FAST LHR (Lab8: 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

Parameter and Argument

Parameter: a place holder like algebric expression

Parameters are used at the time of de ning 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;  } |
| --- |

1

2

3

4

We call this function this way

function\_name()

if you notice

- Left hand side of function is empty

- it does not take any argument

1

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;  } |
| --- |

1

2

3

4

5

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 |
| --- |

2

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 (marks: 5)

print5sTable()

Write a void function with no paramter which prints the table of 5 e.g

5 x 1 = 5

5 x 2 = 10

...

5 x 10 = 50

Q2 (marks: 5)

seriesSum()

3

Write a non void function with no parameter which returns the sum of first 1000 numbers using a for loop inside it.

Then use it in main like this

int sum = series\_sum()

Q3 (marks: 5)

sumBetween(start, end)

Write a non void function with two parameters 'start' and 'end' this function will return sum between these two number inclusive. e.g

int sum = sumBetween(2,5) // 2 + 3 + 4 + 5 = 14

Q4 (marks: 5)

Write a subroutine/function which is void with no parameters This function/Subroutine will have an array of size 1000 - loop\_in\_function\_and\_function\_in\_loop()

at zeroth index: sum of all numbers between 2,5 inclusive at 1st index: sum of all numbers between 2,6 inclusive at 1st index: sum of all numbers between 2,7 inclusive at 1st index: sum of all numbers between 2,1000 inclusive

then print average of this array

example: Calling a function in loop

| void f1(int start, int end){  // here goes your function  int sum = 0;  for (i = start; i < end; ++i) {  sum = sum + i;  }  }  int f2(){  int array[1000];  for (i = 5; i < 1000; ++i) {  array[i] = f1(2,i)  }  } |
| --- |

4