cout<<"enter the dimension of the array N: 0<N<=50: ";
    cin>>N;
}while(N<=0 || N>50);
for(i=0;i<N;i++)
{
    cout<<"enter T["<<i<<"]=:";
    cin>>T[i];
}
for(i=0;i<N;i++)
    cout<<"T["<<i<<"]="<<T[i]<<"\t";
for(S=0,i=0;i<N;i++)
    S=S+T[i];
cout<<"\n sum of all the elements = "<<S;
return 0;
}
```

```
enter the dimension of the array N: 0<N<=50: -4
enter the dimension of the array N: 0<N<=50: 5
enter T[0]=:7
enter T[1]=:-6
enter T[2]=:13
enter T[3]=:2
enter T[4]=:10
T[0]=7  T[1]=-6 T[2]=13 T[3]=2  T[4]=10
 sum of all the elements = 26
--------------------------------
Process exited after 23.37 seconds with return value 0
Press any key to continue . . .
```

EXERCISE 2

Write a program that reads the dimension N of an integer array T (maximum dimension: 50 components), fills the array with values entered on the keyboard and displays the array. The program makes a change in the content of the array, it stores the elements of T in the reverse order without using an auxiliary array. Display the resulting array.

Hint: Swap the elements of the array using two indices that run through the array, starting at the beginning and at the end of the array respectively and meeting in the middle.

```cpp
#include<iostream>
using namespace std;
int main()
{
int T[50],m;
int i,j,N;
do
{
    cout<<"enter the effective dimension of the array >0 and <50 :";
    cin>>N;
}while(N<=0 || N>=50);
//Assign the array T with values from the keyboard
for (i=0;i<N;i++)
{
cout<<"enter value of T["<<i<<"]:";
cin>>T[i];
}
//Display the content of the array T
for (i=0;i<N;i++)
cout<<T[i]<<",";
//make the change
for (i=0,j=N-1;i<N/2;i++,j--)
{
    m=T[i];
    T[i]=T[j];
    T[j]=m;
}
//Display the content of the array T
cout<<"\n array T becomes :\n";
for (i=0;i<N;i++)
cout<<T[i]<<",";
return 0;
}
```

```
enter the effective dimension of the array >0 and <50 :10
enter value of T[0]:5
enter value of T[1]:6
enter value of T[2]:7
enter value of T[3]:8
enter value of T[4]:9
enter value of T[5]:10
enter value of T[6]:11
enter value of T[7]:12
enter value of T[8]:13
enter value of T[9]:14
5,6,7,8,9,10,11,12,13,14,
 array T becomes :
14,13,12,11,10,9,8,7,6,5,
--------------------------------
Process exited after 11.15 seconds with return value 0
Press any key to continue . . .
```

EXERCISE 3

Write a program that reads the dimension N of a float array T (maximum dimension: 50 components), fills the array with values entered on the keyboard and displays the array. Then copy all positive components into a second TPOS array and all strictly negative values into a third TNEG array. Display TPOS and TNEG arrays.

```cpp
#include<iostream>
using namespace std;
int main()
{
float T[50],TPOS[50],TNEG[50];
int i,j,pos,neg,N;
do
{
    cout<<"enter the effective dimension of the array >0 and <50 :";
    cin>>N;
}while(N<=0 || N>=50);
//Assign the array T with values from the keyboard
for (i=0;i<N;i++)
{
cout<<"enter value of T["<<i<<"]:";
cin>>T[i];
}
//Display the content of the array T
for (i=0;i<N;i++)
cout<<T[i]<<",";
//filling the arrays TPOS, TNEG
for (i=0,pos=0,neg=0;i<N;i++)
    if(T[i]>=0)
    {
        TPOS[pos]=T[i];
        pos++;
    }

    else
    {
        TNEG[neg]=T[i];
        neg++;
    }
//Display the content of the array TPOS
cout<<"\n array TPOS :\n";
for (i=0;i<pos;i++)
cout<<TPOS[i]<<",";
//Display the content of the array TNEG
cout<<"\n array TNEG :\n";
for (i=0;i<neg;i++)
cout<<TNEG[i]<<",";
return 0;
}
```

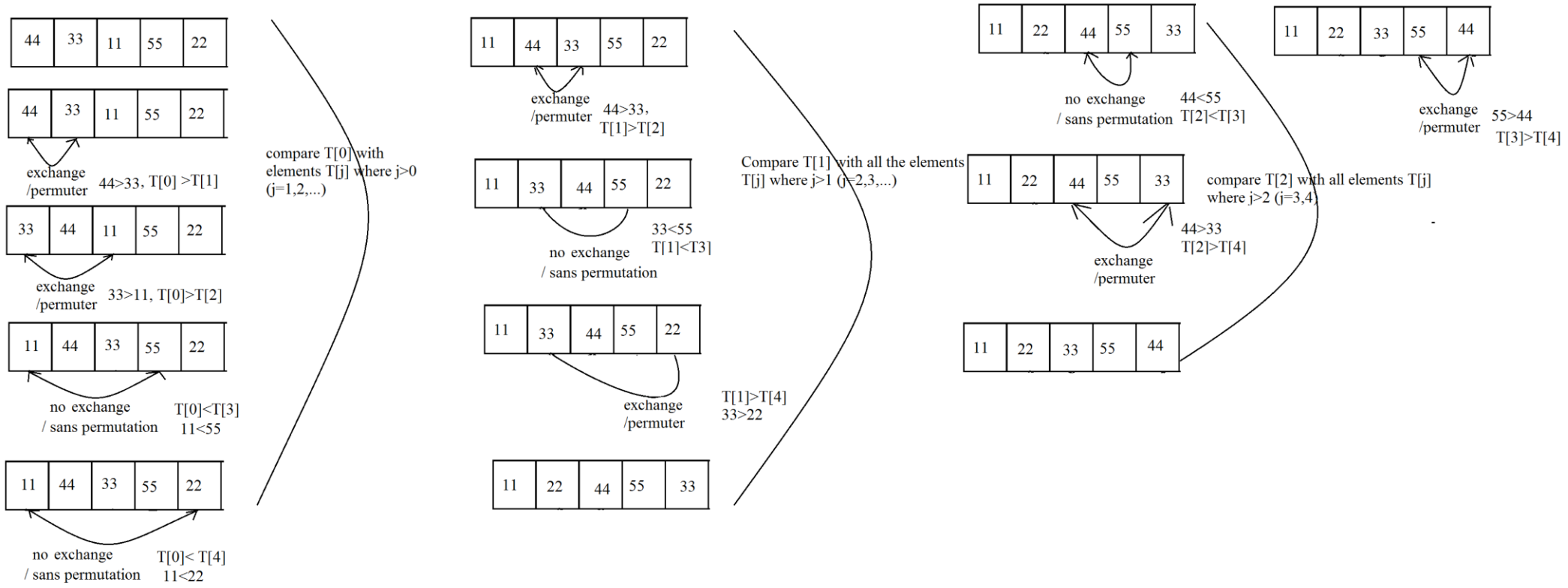
```
enter the effective dimension of the array >0 and <50 :10
enter value of T[0]:8
enter value of T[1]:-6
enter value of T[2]:45
enter value of T[3]:85
enter value of T[4]:-2
enter value of T[5]:-16
enter value of T[6]:35
enter value of T[7]:-45
enter value of T[8]:32
enter value of T[9]:-5
8,-6,45,85,-2,-16,35,-45,32,-5,
 array TPOS :
8,45,85,35,32,
 array TNEG :
-6,-2,-16,-45,-5,
--------------------------------
Process exited after 35.61 seconds with return value 0
Press any key to continue . . .
```

EXERCISE 4

- Write a program that reads the dimension N of an integer array T (maximum dimension: 50 components), fills the array with values entered on the keyboard and displays the array. Then sort the elements of the array in ascending order. Finally, display the array after sorting it.

Hint: using the method described in the next slide.

FOR EACH ELEMENT OF INDEX *I* (T[I]) IN THE ARRAY, WE COMPARE THE VALUE OF THIS ELEMENT WITH ALL THE OTHER ELEMENTS OF INDEX *J* (I.E. T[J]) HAVING INDICES GREATER THAN *I*, I.E. *J*>*I*. THE VALUES OF THE ELEMENTS T[I] AND T[J] WILL BE EXCHANGED WHENEVER T[I] IS FOUND GREATER THAN T[J] (I.E. T[I]>T[J] WE MAKE THE EXCHANGE).



```cpp
#include<iostream>
using namespace std;
int main()
{
float T[50],temp;
int i,j,N;
do
{
    cout<<"enter the effective dimension of the array >0 and <50 :";
    cin>>N;
}while(N<=0 || N>=50);
//Assign the array T with values from the keyboard
for (i=0;i<N;i++)
{
cout<<"enter value of T["<<i<<"]:";
cin>>T[i];
}
//Display the content of the array T
for (i=0;i<N;i++)
cout<<T[i]<<",";
for(i=0;i<N;i++)
    for(j=i+1;j<N;j++)
        if(T[i]>T[j])
        {
            temp=T[i];
            T[i]=T[j];
            T[j]=temp;
        }
//Display the content of the array T after sorting
cout<<" the array after sorted :\n";
for (i=0;i<N;i++)
cout<<T[i]<<",";
return 0;
}
```

```
enter the effective dimension of the array >0 and <50 :10
enter value of T[0]:25
enter value of T[1]:6
enter value of T[2]:-2
enter value of T[3]:45
enter value of T[4]:32
enter value of T[5]:10
enter value of T[6]:-4
enter value of T[7]:2
enter value of T[8]:-10
enter value of T[9]:13
25,6,-2,45,32,10,-4,2,-10,13, the array after sorted :
-10,-4,-2,2,6,10,13,25,32,45,
--------------------------------
Process exited after 26.97 seconds with return value 0
Press any key to continue . . .
```