1. Write a program to declare and initialize variables of different data types (int, float, double, char) and print their sizes using the sizeof operator.

#include<stdio.h>

int main()

{

printf("Size of int: %d bytes\n", sizeof(int));

printf("Size of float: %d bytes\n", sizeof(float));

printf("size of double: %d bytes\n", sizeof(double));

printf("size of char: %d bytes\n", sizeof(char));

return 0;

}

1. Write a program to demonstrate the use of int, float, and char data types by performing arithmetic operations (addition, subtraction, multiplication, division) on variables of these types.

#include<stdio.h>

int main()

{

int num1;

float num2;

printf("Enter an number1: ");

scanf("%d", &num1);

printf("Enter an number2: ");

scanf("%f", &num2);

printf("\nSum of the two numbers is: %f", num1+num2);

printf("\nproduct of the two numbers is: %f", num1\*num2);

printf("\nsubtraction of the two numbers is: %f", num1-num2);

printf("\ndivision of the two numbers is: %f", num1/num2);

}

Sum of the two numbers is: 4.300000

product of the two numbers is: 4.600000

subtraction of the two numbers is: -0.300000

division of the two numbers is: 0.869565

1. Create a program to convert temperature from Fahrenheit to Celsius and vice versa. Use appropriate data types for the variables involved.

#include <stdio.h>

int main() {

float fahrenheit, celsius;

printf("Enter temperature in Fahrenheit: ");

scanf("%f", &fahrenheit);

// Convert Fahrenheit to Celsius

celsius = (fahrenheit - 32) \* 5 / 9;

printf("%.2f Fahrenheit is equal to %.2f Celsius\n", fahrenheit, celsius);

printf("Enter temperature in Celsius: ");

scanf("%f", &celsius);

// Convert Celsius to Fahrenheit

fahrenheit = (celsius \* 9 / 5) + 32;

printf("%.2f Celsius is equal to %.2f Fahrenheit\n", celsius, fahrenheit);

return 0;

}

1. Develop a program that calculates the total cost of items purchased at a store, including tax. Use appropriate data types for prices and quantities.

#include <stdio.h>

int main() {

float item1\_price, item2\_price, item3\_price;

int item1\_quantity, item2\_quantity, item3\_quantity;

float tax\_rate;

float subtotal, total;

// Get the prices and quantities of the items

printf("Enter the price of item 1: ");

scanf("%f", &item1\_price);

printf("Enter the quantity of item 1: ");

scanf("%d", &item1\_quantity);

printf("Enter the price of item 2: ");

scanf("%f", &item2\_price);

printf("Enter the quantity of item 2: ");

scanf("%d", &item2\_quantity);

printf("Enter the price of item 3: ");

scanf("%f", &item3\_price);

printf("Enter the quantity of item 3: ");

scanf("%d", &item3\_quantity);

// Calculate the subtotal

subtotal = (item1\_price \* item1\_quantity) + (item2\_price \* item2\_quantity) + (item3\_price \* item3\_quantity);

// Get the tax rate

printf("Enter the tax rate (in percentage): ");

scanf("%f", &tax\_rate);

// Calculate the total cost including tax

total = subtotal + (subtotal \* (tax\_rate / 100));

// Display the total cost

printf("Total cost including tax: $%.2f\n", total);

return 0;

}

1. Write a program to swap two numbers without using a temporary variable. Use appropriate data types for the variables involved.

#include <stdio.h>

void swapNumbers(int \*a, int \*b) {

\*a = \*a + \*b;

\*b = \*a - \*b;

\*a = \*a - \*b;

}

int main() {

int num1, num2;

printf("Enter the first number: ");

scanf("%d", &num1);

printf("Enter the second number: ");

scanf("%d", &num2);

printf("Before swapping: num1 = %d, num2 = %d\n", num1, num2);

swapNumbers(&num1, &num2);

printf("After swapping: num1 = %d, num2 = %d\n", num1, num2);

return 0;

}

1. Implement a program that converts a decimal number to binary representation. Make sure to handle both integer and fractional parts of the number.

#include <stdio.h>

void decimalToBinary(double num) {

// Extract the integer and fractional parts

int integerPart = (int)num;

double fractionalPart = num - integerPart;

// Convert the integer part to binary

printf("Integer part: ");

while (integerPart > 0) {

printf("%d", integerPart % 2);

integerPart /= 2;

}

printf("\n");

// Convert the fractional part to binary

printf("Fractional part: ");

while (fractionalPart > 0) {

if (fractionalPart \* 2 >= 1) {

printf("1");

fractionalPart = (fractionalPart \* 2) - 1;

} else {

printf("0");

fractionalPart \*= 2;

}

}

printf("\n");

}

int main() {

double decimalNumber = 10.625;

decimalToBinary(decimalNumber);

return 0;

}

1. Write a C program that takes the age and the number of books borrowed by a library member as input. Use the following criteria to determine the membership status: If the member is under 18 years old and has borrowed more than 2 books, display "Junior Member".If the member is between 18 and 60 years old (inclusive) and has borrowed less than or equal to 3 books, display "Regular Member".If the member is 60 years old or older and has borrowed less than or equal to 5 books, display "Senior Member". Otherwise, display "Membership Status: Unknown".

#include<stdio.h>

int main()

{

int age,book;

printf("Enter the age of : \n");

scanf("%d",&age);

printf("Enter the no of books borrowed of : \n");

scanf("%d",&book);

if(age<18 && book>2)

{

        printf("junior member\n");

    }

    if(18<age<=60 && book<=3)

    {

        printf("Regular member member\n");

    }

    if(age>60 && book<=5)

    {

        printf("senior member\n");

    }

}

1. Write a C program for the fast-food restaurant to handle customer orders. The program should display a menu of food items along with their prices. The menu should include items such as burgers, fries, drinks, etc. The program should prompt the customer to select an item by entering its corresponding number from the menu. Once the item is selected, the program should prompt the customer to enter the quantity of the selected item. Finally, the program should calculate and display the total bill amount.

#include <stdio.h>

int main() {

 int choice;

 int quantity;

 float price;

 printf("Menu:\n");

 printf("1. Burger - 100\n");

 printf("2. Pizza - 200\n");

 printf("3. Salad - 300\n");

 printf("4. Sandwich - 400\n");

 printf("5. Pasta - 500\n");

 printf("Enter your choice (1-5): ");

 scanf("%d", &choice);

 switch (choice) {

   case 1:

     price = 100;

     break;

   case 2:

     price = 200;

     break;

   case 3:

     price = 300;

     break;

   case 4:

     price = 400;

     break;

   case 5:

     price = 500;

     break;

   default:

     printf("Invalid choice!\n");

     return 0;

 }

 printf("Enter the quantity: ");

 scanf("%d", &quantity);

 float total = price \* quantity;

 printf("Total amount: $%.2f\n", total);

 return 0;

}

1. Write a program that takes a string as input and transforms it into a new string where each character is replaced by the next character in the alphabet (e.g., 'a' becomes 'b', 'b' becomes 'c', ..., 'z' becomes 'a').

#include <stdio.h>

#include <string.h>

void replaceCharacters(char\* str) {

int length = strlen(str);

for (int i = 0; i < length; i++) {

if (str[i] >= 'a' && str[i] <= 'z') {

if (str[i] == 'z') {

str[i] = 'a';

} else {

str[i] = str[i] + 1;

}

} else if (str[i] >= 'A' && str[i] <= 'Z') {

if (str[i] == 'Z') {

str[i] = 'A';

} else {

str[i] = str[i] + 1;

}

}

}

}

int main() {

char str[100];

printf("Enter a string: ");

fgets(str, sizeof(str), stdin);

replaceCharacters(str);

printf("Transformed string: %s", str);

return 0;

}

1. Write a program that generates a random number between 1 and 100. Prompt the user to guess the number. Keep prompting the user until they guess the correct number. Provide hints such as "Too high" or "Too low". Use a do-while loop for the guessing game.

#include <stdio.h>

#include <stdlib.h>

#include <time.h>

int main() {

int randomNumber, guess;

// Initialize random seed

srand(time(0));

// Generate random number between 1 and 100

randomNumber = rand() % 100 + 1;

do {

// Prompt the user to guess the number

printf("Guess the number (between 1 and 100): ");

scanf("%d", &guess);

// Check if the guess is too high or too low

if (guess > randomNumber) {

printf("Too high!\n");

} else if (guess < randomNumber) {

printf("Too low!\n");

}

} while (guess != randomNumber);

// Print a success message when the correct number is guessed

printf("Congratulations! You guessed the correct number.\n");

return 0;

}

1. Write a program that prompts the user to enter a positive integer and then prints the number in reverse order

#include <stdio.h>

int main() {

int num, reversedNum = 0, remainder;

printf("Enter a positive integer: ");

scanf("%d", &num);

while (num != 0) {

remainder = num % 10;

reversedNum = reversedNum \* 10 + remainder;

num /= 10;

}

printf("The number in reverse order is: %d\n", reversedNum);

return 0;

}

1. Fill in the **reverseArray** function to reverse the elements of the given array **arr** in-place.

#include <stdio.h>

void reverseArray(int arr[], int size) {

// TODO: Implement array reversal logic here

}

int main() {

   int arr[] = {1, 2, 3, 4, 5};

   int size = sizeof(arr) / sizeof(arr[0]);

  reverseArray(arr, size);

  printf("Reversed Array: ");

   for (int i = 0; i < size; i++) {

      printf("%d ", arr[i]);

   }

   return 0;

}

1. Fill in the **findMissingNumber** function to find the missing number in the given array **arr**, which contains consecutive integers starting from 1.

#include <stdio.h>

int findMissingNumber(int arr[], int size) {

   // TODO: Implement logic to find the missing number

   return -1;

}

int main() {

   int arr[] = {1, 2, 4, 5, 6};

   int size = sizeof(arr) / sizeof(arr[0]);

   int missingNumber = findMissingNumber(arr, size);

  printf("Missing Number: %d\n", missingNumber);

   return 0;

}

1. Fill in the **rotateArray** function to rotate the elements of the given array **arr** to the right by **k** positions.

#include <stdio.h>

void rotateArray(int arr[], int size, int k) {

   // TODO: Implement array rotation logic here

}



int main() {

   int arr[] = {1, 2, 3, 4, 5};

   int size = sizeof(arr) / sizeof(arr[0]);

   int k = 2; // Number of rotations

   rotateArray(arr, size, k);

  printf("Rotated Array: ");

   for (int i = 0; i < size; i++) {

      printf("%d ", arr[i]);

   }

   return 0}

1. Fill in the **findIntersection** function to find and print the intersection of two arrays **arr1** and **arr2**.

#include <stdio.h>

void findIntersection(int arr1[], int size1, int arr2[], int size2) {

// Iterate over each element in arr1

for (int i = 0; i < size1; i++) {

// Check if the current element exists in arr2

for (int j = 0; j < size2; j++) {

if (arr1[i] == arr2[j]) {

// Print the intersection element

printf("%d ", arr1[i]);

break;

}

}

}

}

int main() {

int arr1[] = {1, 2, 3, 4, 5};

int size1 = sizeof(arr1) / sizeof(arr1[0]);

int arr2[] = {4, 5, 6, 7, 8};

int size2 = sizeof(arr2) / sizeof(arr2[0]);

findIntersection(arr1, size1, arr2, size2);

return 0;

}

1. void findIntersection(int arr1[], int size1, int arr2[], int size2) {

   // TODO: Implement logic to find and print the intersection of two arrays

}

int main() {

   int arr1[] = {1, 2, 3, 4, 5};

   int size1 = sizeof(arr1) / sizeof(arr1[0]);

   int arr2[] = {3, 4, 5, 6, 7};

   int size2 = sizeof(arr2) / sizeof(arr2[0]);

  findIntersection(arr1, size1, arr2, size2);

   return 0;

}

1. #include <stdio.h>

 void square(int \*num) {

   // TODO: Modify the value pointed to by num to its square

}

#include<stdio.h>  
int main()  
{  
    char str[]="hello pointers";  
    char\* ptr=str;  
    for(int i=0;str[i]!='/0';i++)  
    printf("%c",\*(ptr+i));  
}

1. #include <stdio.h>

void swap(int \*a, int \*b) {

   // TODO: Swap the values pointed to by a and b

}

 #include<stdio.h>  
int swap (int \*a,int \*b){  
int \*ptr;  
ptr=\*a;  
\*a=\*b;  
\*b=ptr;}  
int main()  
{  
    int a=10,b=20;  
    printf("before swapping: %d,%d",a,b);  
    swap(&a,&b);  
    printf("after swapping :%d,%d",a,b);  
    return 0;  
}

1. The **calculateSumAndProduct** function takes two numbers (**num1** and **num2**) and two pointers (**sum** and **product**) as parameters.
2. Inside the function, it calculates the sum and product of the two numbers and stores the results at the memory locations pointed to by **sum** and **product**.
3. In the **main** function, we declare variables **sum** and **product**.
4. We then call the **calculateSumAndProduct** function, passing the values of **num1** and **num2**, as well as the addresses of **sum** and **product**.
5. After the function call, the values of **sum** and **product** are updated, and we can print the results

You are tasked with creating a simple program that sorts an array of integers using different sorting algorithms. However, instead of implementing each sorting algorithm separately, you are required to utilize function pointers to switch between them dynamically.

#include <stdio.h>

// Function pointer type for sorting algorithms

typedef void (\*SortFunc)(int[], int);

// Bubble sort algorithm

void bubbleSort(int arr[], int n) {

// Implementation of bubble sort

// ...

}

// Selection sort algorithm

void selectionSort(int arr[], int n) {

// Implementation of selection sort

// ...

}

// Insertion sort algorithm

void insertionSort(int arr[], int n) {

// Implementation of insertion sort

// ...

}

int main() {

int arr[] = {5, 2, 8, 1, 3};

int n = sizeof(arr) / sizeof(arr[0]);

// Function pointer to hold the selected sorting algorithm

SortFunc sortFunc;

// Prompt the user to select a sorting algorithm

int choice;

printf("Select a sorting algorithm:\n");

printf("1. Bubble Sort\n");

printf("2. Selection Sort\n");

printf("3. Insertion Sort\n");

printf("Enter your choice: ");

scanf("%d", &choice);

// Assign the selected sorting algorithm to the function pointer

switch (choice) {

case 1:

sortFunc = bubbleSort;

break;

case 2:

sortFunc = selectionSort;

break;

case 3:

sortFunc = insertionSort;

break;

default:

printf("Invalid choice!\n");

return 1;

}

// Call the selected sorting algorithm using the function pointer

sortFunc(arr, n);

// Print the sorted array

printf("Sorted array: ");

for (int i = 0; i < n; i++) {

printf("%d ", arr[i]);

}

printf("\n");

return 0;

}

Write a C program that implements three sorting algorithms: bubble sort, selection sort, and insertion sort. Define each sorting function to take an integer array and its size as arguments and sort the array in ascending order. Additionally, declare a function pointer type named SortFunction that points to functions with the same signature.

#include <stdio.h>

// Function pointer type for sorting functions

typedef void (\*SortFunction)(int arr[], int size);

// Swap two integers

void swap(int\* a, int\* b) {

int temp = \*a;

\*a = \*b;

\*b = temp;

}

// Bubble Sort

void bubbleSort(int arr[], int size) {

for (int i = 0; i < size - 1; ++i) {

for (int j = 0; j < size - i - 1; ++j) {

if (arr[j] > arr[j + 1]) {

swap(&arr[j], &arr[j + 1]);

}

}

}

}

// Selection Sort

void selectionSort(int arr[], int size) {

for (int i = 0; i < size - 1; ++i) {

int minIndex = i;

for (int j = i + 1; j < size; ++j) {

if (arr[j] < arr[minIndex]) {

minIndex = j;

}

}

swap(&arr[i], &arr[minIndex]);

}

}

// Insertion Sort

void insertionSort(int arr[], int size) {

for (int i = 1; i < size; ++i) {

int key = arr[i];

int j = i - 1;

while (j >= 0 && arr[j] > key) {

arr[j + 1] = arr[j];

--j;

}

arr[j + 1] = key;

}

}

// Print array

void printArray(int arr[], int size) {

for (int i = 0; i < size; ++i) {

printf("%d ", arr[i]);

}

printf("\n");

}

int main() {

int arr[] = {5, 1, 4, 2, 8};

int N = sizeof(arr) / sizeof(arr[0]);

// Declare function pointers

SortFunction bubbleSortPtr = bubbleSort;

SortFunction selectionSortPtr = selectionSort;

SortFunction insertionSortPtr = insertionSort;

// Sort using function pointers

bubbleSortPtr(arr, N);

printf("Bubble Sorted array: ");

printArray(arr, N);

selectionSortPtr(arr, N);

printf("Selection Sorted array: ");

printArray(arr, N);

insertionSortPtr(arr, N);

printf("Insertion Sorted array: ");

printArray(arr, N);

return 0;

}

Your program should prompt the user to enter the number of elements in the array and the elements themselves. Then, it should display a menu allowing the user to choose the sorting algorithm they want to use. Based on the user's choice, use the appropriate function pointer to sort the array and display the sorted result.

#include <stdio.h>

#include <stdlib.h> // For dynamic memory allocation

// Function pointer type for sorting functions

typedef void (\*SortFunction)(int arr[], int size);

// Swap two integers

void swap(int\* a, int\* b) {

int temp = \*a;

\*a = \*b;

\*b = temp;

}

// Bubble Sort

void bubbleSort(int arr[], int size) {

for (int i = 0; i < size - 1; ++i) {

for (int j = 0; j < size - i - 1; ++j) {

if (arr[j] > arr[j + 1]) {

swap(&arr[j], &arr[j + 1]);

}

}

}

}

// Selection Sort

void selectionSort(int arr[], int size) {

for (int i = 0; i < size - 1; ++i) {

int minIndex = i;

for (int j = i + 1; j < size; ++j) {

if (arr[j] < arr[minIndex]) {

minIndex = j;

}

}

swap(&arr[i], &arr[minIndex]);

}

}

// Insertion Sort

void insertionSort(int arr[], int size) {

for (int i = 1; i < size; ++i) {

int key = arr[i];

int j = i - 1;

while (j >= 0 && arr[j] > key) {

arr[j + 1] = arr[j];

--j;

}

arr[j + 1] = key;

}

}

// Print array

void printArray(int arr[], int size) {

for (int i = 0; i < size; ++i) {

printf("%d ", arr[i]);

}

printf("\n");

}

int main() {

int N;

printf("Enter the number of elements in the array: ");

scanf("%d", &N);

int\* arr = (int\*)malloc(N \* sizeof(int));

if (arr == NULL) {

printf("Memory allocation failed.\n");

return 1;

}

printf("Enter %d elements:\n", N);

for (int i = 0; i < N; ++i) {

scanf("%d", &arr[i]);

}

// Display menu

printf("\nSorting Algorithms Menu:\n");

printf("1. Bubble Sort\n");

printf("2. Selection Sort\n");

printf("3. Insertion Sort\n");

printf("Enter your choice (1/2/3): ");

int choice;

scanf("%d", &choice);

// Declare function pointers

SortFunction sortPtr;

switch (choice) {

case 1:

sortPtr = bubbleSort;

break;

case 2:

sortPtr = selectionSort;

break;

case 3:

sortPtr = insertionSort;

break;

default:

printf("Invalid choice. Exiting.\n");

free(arr);

return 1;

}

// Sort using the selected function pointer

sortPtr(arr, N);

printf("Sorted array: ");

printArray(arr, N);

// Clean up

free(arr);

return 0;

}

You are tasked with writing a C program that checks the strength of a password entered by a user. The program should analyze the password based on the following criteria:

1. The password must be at least 8 characters long.
2. The password must contain at least one uppercase letter, one lowercase letter, one digit, and one special character (such as !, @, #, $, %, etc.).
3. No spaces are allowed in the password.

Your program should prompt the user to enter a password and then evaluate its strength according to the criteria mentioned above. If the password meets all criteria, it should be deemed strong; otherwise, it should be considered weak.

Your program should provide informative messages to the user regarding the strength of their password and which criteria it fails to meet if it is weak.

Here are some guidelines to help you implement the program:

1. Use string functions like strlen(), isupper(), islower(), isdigit(), and strchr() for string manipulation and character checking.
2. You may use loops and conditional statements as necessary.
3. Ensure error handling for cases where the password does not meet the minimum length requirement or if spaces are found.
4. Utilize meaningful prompts and messages for user interaction.

#include<ctype.h>

int main(){  
 char password[15];  
 printf("Enter the password:\n");  
 scanf("%s",password);  
 int len = strlen(password);  
 if(len<8){  
 printf("Your password is less than 8 characters. So it is weak\n");  
 return 0;  
 }  
 int upper = 0, symbol = 0, lower = 0, number = 0;  
 for(int i=0; password[i]; i++){  
 if(upper == 0 && isupper(password[i])){  
 upper++;  
 }  
 else if(symbol == 0 && !isalnum(password[i])){  
 symbol++;  
 }  
 else if(lower == 0 && islower(password[i])){  
 lower++;  
 }  
 else if(number == 0 && isdigit(password[i])){  
 number++;  
 }  
 }  
 if((number && upper && lower && symbol)!=0){  
 printf("Your password is strong.\n");  
 }  
 else{  
 printf("Your password is weak.\n");  
 if(number == 0){  
 printf("There is no digit in your password.\n");  
 }  
 if(upper == 0){  
 printf("There is no upper case in your password.\n");  
 }  
 if(lower == 0){  
 printf("There is no lower case in youe password.\n");  
 }  
 if(symbol == 0){  
 printf("There is no special characters in your password.\n");  
 }  
 }  
 return 0;  
 }

You are tasked with writing a C program that analyzes a sentence entered by a user. The program should prompt the user to enter a sentence containing multiple words. After receiving the input, the program should tokenize the sentence into individual words using strtok and then perform the following tasks:

1. Count the total number of words in the sentence.
2. Determine the average length of the words in the sentence.
3. Identify and print the longest word in the sentence.

#include <stdio.h>

#include <string.h>

int main() {

char source[100];

printf("Enter your sentence: ");

scanf("%[^\n]", source);

char \*delimiters=" ";

char\* token = strtok(source, delimiters);

char\* longest=token;

int count=0,avg=0;

int ln=strlen(longest);

int currentLength;

while (token != NULL) {

if(token ==NULL)

break;

currentLength= strlen(token);

count=count+1;

avg = avg+currentLength;

if(currentLength>ln)

{

longest=token;

ln=currentLength;

}

token = strtok(NULL, delimiters);

}

avg=avg/count;

printf("the no of word is %d\n",count);

printf("the average length od words is %d\n",avg);

printf("the longest word is %s\n",longest);

return 0;

}