FAST LHR (Lab12: PF Sec 3B)

Lab Instructor: Zaeem Yousaf| Kissa Tanvir

Course Instructor: Ms. Arooj Khalil

Learning Outcomes

- 2D Array traversal
- Functions/subroutines
- passing arguments to functions
- Arrays and functions as integrated
- 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

#+BEGIN$_{\text{SRC}}$ cpp / *initialize the array of the size of 10 int array$_{name}$[10];* / now this array can hold upto 10 integers

float array$_{\text{name}}$[5]; // now this array can hold upto 10 real numbers .. char array$_{\text{name}}$[.]; // now this array can hold upto 5 characters . #+END$_{\text{SRC}}$.

Subscripting or Acces.ing element

.+ATTR$_{\text{LATEX}}$: :option. frame=single ..BEGIN$_{\text{SRC}}$ cpp . . . .. int$_{\text{arrayname.1}}$.]; . / *b..defualt, eleme..s of int.arrays are* / [0,0,0,0,0,0,0,0,0,0] . int$_{\text{arr.yname}}$[0] = 10; / *now it will become* /[10,0,0,0,0,0,0,0,0,0]

int$_{\text{arrayname}}$[9] = 20; [10,0,0,0,0,0,0,0,0,20]

char char$_{\text{arrayname}}$[5]; // now this array can hold upto 5 characters char$_{\text{arrayname}}$[0] = "z"; char$_{\text{arrayname}}$[1] = "a"; char$_{\text{arrayname}}$[2] = "e"; char$_{\text{arrayname}}$[3] = "e"; char$_{\text{arrayname}}$[4] = "m"; // now char array becomes ['z','z','e','e','m'] or in other words: "zaeem" #+END$_{\text{SRC}}$

2D Array

```
#include <iostream>
using namespace std;
void print2DArray(int array[][5], int rows){
    // array's first [] can be left blank
```

```cpp
    // but second is compulsory e.g [5]
    for (int i = 0; i < rows; ++i) {
        for (int j = 0; j < 5; j++) {
            cout << array[i][j] << " ";
        }
        cout << "\n";
    }
    cout << "\n";
}
int main(int argc, char *argv[])
{
    // compiler will count 5 rows, so no need to write it
    int array1[][5] = {
        {1,2,3,4,5},
        {5,4,3,2,1},
        {6,7,8,9,10},
        {10,9,8,7,6},
        {10,0,0,2,1}
    };

    print2DArray(array1, 5); // 5 rows, column size already given
    // can compiler find column size as well?
    int array2[][5] = {
        {1,2,3,4,5},
        {5,4,3,2,1},
        {6,7,8,9,10},
        {10,9,8,7,6}
    };
    print2DArray(array2, 4); // 4 rows, column size already given a
    // compiler know that there are 3 rows
    int array3[][5] = {
        {1,2,3,4,5},
        {5,4,3,2,1},
        {6,7,8,9,10}
    };
    print2DArray(array3, 3); // 3 rows, column size already given
    return 0;
}
```

Q1 practice (previous)

Reverse each word without changing the order

```cpp
#include <iostream>
using namespace std;
void reverseSentence(char paragraph[], int sizeOfParagraph){
    // e.g: "Hello zaeem" -> "olleH meeaz";
    // do your stuff here

    // you can remove this for loop and start working from scratch
    // actually we wish that after execution of this function
    // paragraph will have reversed words

    for (int i = 0; i < sizeOfParagraph; ++i) {
        cout << paragraph[i];
    }
    cout << endl;
}
```

```
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 ";

    reverseSentence(p1,sizeof(p1));
    cout << p1;

    reversed = reverseSentence(p2,sizeof(p2));
    cout << p2;

    reverseSentence(p3,sizeof(p3));
    cout << p3;

    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 Practice (previous)

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

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

```
void merge(char matrix1[][100], char matrix2[][100], char array3[][100], int size1, int size2, int siz
    // imagine matrix one can have 50 lines of data
    // and matrix2 has 20 lines of data
    // then matrix 3 will have 70 lines of data in the matrix3
    // complete your code
}

int main(int argc, char *argv[])
{
    char mat1[][100] = {
        "This is first line with no space at start and no space at end",
        "This is second line with no space at start and no space at end",
        "third line with no space at start and no space at end"
    };

    char line1[] = "random text for mat2";
    char line2[] = "second line of random text for mat2";
    char line3[] = "third line of random text for mat2";

    char mat2[][100];
    mat2[0] = line1;
    mat2[1] = line2;
    mat2[2] = line3;

    int mat1_size = sizeof(mat1);
    int mat2_size = sizeof(mat2);

    int m3_size = mat1_size + mat2_size;

    char p3 [m3_size][100 ];

    merge(mat1, mat2, mat3);

    // use priint2DArray to print matrix 3
    // make any modification if necessary
    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;
}
```

Q3 character manipulation (previous)

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

void make_unity_matrix(int matrix[][10], int rows){
    // e.g make 10x10 matrix, just pass 10 rows
    // make a uity matrix
}

void change_diagonal_entry(int matrix[][10], int value, int rows){
    // change all diagonal entries to value
    // where this matrix is a square matrix
}

void swap_row(int matrix, int row1, int row2){
    // swap row1 of matrix with row2
}

void swap_col(int matrix[][10], int c1, int c1){
    // swap col1 with col2 of matrix
}

int main(){
    int array[][10];
    make_unity_matrix(array, 10);
    change_diagonal_entry(array,5, 10);
    swap_row(array, 2, 4);
    swap_col(array, 4, 7);

    // write print function
}
```

Q4 Task Anato's Cannon (previous)

Animation in 2D grid

```
Write a program which displays a grid with a boundary
place your gun in the center of box. then display a menu
in where user can select in which direction to fire.
display the built in animation until it striks any of the wall.
```

Example

```
+--------------------------+
|                          |
|                          |
|                          |
|                          |
|                          |
|           +  ->          |
|                          |
|                          |
|                          |
|                          |
|                          |
|                          |
|                          |
+--------------------------+

0: shot right
1: shot left
2: shot up
3: shot bottom
4: shot all direction
Enter your choice: 0

+--------------------------+
|                          |
|                          |
|           ^              |
|           |              |
|                          |
|     <-    +    ->        |
|                          |
|                          |
|           |              |
|           v              |
|                          |
|                          |
|                          |
+--------------------------+

0: shot right
1: shot left
2: shot up
3: shot bottom
4: shot all direction
Enter your choice: 4
```

Lab Task (Anato's Cannon Grid)

Animation in 2D grid

```
Write a program which displays 'two grid' instead of one (previous lab)
with a boundary. place your gun in the center of box.

Right grid is the reflection of left grid.

Rest is same from previous lab
```

Example

```
0: shot right
1: shot left
2: shot up
3: shot bottom
4: shot all direction
Enter your choice: 0

+--------------------------+   +----------------------------+
|                          |   |                            |
|                          |   |                            |
|                          |   |                            |
|                          |   |                           |
|                          |   |                           |
|           +  ->          |   |        <-    +            |
|                          |   |                           |
|                          |   |                           |
|                          |   |                           |
|                          |   |                           |
|                          |   |                           |
|                          |   |                           |
|                          |   |                           |
+--------------------------+   +---------------------------+


0: shot right
1: shot left
2: shot up
3: shot bottom
4: shot all direction
Enter your choice: 4
+-------------------------+   +----------------------------+
|                          |   |                           |
|                          |   |                           |
|           ^              |   |            ^              |
|           |              |   |            |              |
|                          |   |                           |
|     <-    +    ->        |   |      <-    +     ->       |
|                          |   |                           |
|                          |   |                           |
|           |              |   |            |              |
|           v              |   |            v              |
|                          |   |                           |
|                          |   |                           |
|                          |   |                           |
+--------------------------+   +---------------------------+

0: shot right
1: shot left
2: shot up
3: shot bottom
4: shot all direction
Enter your choice: 1
+--------------------------+   +---------------------------+
|                          |   |                           |
|                          |   |                           |
|                          |   |                           |
|                          |   |                           |
|                          |   |                           |
|<-         +              |   |            +            ->|
|                          |   |                           |
|                          |   |                           |
|                          |   |                           |
|                          |   |                           |
|                          |   |                           |
|                          |   |                           |
|                          |   |                           |
+--------------------------+   +---------------------------+
```

Algorithm & functions

There could be many approaches for this problem.
But I will hint the better one

sample Code

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

const int grid_width = 40;
const int grid_height = 20;
const int gap = 5; // gap between left and right grids
const int canvas_width = 2*grid_width + gap;

void move_bullet_forward(char direction, // 'u' upward, 'l' left, 'r' right, 'd' down
                         int &current_x, // curent col of bullet (initiall center)
                         int &current_y,// curent row of bullet (initiall center)
                         char grid[][grid_height]){
    if (direction == 'u') {
        grid[current_x][current_y-2] = grid[current_x][current_y-1];
        grid[current_x][current_y-1] = grid[current_x][current_y];

        grid[current_x][current_y-1] = "";
        grid[current_x][current_y] = "";

        // update only row, because col remains same while bullet moves up and down
        current_y ++;
    }else if (direction == 'd') {
        // implement
    }else if (direction == 'l') {
        // implement
    }else if (direction == 'r') {
        // implement
    }else if (direction == 'a') {
        // implement all direction
    }else{
        // invalid direction
    }
}

void simulate_bullet_movement(char direction,
                              int &current_x,
                              int &current_y,
                              char grid[][grid_width]){

    char canvas[grid_height][canvas_width];
    make_canvas(grid,canvas);
    print_canvas(canvas);

    // I think it is almost implemented
    if (direction == 'u') {
        // check if next move is not out of the grid
        while (current_y - 1 > 0) {
            move_bullet_forward('u', curent_x, current_y, grid);
            make_canvas(grid,canvas);
            print_canvas(canvas);

        }
    }else if (direction == 'd') {
        // check if next move is not out of the grid
        while (current_y + 1 < grid_height) {
            move_bullet_forward('d', curent_x, current_y, grid);
            make_canvas(grid,canvas);
            print_canvas(canvas);

        }
    }else if (direction == 'l') {
        // check if next move is not out of the grid
        while (current_x - 1 > 0) {
            move_bullet_forward('l', curent_x, current_y, grid);
            make_canvas(grid,canvas);
            print_canvas(canvas);

        }
    }else if (direction == 'r') {
        // check if next move is not out of the grid
        while (current_x + 1 < grid_width) {
            move_bullet_forward('r', curent_x, current_y, grid);
            make_canvas(grid,canvas);
            print_canvas(canvas);

        }
    }else if (direction == 'a') {
        // check if next move is not out of the grid
        while (current_x + 1 < grid_width) {
            move_bullet_forward('a', curent_x, current_y, grid);
            make_canvas(grid,canvas);
            print_canvas(canvas);

        }
    }else{
        cout << "trying to go outside of the grid\n";
        break;
    }
}

void make_canvas(const char grid[][grid_width], char canvas[][canvas_width]){
    // use nested for loop to transfer left and reflection into canvas
    // this should result in anato's 2 cannons side by side.
    char right_grid[grid_height][grid_width];
    take_reflection(grid,right_grid);

    // fill canvas
}

void print_canvas(const char canvas[][canvas_width]){
    // use nested for loop to print a canvas
    // this should result in anato's 2 cannons side by side.
}


void take_reflection(const char left_grid[][grid_height], char right_grid[][grid_height]){
    // copy left gird into right grid in such a way
    // that it becomes a reflection in y-axis

    // i have used const before left_grid
    // because i don't want to change it even accidently

}

void make_grid(char grid[][grid_width]){
    // store all character of grid
    // place + in the center which is our cannon
    // you even no need to put '\0' at the end of each line as there
    // if you will put '\0' at the end of each line then you will
    // have to use nested for loop to check if this grid has been
    // correctly populated
}

void display_menu(){
    // display a menu
}

int main(){
    char left_grid[grid_width][grid_height];
    char right_grid[grid_width][grid_height];
    int current_x = grid_width/2; // correct it according to your need
    int current_y = grid_height/2; // correct it according to your need

    // this canvas will hold both grids
    char canvas[canvas_width][grid_height];

    make_grid(left_grid); // now this left grid has all symbols

    display_menu();

    while(1){
        display_menu();
        int choice = 0;

        if (choice == 0) {
            simulate_bullet_movement('r', current_x, current_y, grid);
        }else  if (choice == 1) {
            simulate_bullet_movement('l', current_x, current_y, grid);
        }else  if (choice == 2) {
            simulate_bullet_movement('d', current_x, current_y, grid);
        }else  if (choice == 3) {
            simulate_bullet_movement('d', current_x, current_y, grid);
        }else  if (choice == 4) {
            simulate_bullet_movement('a', current_x, current_y, grid);
        }
    }
}
```