FAST LHR (Lab9: PF Sec 3B)

Lab Instructor: Zaeem Yousaf| Kissa Tanvir

Course Instructor: Ms. Arooj Khalil

Learning Outcomes

1. Arrays Initialization
2. Arrays subscripts
3. Array's traversal
4. Functions/subroutines
5. passing arguments to functions
6. Arrays and functions as integrated
7. Character Array's Practice

Parameter and Argument

- Parameter: a place holder like algebric expression
  - Parameters are used at the time of defining functions
- Argument: actual data, whether literals or variables
  - Arguments are used at the time of calling function

Functions and Examples

void functions

- Write a void function which has no parameter

```
void function_name(){
    cout << "this is void function" << endl;
    cout << "it has no parameters" << endl;
}
```

```
We call this function this way
function_name()

if you notice
- Left hand side of function is empty
- it does not take any argument
```

void functions taking one argument

- Write a void function which has one parameter

```
void function_name(int p1){
    cout << "this is void function" << endl;
    cout << "but it has one int parameter: p1" << endl;
    cout << "argument is : " << p1 << endl;
}
```

```
We call this function this way
function_name(20)

if you notice
- Left hand side of function is empty
- it takes one argument
```

non void functions

```
// does not return any thing
// this is void function
void add_print(int x, int y){
    cout << x+y;
}

// but what if we don't want to print
// rather we want to store its result
// we will use return keyword
// non void function
int add_return(int x, int y){
    return x + y;
    cout << "this statement will be ignored " << endl;
    cout << "everthing after return is ignored" << endl;
}
```

```
// call void
add_print(10,20);
result: 30

// call non void
int var = add_return(10,20);
//your result goes inside the variable
// do whatever you want
```

Arrays & Examples

Declaration

```
// initialize the array of the size of 10
int array_name[10]; // now this array can hold upto 10 integers

float array_name[5]; // now this array can hold upto 10 real numbers

char array_name[5]; // now this array can hold upto 5 characters
```

Subscripting or Accessing element

```
int int_array_name[10];
// by defualt, elements of int arrays are
```

```
// [0,0,0,0,0,0,0,0,0,0]
int_array_name[0] = 10;
// now it will become
//[10,0,0,0,0,0,0,0,0,0]

int_array_name[9] = 20;
[10,0,0,0,0,0,0,0,0,20]

char char_array_name[5]; // now this array can hold upto 5 characters
char_array_name[0] = "z";
char_array_name[1] = "a";
char_array_name[2] = "e";
char_array_name[3] = "e";
char_array_name[4] = "m";
// now char array becomes
['z','z','e','e','m'] or in other words:  "zaeem"
```

Q1 (10 marks)

Words counting

```
#include <iostream>
using namespace std;
int countWords(char paragraph[], int sizeOfParagraph){
    // do your stuff here
    // make changes here so that it can count words correctly.
    // ignore fullstop if there is any

    // you can remove this for loop and start working from scratch
    for (int i = 0; i < sizeOfParagraph; ++i) {
        cout << paragraph[i];
    }
    cout << endl;
    // return number of words instead of zero
    return 0;
}

int main(int argc, char *argv[])
{
    char p1[] = "This is first paragraph with no space at start and no space at end";
    char p2[] = " This is first paragraph with one space at start and no space at end";
    char p3[] = " This is first paragraph with one space at start and one space at end ";
    char p4[] = "    this is a  sparse     paragraph     ";
    char p5[] = "    it is   multiline paragraph \n second line of paragraph ";

    int words1 = countWords(p1,sizeof(p2));
    cout << words1;

    int x;
    cin >> x;
    return 0;
}
```

Quiz Practice

```
//---------------------- q1(a)
char array1[] = "";
```

```
cout << sizeof(array1); // 1  but why? as it is empty
char array2[] = "1";
cout << sizeof(array2); // 2

//---------------------- q1(b)
// when array has same scope
// we can count its size using sizeof operator
// e.g
void anato(char array[]){
    cout << sizeof(array); // it will give wrong result
    // we cannot determine the size of an array using sizeof operator
    // that's why we have to send its size as well
}

//---------------------- q1(c)
char array3[2] = "12"; // what is wrong here

//---------------------- q1(d)
char array4[2] = {'1','2'};
cout << array4; // can you predict the output?

//---------------------- q1(e)
char array5[3] = {'1','2', '\0'}; // how it is different from q1(d)
cout << array5; // can you predict the output now?
```

Q2 (10 marks)

Anato's Riddle (Merge two paragraph into third one)

```
#include <iostream>
using namespace std;

void merge(char array1[], char array2[], char array3[], int size1, int size2, int size3){
    // concatenate (merge) array1 and array2 into array3

}

int main(int argc, char *argv[])
{
    char p1[] = "This is first paragraph with no space at start and no space at end";
    int p1_size = sizeof(p1);

    char p2[] = " This is first paragraph with one space at start and no space at end";
    int p2_size = sizeof(p2);

    int p3_size = sizeof(p1) + sizeof(p2);

    char p3[p3_size -1 ]; // why -1?
    // because p1_size is one more than the actual
    // p2_size is greater than actual
    // there for subtract 1, it still has space for null char

    merge(p1,p2,p3,p1_size,p2_size,p3_size);
    cout << p3; // in case of wrong input, make sure you have inserted last char as \0

    return 0;
}
```

Quiz: Namika's bombardment

```
#include<iostream>
using namespace std;
void namikaza_bombardment(char array[], int wrong_size){
    for (i = 0; i < wrong_size; ++i) {
        array[i] = 'a'; // store only array's
    }
}

int main(int argc, char *argv[])
{
    char p1[10];
    namikaza_bombardment(p1,100); // run it several time
    cout << p1; // observe the result
    // why it is happening this way

    return 0;
}
```