Programming Fundamental

Week 9

Learning Outcomes:

After this lesson, students will be able to:

- To utilize parallel arrays
- define the term 'array'
- identify arrays in real world settings
- create arrays using manipulative objects
- draw arrays and write matching multiplication equations

Instructions

- Use proper indentation to make your programs readable.
- Use descriptive variables in your programs (Name of the variables should show their purposes)

Parallel Array

Introduction

- Multiple arrays of the same size such that i-th element of each array is closely related and all i-th elements together represent an object or entity. An example parallel array is two arrays that represent x and y co-ordinates of n points.
  Below is another example where we store the first name, last name and heights of 5 people in three different arrays.

```
first_name= [“Bones”,”welma”,”Frank”,”Han”,”Jack”];
last_name= [“Smith”,”Seger”,”Mathers”,”Solo”,“Jackles”];
height = [169,158,201,183,172];
```

Application:

Two of the most essential applications perform on an array or a record are searching and sorting.

Searching: Each index in a parallel array corresponds to the data belonging to the same entity in a record. Thus, for searching an entity based on a specific value of an attribute ,for example.
We need to find the name of a person having height >200 cms in the above example. Thus, we search for the index in the height array having value greater than 200. Now, once we have obtained the index, we print the values of the index in the first_name and last_name arrays. This way searching becomes an easy task in parallel array.

Sorting: Now, using the same fact that each index corresponds to data items in different arrays corresponding to the same entity. Thus, we sort all arrays based on the same criteria.
For example, in the above-displayed example, we need to sort the people in increasing order of their respective heights. Thus, when we swap the two heights, we even swap the corresponding values in other arrays using the same index.

But First, recall these concepts that you were taught in the earlier class.

Array Declaration

SYNTAX

```
Type_Name Array_Name[Declared_Size];
```

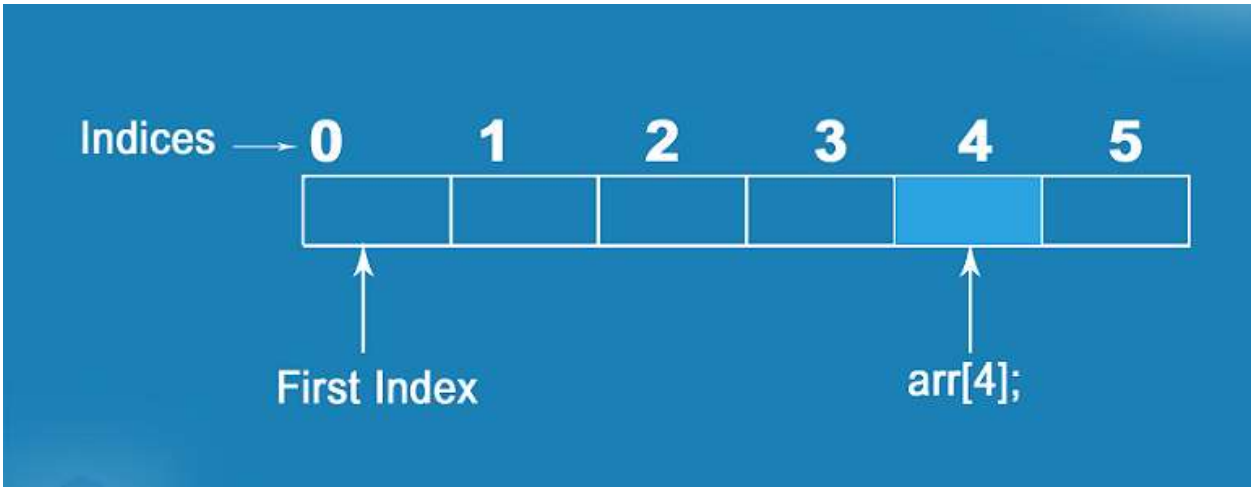
EXAMPLES

```
int bigArray[100];
double a[3];
double b[5];
char grade[10], oneGrade;
```

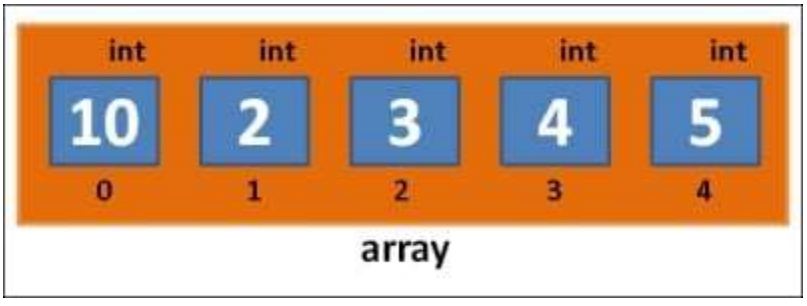
Declaration and initializing of different Type of Array:

Memory View of array with indexes:

```
a. int list[] = {18, 13, 14, 16};
b. int x[10] = {1, 7, 5, 3, 2, 8};
c. double y[5 ] = {2.0, 5.0, 8.0, 11.0, 14.0};
d. double lengths[] = {8.2, 3.9, 6.4, 5.7, 7.3};
e. int list[7] = {12, 13, , 14, 16, , 8};
f. string names[8] = {"John","Lisa", "Chris", "Katie"};
```



Array element with indexes view in Memory



Parallel Array :

Example #1

Write a program that declare and initialize two parallel arrays, show corresponding elements and their index one by one.

Solution

```
#include<iostream>
using namespace std;
int roll_no[] = {1, 2, 3, 4, 5};
    int age[] = {25, 20, 32, 30, 18};
void parallel_value()
{
     for(int i=0;i<5;i++)
    {
    cout<<"Roll number["<<i<<"]"<<roll_no[i]<<"\t";
    cout<<"Age["<<i<<"]"<<age[i]<<endl;

   }
}

int main()
 {

    parallel_value()  ;

    return 0;
}
```

The code produces the following output

```
Roll number[0]1 Age[0]25
Roll number[1]2 Age[1]20
Roll number[2]3 Age[2]32
Roll number[3]4 Age[3]30
Roll number[4]5 Age[4]18
```

Example #2

Write a program that declare two parallel array of roll_no and cgpa. Program should prompt the user to enter roll number and display corresponding cgpa.

Solution

```
#include<iostream>
using namespace std;
 int roll_no[] = {1, 2, 3, 4, 5};
float cgpa[] = {2.66, 2.78, 3.67, 3.70, 3.57};
void find_cgpa(int number)
{
    for(int i=0;i<5;i++)
    {
        if(roll_no[i]==number)
        {
    cout<<"Roll number="<<roll_no[i]<<"\t";
        cout<<"cgpa="<<cgpa[i]<<endl;
        }
    }
}
  int main()
   {


      int n;
      cout<<"enter rollnumber for cgpa"<<endl;
      cin>>n;
      find_cgpa(n);



      return 0;
  }
```

The code produces the following output

```
enter rollnumber for cgpa
3
Roll number=3   cgpa=3.67
```

Example #3:

Write a program that creates two parallel-sized arrays named "roll no" and "marks." The program should display the student's roll number who received the highest marks.

Solution

```cpp
#include<iostream>
using namespace std;
const int size=5;
int roll_no[] = {1, 2, 3, 4, 5};
int marks[] = {25, 20, 32, 30, 18};
int check_highest()
{
    int index,max=0;
    for (int i = 0; i < size; i++)
     {
        if (marks[i] > max)
        {
            max = marks[i];
            index = i;
        }
    }
    return index;
}
int main() {
    int  index ;

    index=check_highest();
    cout<<"Roll no %d has highest marks"<< roll_no[index]<<endl;
    return 0;
}
```

The code produces the following output

Output

```
Roll no 3 has highest marks
```

Example #4:

Write a program that declares two arrays same size name as character array of code and float discount array. Each code will represent discount value. Program should prompt the user to enter total number item, price and code. Program will return discount value against code and will display total value of bill.

Solution

```
#include<iostream>
using namespace std;
char code[] = {'a', 'b', 'c', 'd','e'};
float discount[] = {.25,.10,.20,.30,.50};
int price;
int find_index_code(char c)
{

    for(int i=0;i<5;i++)
        {
            if(c==code[i])
            {
                return i;
            }
        }
        return -1;

}
```

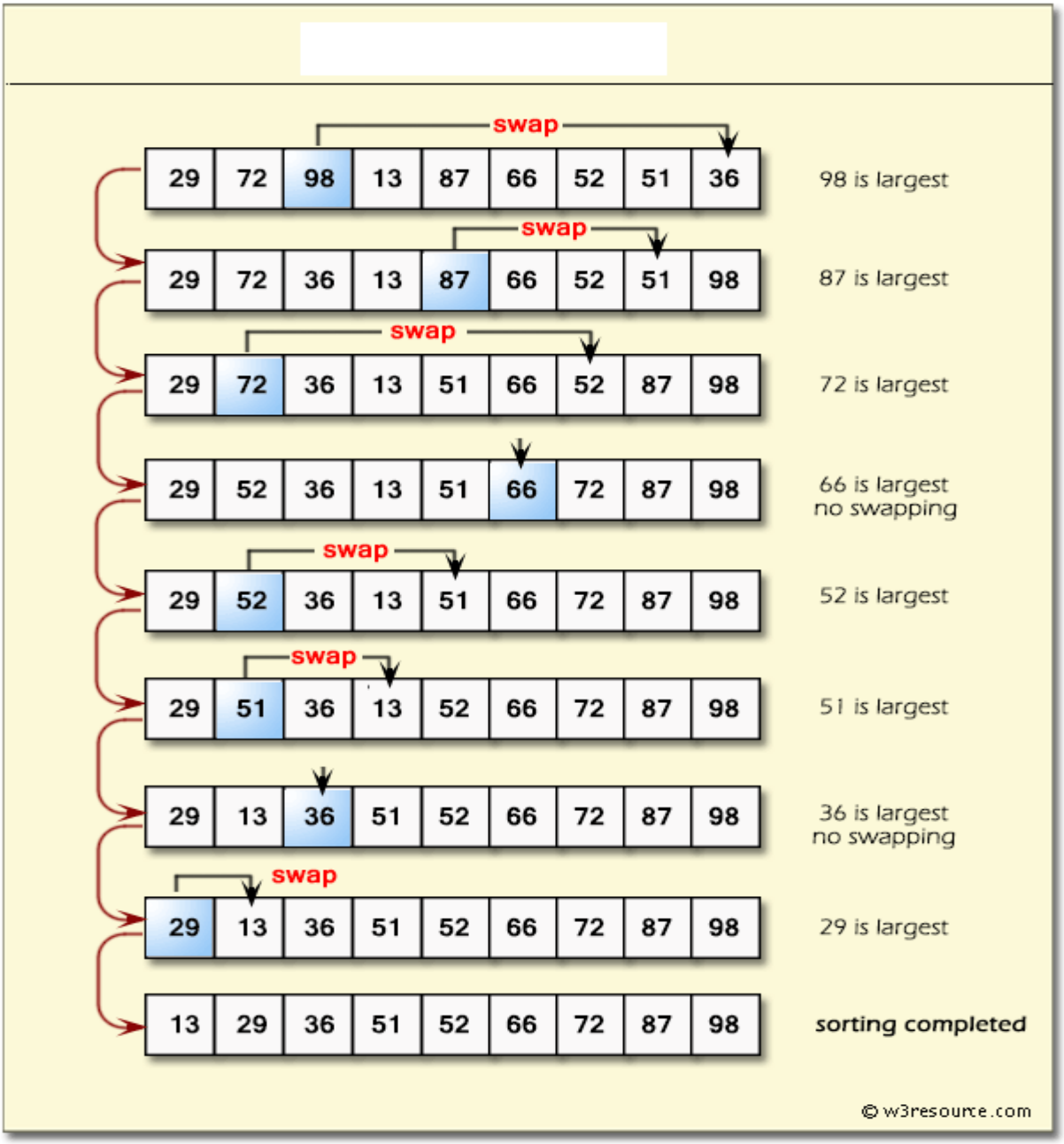
```
void calculate_bill(int index, int item, int price)
{
    int total;
    if(index>=0 && index<=4)
        {
            total=price*item-(price*item*discount[index]);
            cout<<"Total bill="<<total<<endl;       }
        else        {
            cout<<"invalid code entered";}
}

          int main() {

              int price,item,index;
              float total;
              char c;
              cout<<"enter total item";
              cin>>item;
              cout<<"enter price per item";
              cin>>price;
              cout<<"enter  code for discount";
              cin>>c;
              index=find_index_code(c);
              calculate_bill(index, item, price);

              return 0;
          }
```

The code produces the following output

```
enter total item 40
enter price per item 5
enter  code for discount b
Total bill=180
```

Sorting:

A Sorting Algorithm is used to rearrange a given array or list elements according to a comparison operator on the elements. The comparison operator is used to decide the new order of element in the respective data structure.



Example #5:

Write a program to read 5 people and their ages from the keyboard and save them into memory using arrays. Display the list starting with the oldest person to the youngest.

Solution

```
int find_largest_ages(int i)
{
    int large=-1;
    int index;
    for(int j=i;j<limit;j++)
    {
        if(ages[j]>large)
    {
        large=ages[j];
        index=j;
    }
    }
    return index;
}
void display()
{
    cout<<"The sorted list is :"<<endl;
    cout<<"Name"<<"\t"<<"Age"<<endl;
    for(int i=0;i<=4;i++){
    cout<<code[i]<<"\t"<<ages[i]<<endl;}

}
```

```
int main(){

input();
int temp;
int index;
for(int i=0;i<limit;i++ )
{
    index=find_largest_ages(i);
    temp=ages[i];
    ages[i]=ages[index];
    ages[index]=temp;

}

display();
cout<<endl;

return 0;}
```

The code produces the following output

```
A sample output of the program :

Enter 5 names:
Enter name and age for person 1 : Kim 23
Enter name and age for person 2 : Jason 18
Enter name and age for person 3 : Lee 14
Enter name and age for person 4 : Dan 34
Enter name and age for person 5 : May 8

The sorted list is :
Name Age
May 8
Lee 14
Jason 18
Dan 34
Press any key to continue.....
```

Note: Here we have only sorted the array of ages. You also have to sort the array of people's names according to the age.

Challenge#1:

Using Parallel Arrays to enter Names, Exercise Marks and Compute Average of Exercise Marks and Display

Challenge#2

Write a program that administers a basic addition quiz to the user. There should be ten questions. Each question is a simple addition problem such as 17 + 42 stored in a string array. The program should ask the user all ten questions. Each question contains three answer choices. Also create a parallel array that holds the correct answer to each question—A, B, or C. Display each question and verify that the user enters only A, B, or C as the answer—if not, keep prompting the user until a valid response is entered. get the user's answers.

After asking all the questions, the user should print each question again, with the user's answer. If the user got the answer right, the program should say so; if not, the program should give the correct answer. At the end, tell the user their score on the quiz, where each correct answer counts for ten points and each wrong answer should deduct 25% of 10.

The program should make separate functions, one to take the quiz, and one to grade the quiz and one to show all the correct answers and user’s choices. It can use global arrays.

Make a menu driven program. And take the quiz 3 times from the users (different users) and then show their scores in Descending Order.