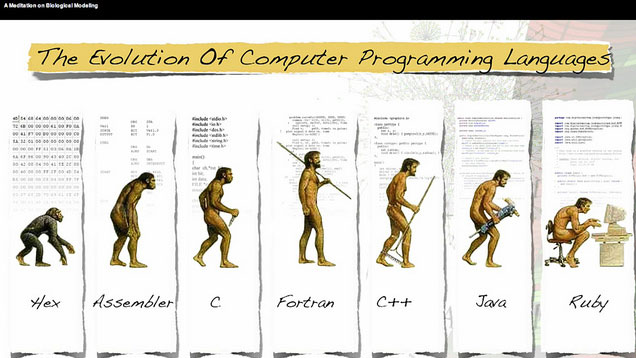
***COP2270***

***M/W***

***Spring 2017-2018***



***Professor: Yassin Raef***

***Anaisy Garcia***

***Chapter 6 Homework/ Exercises***

***One problem per page Please***

***Figure 6-3***

|  |
| --- |
| ***Commands*** |
| ***// Fig. 6.3: fig06\_03.c***  ***// Initializing the elements of an array to zeros.***  ***#include <stdio.h>***  ***// function main begins program execution***  ***int main(void)***  ***{***  ***int n[5]; // n is an array of five integers***  ***// set elements of array n to 0***  ***for (size\_t i = 0; i < 5; ++i) {***  ***n[i] = 0; // set element at location i to 0***  ***}***  ***printf("%s%13s\n", "Element", "Value");***  ***// output contents of array n in tabular format***  ***for (size\_t i = 0; i < 5; ++i) {***  ***printf("%7u%13d\n", i, n[i]);***  ***}***  ***}*** |

|  |
| --- |
| ***Output*** |
|  |

***Figure 6-4***

|  |
| --- |
| ***Commands*** |
| ***// Fig. 6.4: fig06\_04.c***  ***// Initializing the elements of an array with an initializer list.***  ***#include <stdio.h>***  ***// function main begins program execution***  ***int main(void)***  ***{***  ***// use initializer list to initialize array n***  ***int n[5] = {32, 27, 64, 18, 95};***  ***printf("%s%13s\n", "Element", "Value");***  ***// output contents of array in tabular format***  ***for (size\_t i = 0; i < 5; ++i) {***  ***printf("%7u%13d\n", i, n[i]);***  ***}***  ***}*** |

|  |
| --- |
| ***Output*** |
|  |

***Figure 6-5***

|  |
| --- |
| ***Commands*** |
| ***// Fig. 6.5: fig06\_05.c***  ***// Initializing the elements of array s to the even integers from 2 to 10.***  ***#include <stdio.h>***  ***#define SIZE 5***  ***// funtion main begins program execution***  ***int main(void)***  ***{***  ***// symbolic constant SIZE can be used to specify array size***  ***int s[SIZE]; // array s has SIZE elements***  ***for (size\_t j = 0; j < SIZE; ++j) { // set the values***  ***s[j] = 2 + 2 \* j;***  ***}***  ***printf("%s%13s\n", "Element", "Value");***  ***// output contents of array s in tabular format***  ***for (size\_t j = 0; j < SIZE; ++j) {***  ***printf("%7u%13d\n", j, s[j]);***  ***}***  ***}*** |

|  |
| --- |
| ***Output*** |
|  |

***Figure 6-6***

|  |
| --- |
| ***Commands*** |
| ***// Fig. 6.6: fig06\_06.c***  ***// Computing the sum of the elements of an array.***  ***#include <stdio.h>***  ***#define SIZE 12***  ***// funtion main begins program execution***  ***int main(void)***  ***{***  ***// use an initializer list to initialize the array***  ***int a[SIZE] = {1, 3, 5, 4, 7, 2, 99, 16, 45, 67, 89, 45};***  ***int total = 0; // sum of array***  ***// sum contents of array a***  ***for (size\_t i = 0; i < SIZE; ++i) {***  ***total += a[i];***  ***}***  ***printf("Total of array element values is %d\n", total);***  ***}*** |

|  |
| --- |
| ***Output*** |
|  |

***Figure 6-7***

|  |
| --- |
| ***Commands*** |
| ***//Fig. 6.7: fig06\_07.c***  ***// Analyzing a student poll.***  ***#include <stdio.h>***  ***#define RESPONSES\_SIZE 40 // define array sizes***  ***#define FREQUENCY\_SIZE 11***  ***//function main begins program execution***  ***int main(void)***  ***{***  ***// initialize frequency counters to 0***  ***int frequency[FREQUENCY\_SIZE] = {0};***  ***// plce the survey responses in the responses array***  ***int responses[RESPONSES\_SIZE] = {1, 2, 6, 4, 8, 5, 9, 7, 8, 10, 1, 6, 3, 8, 6, 10, 3, 8, 2, 7, 6, 5, 7, 6, 8, 6, 7, 5, 6, 6, 5, 6, 7, 5, 6, 4, 8, 6, 8, 10};***  ***// for each answer, select value of an element of array responses***  ***// and use that value as an index in array frequency to***  ***// determine element to increment***  ***for(size\_t answer = 0; answer < RESPONSES\_SIZE; ++answer) {***  ***++frequency[responses[answer]];***  ***}***  ***// display results***  ***printf("%s%17s\n", "Rating", "Frequency");***  ***// output the frequencies in a tabular format***  ***for(size\_t rating = 1; rating < FREQUENCY\_SIZE; ++rating) {***  ***printf("%6d%17d\n", rating, frequency[rating]);***  ***}***  ***}*** |

|  |
| --- |
| ***Output*** |
|  |

***Figure 6-8***

|  |
| --- |
| ***Commands*** |
| ***// Fig. 6.8: fig06\_08.c***  ***// Displaying a histogram.***  ***#include <stdio.h>***  ***#define SIZE 5***  ***// function main begins program execution***  ***int main(void)***  ***{***  ***// use initializer list to initialize array n***  ***int n[SIZE] = {19, 3, 15, 7, 11};***  ***printf("%s%13s%17s\n", "Element", "Value", "Histogram");***  ***// for each element of array n, output a bar of the histogram***  ***for (size\_t i = 0; i < SIZE; ++i) {***  ***printf("%7u%13d ", i, n[i]);***  ***for (int j = 1; j <= n[i]; ++j) { // print one bar***  ***printf("%c", ‘\*’);***  ***}***  ***puts(""); // end a histogram bar with a newline***  ***}***  ***}*** |

|  |
| --- |
| ***Output*** |
|  |

***Figure 6-9***

|  |
| --- |
| ***Commands*** |
| ***// Fig. 6.9: fig06\_09.c***  ***// Roll a six-sided die 60,000,000 times***  ***#include <stdio.h>***  ***#include <stdlib.h>***  ***#include <time.h>***  ***#define SIZE 7***  ***// fuunction main begins program execution***  ***int main(void)***  ***{***  ***unsigned int frequency[SIZE] = {0}; // clear counts***  ***srand(time(NULL)); // seed random number generator***  ***// roll die 60,000,000 times***  ***for (unsigned int roll = 1; roll <= 60000000; ++roll) {***  ***size\_t face = 1 + rand() % 6;***  ***++frequency[face]; // replaces entire switch of Fig. 5.12***  ***}***  ***printf("%s%17s\n", "Face", "Frequency");***  ***// output frequency elements 1-6 in tabular format***  ***for (size\_t face = 1; face < SIZE; ++face) {***  ***printf("%4d%17d\n", face, frequency[face]);***  ***}***  ***}*** |

|  |
| --- |
| ***Output*** |
|  |

***Figure 6-10***

|  |
| --- |
| ***Commands*** |
| ***// Fig. 6.10: fig06\_10.c***  ***// Trearing character arrays as strings.***  ***#include <stdio.h>***  ***#define SIZE 20***  ***// function main begins program execution***  ***int main(void)***  ***{***  ***char string1[SIZE]; // reserves 20 characters***  ***char string2[] = "string literal"; // reserves 15 characters***  ***// read string from user into array string1***  ***printf("%s", "Enter a string (no longer than 19 characters): ");***  ***scanf("%19s", string1); // input no more than 19 characters***  ***// output strings***  ***printf("string1 is: %s\nstring2 is: %s\n"***  ***"string1 with spaces between characters is:\n",***  ***string1, string2);***  ***// output characters until null character us reached***  ***for (size\_t i = 0; i < SIZE && string1[i] != '\0'; ++i) {***  ***printf("%c ", string1[i]);***  ***}***  ***puts("");***  ***}*** |

|  |
| --- |
| ***Output*** |
|  |

***Figure 6-11***

|  |
| --- |
| ***Commands*** |
| ***// Fig. 6.11: fig06\_11.c***  ***// Static arrays are initialized to zero if not explicitly initialized.***  ***#include <stdio.h>***  ***void staticArrayInit(void); // function prototype***  ***void automaticArrayInit(void); // function prototype***  ***// function main begins program execution***  ***int main(void)***  ***{***  ***puts("First call to each funtion:");***  ***staticArrayInit();***  ***automaticArrayInit();***  ***puts("\n\nSecond call to each function:");***  ***staticArrayInit();***  ***automaticArrayInit();***  ***}***  ***// function to demonstrate a static local array***  ***void staticArrayInit(void)***  ***{***  ***// initializes elements to 0 before the function is called***  ***static int array1[3];***  ***puts("\vValues on entering staticArrayInit:");***  ***// output contents of array1***  ***for (size\_t i = 0; i <= 2; ++i) {***  ***printf("array1[%u] = %d ", i, array1[i]);***  ***}***  ***puts("\nValues on existing staticArrayInit:");***  ***// modify and output contents of array1***  ***for (size\_t i = 0; i <= 2; ++i) {***  ***printf("array1[%u] = %d ", i, array1[i] += 5);***  ***}***  ***}***  ***// function to demonstrate an automatic local array***  ***void automaticArrayInit(void)***  ***{***  ***// initializes elements each time function is called***  ***int array2[3] = {1, 2, 3};***  ***puts("\n\nValues on entering automaticArrayInit:");***  ***// output contents of array2***  ***for (size\_t i = 0; i <= 2; ++i){***  ***printf("array2[%u] = %d ", i, array2[i]);***  ***}***  ***puts("\nValues on exiting automaticArrayInit:");***  ***// modify and output contents of array2***  ***for (size\_t i = 0; i <= 2; ++i) {***  ***printf("array2[%u] = %d ", i, array2[i] += 5);***  ***}***  ***}*** |

|  |
| --- |
| ***Output*** |
|  |

***Figure 6-12***

|  |
| --- |
| ***Commands*** |
| ***// Fig. 6.12: fig06\_12.c***  ***// Array name is the same as the address of the array's first element.***  ***#include <stdio.h>***  ***// function main begins program execution***  ***int main(void)***  ***{***  ***char array[5]; // define an array of size 5***  ***printf(" array = %p\n&array[0] = %p\n &array = %p\n",***  ***array, &array[0], &array);***  ***}*** |

|  |
| --- |
| ***Output*** |
|  |

***Figure 6-13***

|  |
| --- |
| ***Commands*** |
| ***// Fig. 6.13: fig06\_13.c***  ***// Passing arrays and individual array elements to functions.***  ***#include <stdio.h>***  ***#define SIZE 5***  ***// function prototypes***  ***void modifyArray(intb[], size\_t size);***  ***void modifyElement(int e);***  ***// function main begins program execution***  ***int main (void)***  ***{***  ***int a[SIZE] = {0, 1, 2, 3, 4}; // initialize array a***  ***puts("Effects of passing entire array by reference:\n\nThe values pf the original array are:");***  ***// output original array***  ***for(size\_t i = 0; i < SIZE; ++i) {***  ***printf("%3d", a[i]);***  ***}***  ***puts(""); // outputs a newline***  ***modifyArray(a, SIZE); // pass array a to modifyArray by reference***  ***puts("The values of the modified array are:");***  ***// output modified array***  ***for (size\_t i = 0; i < SIZE; ++i) {***  ***printf("%3d", a[i]);***  ***}***  ***// output value of a[3]***  ***printf("\n\nEffects of passing array element by value:\n\nThe value of a[3] is %d\n", a[3]);***  ***modifyElement(a[3]); // pass array element a[3] by value***  ***// output value of a[3]***  ***printf("The value of a[3] is %d\n", a[3]);***  ***}***  ***// in function modifyArray, "b" points to the original array "a" in memory***  ***void modifyArray(int b[], size\_t size)***  ***{***  ***// multiply each array element by 2***  ***for (size\_t j = 0; j < size; ++j) {***  ***b[j] \*= 2; // actually modifies original array***  ***}***  ***}***  ***// in function modifyElement, "e" is a local copy of array element a[3] passed from main***  ***void modifyElement(int e)***  ***{***  ***// multiply parameter by 2***  ***printf("Value in modifyElement is %d\n", e \*= 2);***  ***}*** |

|  |
| --- |
| ***Output*** |
|  |

***Figure 6-15***

|  |
| --- |
| ***Commands*** |
| ***// Fig. 6.15: fig06\_15.c***  ***// Sorting an array's values into accending order.***  ***#include <stdio.h>***  ***#define SIZE 10***  ***// function main begins program execution***  ***int main(void)***  ***{***  ***// initialize a***  ***int a[SIZE] = {2, 6, 4, 8, 10, 12, 89, 68, 45, 37};***  ***puts("Data items in original order");***  ***// output original array***  ***for (size\_t i = 0; i < SIZE; ++i) {***  ***printf("%4d", a[i]);***  ***}***  ***// bubble sort***  ***// loop control number of passed***  ***for (unsigned int pass = 1; pass < SIZE; ++pass) {***  ***// loop to control number of passes***  ***for (size\_t i = 0; i < SIZE - 1; ++i) {***  ***// compare adjacent elements and swap them if first***  ***// element is greater than second element***  ***if (a[i] > a[i + 1]) {***  ***int hold = a[i];***  ***a[i] = a[i+ 1];***  ***a[i + 1] = hold;***  ***}***  ***}***  ***}***  ***puts("\nData items in ascending order");***  ***// output sorted array***  ***for (size\_t i = 0; i < SIZE; ++i) {***  ***printf("%4d", a[i]);***  ***}***  ***puts("");***  ***}*** |

|  |
| --- |
| ***Output*** |
|  |

***Figure 6-16***

|  |
| --- |
| ***Commands*** |
| ***// Fig. 6.16.c***  ***// Survey data analysis with arrays:***  ***// computing the mean, median, and mode of data.***  ***#include <stdio.h>***  ***#define SIZE 99***  ***// function prototype***  ***void mean(const unsigned int answer[]);***  ***void median(unsigned int answer[]);***  ***void mode(unsigned int freq[], unsigned const int answer[]);***  ***void bubbleSort(unsigned int a[]);***  ***void printArray(unsigned const int a[]);***  ***// function main begins program execution***  ***int main(void)***  ***{***  ***unsigned int frequency[10] = {0}; // initialize array frequency***  ***// initialize array response***  ***unsigned int response[SIZE] =***  ***{6, 7, 8, 9, 8, 7, 8, 9, 8, 9,***  ***7, 8, 9, 5, 9, 8, 7, 8, 7, 8,***  ***6, 7, 8, 9, 3, 9, 8, 7, 8, 7,***  ***7, 8, 9, 8, 9, 8, 9, 7, 8, 9,***  ***6, 7, 8, 7, 8, 7, 9, 8, 9, 2,***  ***7, 8, 9, 8, 9, 8, 9, 7, 5, 3,***  ***5, 6, 7, 2, 5, 3, 9, 4, 6, 4,***  ***7, 8, 9, 6, 8, 7, 8, 9, 7, 8,***  ***7, 4, 4, 2, 5, 3, 8, 7, 5, 6,***  ***4, 5, 6, 1, 6, 5, 7, 8, 7};***  ***// process responses***  ***mean(response);***  ***median(response);***  ***mode(frequency, response);***  ***}***  ***// calculate average of all response values***  ***void mean(const unsigned int answer[])***  ***{***  ***printf("%s\n%s\n%s\n", "\*\*\*\*\*\*\*\*", " Mean", "\*\*\*\*\*\*\*\*");***  ***unsigned int total = 0; // variable to hold sum of array elements***  ***// total response values***  ***for (size\_t j = 0; j < SIZE; ++j) {***  ***total += answer[j];***  ***}***  ***printf("The mean is the average value of the data\n"***  ***"items. The mean is equal to the total of\n"***  ***"all the data items divided by the number\n"***  ***"of data items (%u). The mean value for\n"***  ***"this run is: %u / %u = %.4f\n\n",***  ***SIZE, total, SIZE, (double) total / SIZE);***  ***}***  ***// sort array and determine median element's value***  ***void median(unsigned int answer[])***  ***{***  ***printf("\n%s\n%s\n%s\n%s",***  ***"\*\*\*\*\*\*\*\*", " Median", "\*\*\*\*\*\*\*\*",***  ***"The unsorted array of responses is");***  ***printArray(answer); // output unsorted array***  ***bubbleSort(answer); // sort array***  ***printf("%s", "\n\nThe sorted array is");***  ***printArray(answer); // output sorted array***  ***// display median element***  ***printf("\n\nThe median is element %u of\n"***  ***"the sorted %u element array.\n"***  ***"For this run the median is %u\n\n",***  ***SIZE / 2, SIZE, answer[SIZE / 2]);***  ***}***  ***// determine most frequent response***  ***void mode(unsigned int freq[], const unsigned int answer[])***  ***{***  ***printf("\n%s\n%s\n%s\n", "\*\*\*\*\*\*\*\*", " Mode", "\*\*\*\*\*\*\*\*");***  ***// initialize frequencies to 0***  ***for (size\_t rating = 1; rating <= 9; ++rating) {***  ***freq[rating] = 0;***  ***}***  ***//summarize frequencies***  ***for(size\_t j = 0; j < SIZE; ++j) {***  ***++freq[answer[j]];***  ***}***  ***// output headers for result columns***  ***printf("%s%11s%19s\n\n%54s\n%54s\n\n",***  ***"Response", "Frequency", "Histogram",***  ***"1 1 2 2", "0 5 0 5");***  ***// output results***  ***unsigned int largest = 0; // represent largest frequency***  ***unsigned int modeValue = 0; // represents most frequent response***  ***for (unsigned int rating = 1; rating <= 9; ++rating) {***  ***printf("%8u%11u ", rating, freq[rating]);***  ***// keep track of mode value and largest frequency value***  ***if (freq[rating] > largest) {***  ***largest = freq[rating];***  ***modeValue = rating;***  ***}***  ***// output histogram bar respresenting frequency value***  ***for (unsigned int h = 1; h <= freq[rating]; ++h) {***  ***printf("%s", "\*");***  ***}***  ***puts(""); // being new line of output***  ***}***  ***// display the mode value***  ***printf("\nThe mode is the most frequent value.\n"***  ***"For this run the mode is %u which occured"***  ***" %u times.\n", modeValue, largest);***  ***}***  ***// fnction that sorts an array with bubble sort algorithm***  ***void bubbleSort(unsigned int a[])***  ***{***  ***// loop to control number of passes***  ***for (unsigned int pass = 1; pass < SIZE; ++pass) {***  ***// loop to control number of comparisons per pass***  ***for (size\_t j = 0; j < SIZE - 1; ++j) {***  ***// swap elements if out of order***  ***if (a[j] > a[j + 1]) {***  ***unsigned int hold = a[j];***  ***a[j] = a[j + 1];***  ***a[j + 1] = hold;***  ***}***  ***}***  ***}***  ***}***  ***// output array contents (20 values per row)***  ***void printArray(const unsigned int a[])***  ***{***  ***// output array contents***  ***for (size\_t j = 0; j < SIZE; ++j) {***  ***if (j % 20 == 0) { // begin new line every 20 values***  ***puts("");***  ***}***  ***printf("%2u", a[j]);***  ***}***  ***}*** |

|  |
| --- |
| ***Output*** |
|  |

***Figure 6-18***

|  |
| --- |
| ***Commands*** |
| ***// Fig. 6.18: fig06\_18.c***  ***// Linear search of an array.***  ***#include <stdio.h>***  ***#define SIZE 100***  ***// function prototype***  ***size\_t linearSearch(const int array[], int key, size\_t size);***  ***// function main begins program execution***  ***int main(void)***  ***{***  ***int a[SIZE]; // create array a***  ***// create some data***  ***for (size\_t x = 0; x < SIZE; ++x) {***  ***a[x] = 2 \* x;***  ***}***  ***printf("Enter integer search key: ");***  ***int searchKey; // value to locate in array a***  ***scanf("%d", &searchKey);***  ***// attempt to locate searchKEy in array a***  ***size\_t index = linearSearch(a, searchKey, SIZE);***  ***// display results***  ***if (index != -1) {***  ***printf("Found value at index %d\n", index);***  ***}***  ***else {***  ***puts("Value not found");***  ***}***  ***}***  ***// compare key to every element of array until the location is found***  ***// or until the end of array is reacher; return index of element***  ***// if key is found or -1 if key is not found***  ***size\_t linearSearch(const int array[], int key, size\_t size)***  ***{***  ***// loop through array***  ***for (size\_t n = 0; n < size; ++n) {***  ***if (array[n] == key) {***  ***return n; // return location of key***  ***}***  ***}***  ***return -1; // key not found***  ***}*** |

|  |
| --- |
| ***Output*** |
|  |

***Figure 6-19***

|  |
| --- |
| ***Commands*** |
| ***// Fig. 6.19.c***  ***// Binary search of a sorted array.***  ***#include <stdio.h>***  ***#define SIZE 15***  ***// function prototypes***  ***size\_t binarySearch(const int b[], int searchKey, size\_t low, size\_t high);***  ***void printHeader(void);***  ***void printRow(const int b[], size\_t low, size\_t moid, size\_t high);***  ***// function main begins program execution***  ***int main(void)***  ***{***  ***int a[SIZE]; // create array a***  ***// create data***  ***for (size\_t i = 0; i < SIZE; ++i) {***  ***a[i] = 2 \* i;***  ***}***  ***printf("%s", "Enter a number between 0 and 28: ");***  ***int key; // value to locate in array a***  ***scanf("%d", &key);***  ***printHeader();***  ***// search for key in array a***  ***size\_t result = binarySearch(a, key, 0, SIZE - 1);***  ***// display results***  ***if(result != -1) {***  ***printf("\n%d found at index %d\n", key, result);***  ***}***  ***else {***  ***printf("\n%d not found\n", key);***  ***}***  ***}***  ***// function to perform binary search of an array***  ***size\_t binarySearch(const int b[], int searchKey, size\_t low, size\_t high)***  ***{***  ***// loop until low index is greater than high index***  ***while (low <= high) {***  ***// determine middle element of subarray being searched***  ***size\_t middle = (low + high) / 2;***  ***// display subarray used in this loop iteration***  ***printRow(b, low, middle, high);***  ***// if searchKey matched middle element, return middle***  ***if (searchKey == b[middle]){***  ***return middle;***  ***}***  ***// if searchKey is less than middle element, set new high***  ***else if (searchKey < b[middle]) {***  ***high = middle - 1; // search low end of array***  ***} // if***  ***// if searchKey is greater than middle element, set new low***  ***else {***  ***low = middle + 1; // search high end of array***  ***}***  ***} // end while***  ***return -1; // searchKey not found***  ***}***  ***// print a header for the output***  ***void printHeader(void)***  ***{***  ***puts("\nIndices:");***  ***// output column head***  ***for (unsigned int i = 0; i < SIZE; ++i) {***  ***printf("%3u", i);***  ***}***  ***puts(""); // start new line of output***  ***// output line of - characters***  ***for (unsigned int i = 1; i <= 4 \* SIZE; ++i) {***  ***printf("%s", "-");***  ***}***  ***puts(""); // start new line output***  ***}***  ***// print one row of outpt showing the current***  ***// part of the array being processed.***  ***void printRow(const int b[], size\_t low, size\_t mid, size\_t high)***  ***{***  ***// loop through entire array***  ***for (size\_t i = 0; i < SIZE; ++i) {***  ***// display spaces if outside current subarray range***  ***if (i < low || i > high) {***  ***printf("%s"," ");***  ***}***  ***else if (i == mid) { // display middle element***  ***printf("%3d\*", b[i]); // mark middle value***  ***}***  ***else { // display other elements in subarray***  ***printf("%3d", b[i]);***  ***}***  ***}***  ***puts(""); // start new line of output***  ***}*** |

|  |
| --- |
| ***Output*** |
|  |

***Figure 6-21***

|  |
| --- |
| ***Commands*** |
| ***// Fig. 6.21: fig06\_21.c***  ***// Initializing multidimensional arrays.***  ***#include <stdio.h>***  ***void printArray(int a[] [3]); // function prototype***  ***// function main begins program execution***  ***int main(void)***  ***{***  ***int array1[2] [3] = {{1, 2, 3}, {4, 5, 6}};***  ***puts("Values in array1 by row are:");***  ***printArray(array1);***  ***int array2[2] [3] = {1, 2, 3, 4, 5};***  ***puts("Values in array2 by row are:");***  ***printArray(array2);***  ***int array3[2] [3] = {{1, 2}, {4}};***  ***puts("Values in arrray3 by row are:");***  ***printArray(array3);***  ***}***  ***// function to output array with two rows and three columns***  ***void printArray(int a[] [3])***  ***{***  ***// loop through rows***  ***for (size\_t i = 0; i <= 1; ++i) {***  ***// output column values***  ***for (size\_t j = 0; j <= 2; ++j) {***  ***printf("%d ", a[i] [j]);***  ***}***  ***printf("\n"); // start new line of output***  ***}***  ***}*** |

|  |
| --- |
| ***Output*** |
|  |

***Figure 6-22***

|  |
| --- |
| ***Commands*** |
| ***// Fig. 6.22: fig06\_22.c***  ***// Two-dimensional array manipulations***  ***#include <stdio.h>***  ***#define STUDENTS 3***  ***#define EXAMS 4***  ***// function prototype***  ***int minimum(const int grades[][EXAMS], size\_t pupils, size\_t tests);***  ***int maximum(const int grades[][EXAMS], size\_t pupils, size\_t tests);***  ***double average(const int setOfGrades[], size\_t tests);***  ***void printArray(const int grades[][EXAMS], size\_t pupils, size\_t tests);***  ***// function main begins program execution***  ***int main(void)***  ***{***  ***// initialize student grades for three students (rows)***  ***int studentGrades[STUDENTS][EXAMS] =***  ***{ { 77, 68, 86, 73 },***  ***{ 96, 87, 89, 78 },***  ***{ 70, 90, 86, 81 } };***  ***// output array studentGrades***  ***puts("The array is:");***  ***printArray(studentGrades, STUDENTS, EXAMS);***  ***// determine smallest and largest grade values***  ***printf("\n\nLowest grade; %d\nHighest grade: %d\n",***  ***minimum(studentGrades, STUDENTS, EXAMS),***  ***maximum(studentGrades, STUDENTS, EXAMS));***  ***// calculate average grade for each student***  ***for (size\_t student = 0; student < STUDENTS; ++student) {***  ***printf("The average grade for student %u is %.2f\n",***  ***student, average(studentGrades[student], EXAMS));***  ***}***  ***}***  ***// Find the maximum grade***  ***int minimum(const int grades[][EXAMS], size\_t pupils, size\_t tests)***  ***{***  ***int lowGrade = 100; // initialize to highest possible grade***  ***// loop through row of grades***  ***for (size\_t i = 0; i < pupils; ++i) {***  ***// loop through columns of grades***  ***for (size\_t j = 0; j < tests; ++j) {***  ***if (grades[i][j] < lowGrade) {***  ***lowGrade = grades[i][j];***  ***}***  ***}***  ***}***  ***return lowGrade; // return minimum grades***  ***}***  ***// Find the maximum grade***  ***int maximum(const int grades[][EXAMS], size\_t pupils, size\_t tests)***  ***{***  ***int highGrade = 0; // initialize to lowest possible grade***  ***// loop through rows of grades***  ***for (size\_t i = 0; i < pupils; ++i) {***  ***// loop through columns of grades***  ***for (size\_t j = 0; j < tests; ++j) {***  ***if (grades[i][j] > highGrade) {***  ***highGrade = grades[i][j];***  ***}***  ***}***  ***}***  ***return highGrade; // return maximum grade***  ***}***  ***// Determine the average grade for a particular student***  ***double average(const int setOfGrades[], size\_t tests)***  ***{***  ***int total = 0; // sum of test grades***  ***// total all grades for one student***  ***for (size\_t i = 0; i < tests; ++i) {***  ***total += setOfGrades[i];***  ***}***  ***return (double) total / tests; // average***  ***}***  ***// Print the array***  ***void printArray(const int grades[][EXAMS], size\_t pupils, size\_t tests)***  ***{***  ***// output column heads***  ***printf("%s", " [0] [1] [2] [3]");***  ***// output grades in tabular format***  ***for (size\_t i = 0; i < pupils; ++i) {***  ***// output label for row***  ***printf("\nstudentGrades[%u] ", i);***  ***// output grades for one student***  ***for (size\_t j = 0; j < tests; ++j) {***  ***printf("%-5d", grades[i][j]);***  ***}***  ***}***  ***}*** |

|  |
| --- |
| ***Output*** |
|  |

***Figure 6-23***

|  |
| --- |
| ***Commands*** |
| ***// Fig. 6.23: fig06\_23.c***  ***// Using variable-length arrays in C99***  ***#include <stdio.h>***  ***// Function prototypes***  ***void print1DArray(size\_t size, unsigned int array[size]);***  ***void print2DArray(unsigned int row, unsigned int col, unsigned int array[row][col]);***  ***int main(void)***  ***{***  ***printf("%s", "Enter size of a one-dimensional array: ");***  ***unsigned int arraySize; // size of 1-D array***  ***scanf("%d", &arraySize);***  ***unsigned int array[arraySize]; // declare 1-D variable length array***  ***printf("%s", "Enter number of rows and columns in a 2-D array: ");***  ***unsigned int row1, col1; // number of rows and columns in a 2-D array***  ***scanf("%d %d", &row1, &col1);***  ***unsigned int array2D1[row1][col1]; // declare 2-D variable length array***  ***printf("%s",***  ***"Enter number of rows and columns in another 2-D array: ");***  ***unsigned int row2, col2; // number of rows and columns in another 2-D array***  ***scanf("%d %d", &row2, &col2);***  ***unsigned int array2D2[row2][col2]; // declare 2-D variable length array***  ***// test sizeof operator on VLA***  ***printf("\nsizeof(array) yields array size of %d bytes\n",***  ***sizeof(array));***  ***// assign elements of 1-D VLA***  ***for (size\_t i = 0; i < arraySize; ++i) {***  ***array[i] = i \* i;***  ***}***  ***// assign elements of first 2-D VLA***  ***for (size\_t i = 0; i < row1; ++i) {***  ***for (size\_t j = 0; j < col1; ++j) {***  ***array2D1[i][j] = i + j;***  ***}***  ***}***  ***// assign elements of second 2-D VLA***  ***for (size\_t i = 0; i < row2; ++i) {***  ***for (size\_t j = 0; j < col2; ++j) {***  ***array2D2[i][j] = i + j;***  ***}***  ***}***  ***puts("\nOne-dimensional array:");***  ***print1DArray(arraySize, array); // pass 1-D VLA to function***  ***puts("\nFirst two-dimensional array:");***  ***print2DArray(row1, col1, array2D1); // pass 2-D VLA to function***  ***puts("\nSecond tow-dimensional array:");***  ***print2DArray(row2, col2, array2D2); // pass other 2-D VLA to function***  ***}***  ***void print1DArray(size\_t size, unsigned int array[size])***  ***{***  ***// output contents of array***  ***for (size\_t i = 0; i < size; i++) {***  ***printf("array[%d] = %d\n", i, array[i]);***  ***}***  ***}***  ***void print2DArray(unsigned int row, unsigned int col, unsigned int array[row][col])***  ***{***  ***// output contents of array***  ***for (size\_t i = 0; i < row; ++i) {***  ***for (size\_t j = 0; j < col; ++j) {***  ***printf("%5d", array[i][j]);***  ***}***  ***puts("");***  ***}***  ***}*** |

|  |
| --- |
| ***Output*** |
|  |

***Figure 6-10***

|  |
| --- |
| ***Commands*** |
|  |

|  |
| --- |
| ***Output*** |
|  |

***Figure 6-14***

|  |
| --- |
| ***Commands*** |
|  |

|  |
| --- |
| ***Output*** |
|  |

***Figure 6-15***

|  |
| --- |
| ***Commands*** |
|  |

|  |
| --- |
| ***Output*** |
|  |

***Figure 6-22***

|  |
| --- |
| ***Commands*** |
| ***//Fig. 6.22.c***  ***#include <stdio.h>***  ***void getSales(float[][5]);***  ***void printHeader(void);***  ***void printSales(float[][5]);***  ***int main(void)***  ***{***  ***float sales[4][5] = {0.0};***  ***getSales(sales);***  ***printHeader();***  ***printSales(sales);***  ***return 0;***  ***}***  ***void getSales(float sales[][5])***  ***{***  ***int salesPerson;***  ***int product;***  ***float value;***  ***printf("\nEnter the salepersons number, the product number, and the total dollar value of the products sold that day (-1 to end): ");***  ***scanf("%d", &salesPerson);***  ***while (salesPerson != -1) {***  ***scanf("%d%f", &product, &value);***  ***sales[salesPerson - 1][product - 1] += value;***  ***printf("\nEnter the salepersons number, the product number, and the total dollar value of the products sold that day (-1 to end): ");***  ***scanf("%d", &salesPerson);***  ***}***  ***}***  ***void printHeader(void)***  ***{***  ***printf("\n\n\tThe total sales for each salesperson and the total sales for each product are displayed.\n\n");***  ***printf("Sales-%30s\n", "Products");***  ***printf("person %8d%8d%8d%8d%8d%9s\n", 1, 2, 3, 4, 5, "Total");***  ***}***  ***void printSales(float sales[][5])***  ***{***  ***float totalSales;***  ***int i;***  ***int j;***  ***float productSales[5] = {0.0};***  ***for (i = 0; i < 4; i++) {***  ***totalSales = 0;***  ***printf("%3d ", i + 1);***  ***for (j = 0; j < 5; j++)***  ***{***  ***totalSales += sales[i][j];***  ***productSales[j] += sales[i][j];***  ***printf("%8.2f", sales[i][j]);***  ***}***  ***printf("%8.2f\n", totalSales);***  ***}***  ***printf("Total ");***  ***for (j = 0; j < 5; j++) {***  ***printf("%8.2f", productSales[j]);***  ***}***  ***printf("\n");***  ***}*** |

|  |
| --- |
| ***Output*** |
|  |