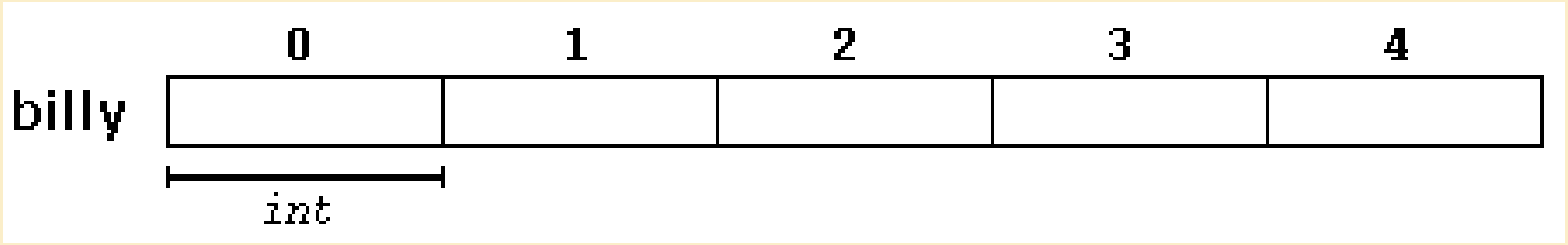
Array



Dr. Zaid Ameen

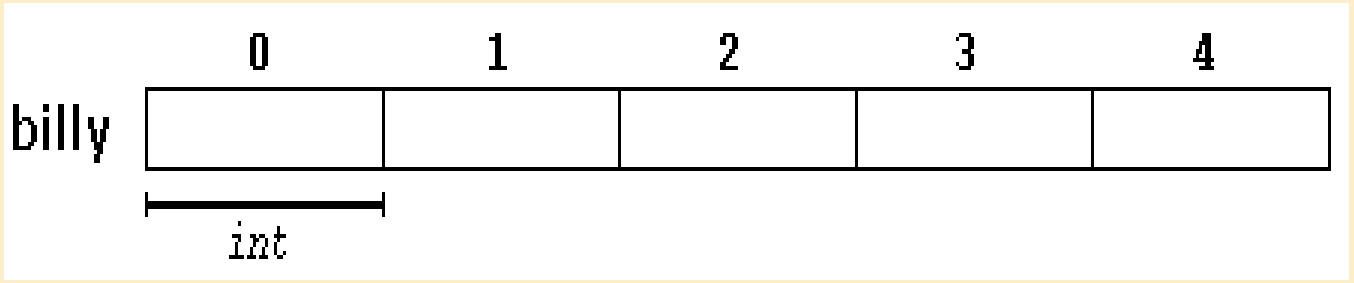
5. Array

An array is a series of elements of the same type placed in contiguous memory locations that can be individually referenced by adding an index to a unique identifier. Arrays can be divided to one dimension, two dimension or multi-dimension.

5.1 One Dimension Array

Just like any variable you are already familiar with, an array has to be declared before being used. Yet the difference this time is that you need to tell the compiler what kind of array you are defining, an array of books? An array of students? An array of billiard balls? An array of clothes? This is because, once more, the compiler wants to know how much space your array is going to occupy in the computer memory. This is because when you declare an array of items, the compiler puts each one of the items in an appropriate location.
For example, an array to contain 5 integer values of type int called billy could be

represented like this:



Where each blank panel represents an element of the array, that in this case is integer values of type int. These elements are numbered from 0 to 4 since in arrays the first index is always 0, independently of its length. A typical declaration for an array in C++ is: DataType ArrayName [dimension\order]

The array is first identified by its kind, which could be a char, an int, a float, etc.; followed by its name that follows the C++ naming rules. The name is then followed by square brackets that specify the dimension of the array or its size.

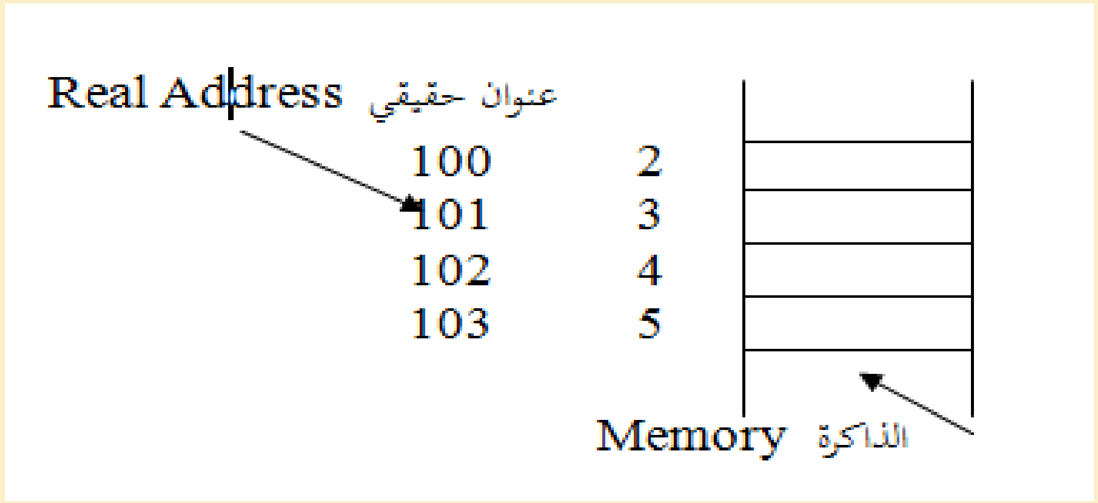
Here are examples of declaring arrays:

```
int age[12];

float grade[100];

double angle[360];
```

int Age[12]; declares a group or array of 12 values, each one being an integer. float Grade[100]; declares an array of 100 floating-point values.
double Angle[360]; declares an array of double-precision numbers. There are 360 of these items in the group.

An individual element within an array is accessed by use of an Index which describes the position of an element within an array.

Loc (A [I]) = Real Address + I

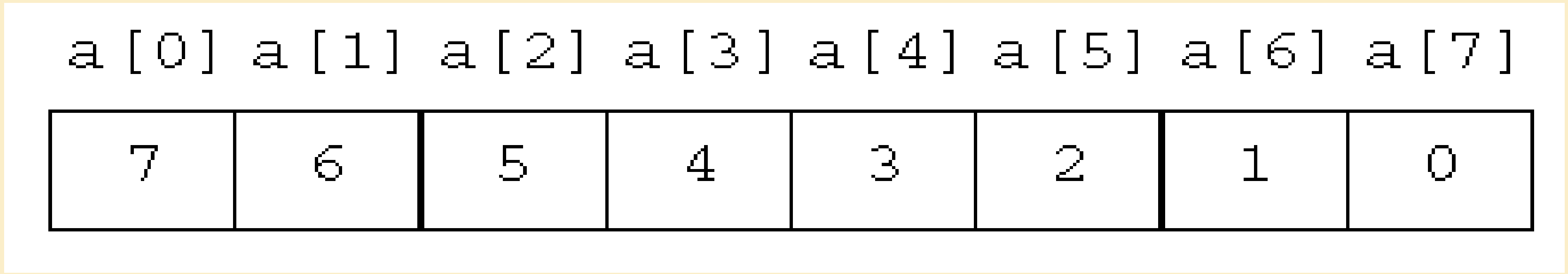


Note: In C++ the first element has the index zero!

Example: int a [8]; int j;

for(j=0; j<8; j++) a[j] = 7-j; // a[j=0] =7-0= 7; a[j=1]=7-1=6;......a[j=7]= 7-7=0

Then the memory representation of array a looks like this:



a[0]	a[1]	a[2]	a[3]	a[4]	a[5]	a[6]	a[7]
7	6	5	4	3	2	1	0

1) Write a C++ program to read integer array which contains n elements and then print array elements.

```
#include <iostream>
using namespace std;
void Read1D(int [], int n);
void Print1D(int [], int n);
int main()
{
   int n, a[100];
   cout << "enter n" << endl;
   cin >> n;
   Read1D(a, n);
   Print1D(a, n);
   return 0;
}
void Read1D(int a[100], int n)
{
   for (int i = 0; i < n; i++)
   {
      cout << "enter a[" << i << "]" << endl;
      cin >> a[i];
   }
}
void Print1D(int a[100], int n)
{
   for (int i = 0; i < n; i++)
   {
      cout << a[i] << endl;
   }
}
```

2) Write a C++ program to read integer array a[100] which contains n elements and then compute the following operations: 1) print inverse array. 2) Maximum elements in array a. 3) C= X+4a; X is float variable.

```
#include <iostream>    a= {1, 2, 3, 4, 5, 6}; //  inverse a ={6, 5, 4, 3, 2, 1}// x = 2, c= { 2+4 * 1= 6, 2 + 4 *2 =10, 2+ 4 * 3= 14,..
using namespace std;
void Read1D(int [100], int);
void Printinv1D(int[100], int);
void Max(int[100], int);
void GenC(int[100], int, int, int[100]);
void Print1D(int[100], int);
int main()
{
  int n, X, a[100], c[100];
  cout << "enter n" << endl;
  cin >> n;
  Read1D(a, n);
  Printinv1D(a, n);
  Max(a, n);
  cout << "Enter X" << endl;
  cin >> X;
  GenC(a, n, X, c);
  Print1D(c, n);
  return 0;
}
void Read1D(int a[100], int n)
{
   f or (int i = 0;i < n;i++) {
       cout << "enter a[" << i << "]" << endl; cin >> a[i];
   }
}
```

```
void Printinv1D(int a[100], int n)
{
   for (int i = n-1;i >= 0;i--)
    {
     cout << a[i] << endl;
    }
}
void Max(int a[100], int n)
{
  int M = a[0];
  for (int i = 0;i < n;i++)
   {
    if (a[i] > M)
     M = a[i];
    }
   cout << "Max = " << M << endl;
}
void GenC(int a[100], int n, int X, int c[100])
{
for (int i = 0;i < n;i++)
  {
    c[i] = X + 4 * a[i];
  }
}
void Print1D(int c[100], int n)
{
   for (int i = 0;i < n;i++)
     {
      cout << "enter c[" << i << "]=  " << c[i]<< endl;
     }
}
```

3) Write a C++ program to allow a user to enter 5 quiz scores to calculate an average. It will then output the average on screen.

```
#include <iostream>
using namespace std;
void Read1D(int [5], int);
void Avg(int[5], float&);
int main()
{
int scores[5];
float av;
Read1D(scores, 5);
Avg(scores, av);
cout << "average is" << av << endl;
return 0;
}

void Read1D(int scores[5], int n)
{
for (int i = 0;  i < n; i++)
{
cout << "enter scores[" << i << "]" << endl;
cin >> scores[i];
}
}
void Avg(int scores[5], float& av)
{
float s = 0;
for (int i = 0; i < 5; i++)
s += scores[i];
av = s / 5;
}
```

4) Write a C++ program to read integer array A[10] and then compute the following operations:

1) Summation of even element in A. 2) Count the prim element in A (H.W).

3) print odd element in A.

```
#include <iostream>
using namespace std;
void Read1D(int[10], int);
void Sum(int[10], int);
void printodd(int[10], int);
int main()
{
    int A[10];
    Read1D(A, 10);
    Sum(A, 10);
    printodd(A, 10);
}
void Read1D(int  A[10], int n)
{
    for (int i = 0;i < n;i++)
    {
        cout << "enter A[" << i << "]" << endl;
        cin >> A[i];
    }
}
```

```
void Sum(int A[10], int n)
    {
        int s = 0;
        for (int i = 0;i < n;i++)
        {
            if (A[i] % 2 == 0)
                s += A[i];
        }
        cout << "sum is" << s << endl;
    }

    void printodd(int A[10], int n)
    {
        for (int i = 0;i < n;i++)
        {
            if (A[i] % 2 != 0)
                cout << A[i] << endl;
        }
    }
```

Searching for data is one of the fundamental fields of computing. Often, the difference between a fast program and a slow one is the use of a good algorithm for the data set. This subsection will focus on searching for data stored in a linear data structure such as an array.

1-Linear Search

The most obvious algorithm is to start at the beginning and walk to the end, testing for a match at each item:

The sequential search is simple: we start at the first array element and compare each key with the target until we have either found the target or reached the end of the array without finding it. The result of function is returned true if the search is successful, and false value if the search fails.

item: 101

Location	0	1	2	3	4	5	6	7	8	9
Value	23	11	87	2	9	78	88	54	39	63

1) Sequential search

```
#include <iostream>
using namespace std;
void Read1D(int  array[100], int n)
{
   for (int i = 0;i < n;i++)
     {
       cout << "enter A[" << i << "]" << endl;
       cin >> array[i];
      }
}

void Seq_search(int array[], int size, int key, int& loc, bool& found)
{
// Basic sequential search bool found = false;
 int i;
 found = false;
 for (i = 0; i < size; i++)
 {
  if (key == array[i])
  break;
  }
  if (i < size)
  {
   found = true;
   loc = i;
  }
}
```

```cpp
#include <iostream>
using namespace std;
int main()
{
  int array[100], n, key, loc;
  bool found;
  cout << "enter the dim of matrix:  ";
  cin >> n;
  Read1D(array, n);
  cout << "enter the value to be searched  " << endl;
  cin >> key;
  Seq_search(array, n, key, loc, found);
  if (found) {
    cout << "Its found" << endl;
    cout << " The location is   " << loc+1 << endl;
  }
  else
  cout << "Its not found" << endl;

}
```

2) Write a program to read two integer arrays A[10] and B[10]. Then, compute the following operations:
1- C=2A+3B+X   2- E= A ^ B

```
#include <iostream>
using namespace  std;
void Read1D(int[], int, char);
void Com_C(int[], int[], int[], int, int);
void print1D(int[], int);
void Inst(int[], int[], int[], int, int&);
bool boolsearch(int[], int, int);

void main()
{
   int X, m, A[10], B[10], C[10], E[10];
   Read1D(A, 10, 'A');
   Read1D(B, 10, 'B');
   cout << "enter X: "<< endl;
   cin >> X;
   Com_C(A, B, C, X, 10);
   cout << " The result of 2A+3B+X   " << endl;
   print1D(C, 10);
   Inst(A, B, E, 10, m);
   cout << " The result of A ^ B" << endl;
   print1D(E, m);

}
void Read1D(int A[10], int n, char M)
{
   for (int i = 0;i < n;i++)
   {
      cout << "enter"<<M<< "[" << i << "]" << endl;
      cin >> A[i];
   }
}
```

```
void Com_C(int A[], int B[], int C[], int X, int n)
{
    for (int i = 0;i < n;i++)
    {
        C[i] = 2 * A[i] + 3 * B[i] + X;
    }
}
void print1D(int A[10], int n)
{
    for (int i = 0;i < n;i++)
    {
        cout << A[i] << endl;
    }
}
void Inst(int A[10], int B[10], int E[10], int n, int& m)
{
    int X;
    m = 0;
    for (int i = 0; i < n; i++)
    {
        X = B[i];
        if(boolsearch(A, 10, X) == true)
        {
            E[m] = X;
            m++;
        }
    }
}
bool boolsearch(int A[10], int n, int X)
{
    bool F = false;
    for (int i = 0; i < n; i++)
    {
        if(A[i] == X)
            F = true;
    }
    return F;
```

3) Write a program to read integer one dimension array A(10)and then compute the following operations:
1- Generate one dimension array B which includes all the positive even numbers of A .   2- C= A!.