

ОТГОВОРИ, УПЪТВАНИЯ И ПРИМЕРНИ ИЗГЛЕДИ
НА РЕШЕНИЯТА НА ПРАКТИЧЕСКИТЕ
ЗАДАЧИ ОТ

ТЕМА 8

| | | | | | | | | | | | | | | | | |
|---------|---|---|---|---|---|---|---|---|---|----|----|----|----|----|----|----|
| Въпрос | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 |
| Отговор | Б | Б | В | Б | Г | Б | А | Г | В | Б | В | А | А | А | Г | Б |

17.

C#

```
using System;
class Program
{
    static void Main()
    {
        int i, number, fact;
        Console.WriteLine("Въведете число:");
        number = int.Parse(Console.ReadLine());
        fact = number;
        for (i = number - 1; i >= 1; i--)
        {
            fact = fact * i;
        }
        Console.WriteLine("Факториел = " + fact);
        Console.ReadLine();
    }
}
```

Java

```
import java.util.Scanner;
public class Main {
    public static void main(String[] args) {
        int i, number, fact;
        Scanner scanner = new Scanner(System.in);
        System.out.println("Въведете число:");
        number = scanner.nextInt();
        fact = number;
        for (i = number - 1; i >= 1; i--) {
            fact = fact * i;
        }
        System.out.println("Факториел = " + fact);
        scanner.close();
    }
}
```

18.

C#

```

using System;
class Program
{
    static void Main()
    {
        int[] intArray = new int[] { 89, 4, 23, 33, 3, 85, 40,
18, 23 };
        int temp = 0;

        for (int i = 0; i <= intArray.Length - 1; i++)
        {
            for (int j = i + 1; j < intArray.Length; j++)
            {
                if (intArray[i] < intArray[j])
                {
                    temp = intArray[i];
                    intArray[i] = intArray[j];
                    intArray[j] = temp;
                }
            }
        }

        Console.WriteLine("Массив в низходящ ред:");

        foreach (int element in intArray)
        {
            Console.Write(element + " ");
        }

        Console.ReadLine();
    }
}

```

Java

```

public class Main {
    public static void main(String[] args) {
        int[] intArray = { 89, 4, 23, 33, 3, 85, 40, 18, 23 };
        int temp = 0;
        for (int i = 0; i <= intArray.length - 1; i++) {
            for (int j = i + 1; j < intArray.length; j++) {
                if (intArray[i] < intArray[j]) {
                    temp = intArray[i];
                    intArray[i] = intArray[j];
                    intArray[j] = temp;
                }
            }
        }
        System.out.println("Массив в низходящ ред:");
        for (int element : intArray) {
            System.out.print(element + " ");
        }
    }
}

```

19.

| C# | Java |
|--|------|
| Програмата преброява думите в изречението, т.е. резултатът ще е 4. | |

20.

| teacher | course |
|---------|------------|
| Петрова | математика |

21.

| Exam_Title | Average_Result |
|---------------------|----------------|
| Програмиране в Java | 5.50 |

22. Г

23. А

24. (1) Брой на операциите на изваждане и добавяне: 3

(2) Елементите в стека след изпълнението на операциите: 0, 3, -2, 6

(3) Елементите на опашката след изпълнението на операциите: 10, -5, 8

25. Възможно решение

| C# |
|---|
| <pre>using System; class Program { static void Main() { try { Console.Write("Enter a positive integer N: "); int N = int.Parse(Console.ReadLine()); if (N <= 0) { Console.WriteLine("Please enter a positive integer."); return; } Console.Write(\$"Divisors of {N}: "); for (int i = 1; i <= N; i++) { if (N % i == 0) { Console.Write(i + " "); } } } catch (Exception) { Console.WriteLine("Something went wrong!"); } } }</pre> |

Java

```
import java.util.Scanner;
public class Main {
    public static void main(String[] args) {
        try {
            Scanner scanner = new Scanner(System.in);
            System.out.print("Enter a positive integer N: ");
            int N = scanner.nextInt();

            if (N <= 0) {
                System.out.println("Please enter a positive
integer.");
                return;
            }

            System.out.print("Divisors of " + N + ": ");
            for (int i = 1; i <= N; i++) {
                if (N % i == 0) {
                    System.out.print(i + " ");
                }
            }
        } catch (Exception e) {
            System.out.println("Something went wrong!");
        }
    }
}
```

26.

А) Създаване на таблица **Offices** и добавяне на записи:

```
CREATE TABLE Offices (
    ID INT PRIMARY KEY,
    City VARCHAR(100) NOT NULL,
    Country VARCHAR(100) NOT NULL
);
```

Добавяне на записи

```
INSERT INTO Offices (ID, City, Country) VALUES (1, 'София', 'България');
INSERT INTO Offices (ID, City, Country) VALUES (2, 'Берлин', 'Германия');
```

Б) Създаване на таблица **Employees** и добавяне на записи:

*Създаване на таблица **Employees***

```
CREATE TABLE Employees (
    Empl_ID INT PRIMARY KEY,
    name VARCHAR(100) NOT NULL,
    Job_title VARCHAR(100) NOT NULL,
    Office_code INT,
```

FOREIGN KEY (Office_code) REFERENCES Offices (ID)
);

В) За създадената таблица **Employees**, напишете SQL заявки, които да добавят следните кортежи/записи:

```
INSERT INTO Employees (Empl_ID, name, Job_title, Office_code) VALUES (1, 'Иван Петров', 'директор', 1);
```

```
INSERT INTO Employees (Empl_ID, name, Job_title, Office_code) VALUES (2, 'Анна Смит', 'директор', 2);
```

```
INSERT INTO Employees (Empl_ID, name, Job_title, Office_code) VALUES (3, 'Никола Николов', 'Продажби и маркетинг', 1);
```

```
INSERT INTO Employees (Empl_ID, name, Job_title, Office_code) VALUES (4, 'Елена Стоянова', 'администратор', 1);
```

Г) Напишете заявка, която да връща информация за кода на офиса, града и държавата, в които се намира, и имената и длъжностите на служителите във всеки от офисите, сортирани по азбучен ред на офиса.

```
SELECT o.ID AS Office_ID, o.City, o.Country, e.name AS Employee_Name,  
e.Job_title  
FROM Offices o  
JOIN Employees e ON o.ID = e.Office_code  
ORDER BY o.ID;
```

Д) Заявка за коригиране на длъжността на директора:

```
UPDATE Employees  
SET Job_title = 'координатор'  
WHERE Empl_ID = 1;
```

Е) Заявка за градовете на офисите с повече от 2-ма служители:

```
SELECT o.City  
FROM Offices o  
JOIN Employees e ON o.ID = e.Office_code  
GROUP BY o.City  
HAVING COUNT(e.Empl_ID) > 2;
```

27.

Създаване на базата данни

```
CREATE DATABASE IF NOT EXISTS diploma_management;
```

Използване на базата данни

```
USE diploma_management;
```

Създаване на таблица „**students**“

```
CREATE TABLE students (  
    id INT PRIMARY KEY,  
    name VARCHAR(255),  
    faculty VARCHAR(255),  
    enrollment_year INT,  
    average_grade DECIMAL(5, 2)  
);
```

Създаване на таблица „**diplomas**“

```
CREATE TABLE diplomas (  
    id INT PRIMARY KEY,  
    student_id INT,  
    graduation_year INT,  
    degree VARCHAR(255),  
    honors BOOLEAN,  
    FOREIGN KEY (student_id) REFERENCES students(id)  
);
```

А) Добавете в таблицата „**students**“ следните данни за студенти:

```
INSERT INTO students (id, name, faculty, enrollment_year, average_grade)  
VALUES
```

```
(1, 'Иван Иванов', 'Информатика', 2018, 5.55),  
(2, 'Мария Петрова', 'Медицина', 2017, 5.95),  
(3, 'Георги Георгиев', 'Икономика', 2019, 5.20),  
(4, 'Петър Петров', 'Електротехника', 2016, 5.80),  
(5, 'Анна Иванова', 'Право', 2020, 5.65);
```

Б) Добавете в таблицата „**diplomas**“ следните данни за дипломи:

```
INSERT INTO diplomas (id, student_id, graduation_year, degree, honors)  
VALUES
```

```
(1, 2, 2021, 'Доктор по Медицина', true),  
(2, 4, 2020, 'Инженер по Електротехника', false),  
(3, 1, 2019, 'Бакалавър по Информатика', true),  
(4, 3, 2021, 'Магистър по Икономика', false),  
(5, 5, 2022, 'Бакалавър по Право', true);
```

В) Напишете заявка, която извежда имената и факултетите на студентите, завършили с отличие.

```
SELECT s.name, s.faculty  
FROM students s  
JOIN diplomas d ON s.id = d.student_id  
WHERE d.honors = true;
```

Г) Напишете заявка, която извежда средния успех на студентите, които са завършили през последните три години.

```
SELECT AVG(s.average_grade) AS average_grade  
FROM students s  
JOIN diplomas d ON s.id = d.student_id  
WHERE d.graduation_year >= YEAR(CURRENT_DATE) - 3;
```

Д) Напишете заявка, която обновява средния успех на студента с име Мария Петрова на 5.75.

```
UPDATE students  
SET average_grade = 5.75  
WHERE name = 'Мария Петрова';
```

Е) Напишете заявка, която извежда имената на студентите и техните факултети, които са записани след 2018 година и имат среден успех над 5.50.

```
SELECT name, faculty  
FROM students  
WHERE enrollment_year > 2018 AND average_grade > 5.50;
```

Ж) Напишете заявка, която извежда общия брой студенти във всеки факултет.

```
SELECT faculty, COUNT(id) AS total_students  
FROM students  
GROUP BY faculty;
```

28. Примерно решение

C#

```
using System;
using System.Collections.Generic;
class Product
{
    private string name;
    private decimal price;
    private int quantity;
    public Product(string name, decimal price, int quantity)
    {
        this.name = name;
        this.price = price;
        this.quantity = quantity;
    }
    public override string ToString()
    {
        return $"Product: {name}, Price: {price} lv., Available
Quantity: {quantity} pcs.";
    }
}
class Customer
{
    private string name;
    private int age;
    public Customer(string name, int age)
    {
        this.name = name;
        this.age = age;
    }
    public override string ToString()
    {
        return $"Customer: {name}, Age: {age} years";
    }
}
class Store
{
    private string name;
    private List<Product> products;
    private List<Customer> customers;
    public Store(string name)
    {
        this.name = name;
        this.products = new List<Product>();
        this.customers = new List<Customer>();
    }
    public void AddProduct(Product product)
    {
        products.Add(product);
    }
}
```



```

public void AddCustomer(Customer customer)
{
    customers.Add(customer);
}
public void ShowInformation()
{
    Console.WriteLine($"Store: {name} Products: ");
    foreach (Product product in products)
    {
        Console.WriteLine(product);
    }
    Console.WriteLine("Customers: ");
    foreach (Customer customer in customers)
    {
        Console.WriteLine(customer);
    }
}
}
class Program
{
    static void Main()
    {
        Console.Write("Enter the store name: ");
        string storeName = Console.ReadLine();
        Store store = new Store(storeName);
        while (true)
        {
            Console.WriteLine("Choose an option:");
            Console.WriteLine("p - Add product");
            Console.WriteLine("c - Add customer");
            Console.WriteLine("i - Show information");
            Console.WriteLine("q - Quit");
            char choice = Console.ReadLine().ToLower()[0];
            switch (choice)
            {
                case 'p':
                    Console.Write("Enter product name: ");
                    string productName = Console.ReadLine();
                    Console.Write("Enter product price: ");
                    decimal productPrice;
                    while (!decimal.TryParse(Console.ReadLine(),
out productPrice))
                    {
                        Console.WriteLine("Invalid price format.
Please enter a valid number.");
                        Console.Write("Enter product price: ");
                    }
                    Console.Write("Enter product quantity: ");
                    int productQuantity;
                    while (!int.TryParse(Console.ReadLine(), out
productQuantity))
                    {
                        Console.WriteLine("Invalid quantity
format. Please enter a valid number.");
                        Console.Write("Enter product quantity: ");
                    }
                }
            }
        }
    }
}

```

```

        store.AddProduct(new Product(productName,
productPrice, productQuantity));
        break;
    case 'c':
        Console.Write("Enter customer name: ");
        string customerName = Console.ReadLine();
        Console.Write("Enter customer age: ");
        int customerAge;
        while (!int.TryParse(Console.ReadLine(), out
customerAge))
        {
            Console.WriteLine("Invalid age format.
Please enter a valid number.");
            Console.Write("Enter customer age: ");
        }
        store.AddCustomer(new Customer(customerName,
customerAge));
        break;
    case 'i':
        store.ShowInformation();
        break;
    case 'q':
        Environment.Exit(0);
        break;
    default:
        Console.WriteLine("Invalid choice. Try
again.");
        break;
    }
}
}
}

```

Java

```

import java.util.ArrayList;
import java.util.List;
import java.util.Scanner;
class Product {
    private String name;
    private double price;
    private int quantity;
    public Product(String name, double price, int quantity) {
        this.name = name;
        this.price = price;
        this.quantity = quantity;
    }
    @Override
    public String toString() {
        return "Product: " + name + ", Price: " + price + " lv.,
Available Quantity: " + quantity + " pcs.";
    }
}

```

```

class Customer {
    private String name;
    private int age;
    public Customer(String name, int age) {
        this.name = name;
        this.age = age;
    }
    @Override
    public String toString() {
        return "Customer: " + name + ", Age: " + age + "years";
    }
}

class Store {
    private String name;
    private List<Product> products;
    private List<Customer> customers;
    public Store(String name) {
        this.name = name;
        this.products = new ArrayList<>();
        this.customers = new ArrayList<>();
    }
    public void addProduct(Product product) {
        products.add(product);
    }
    public void addCustomer(Customer customer) {
        customers.add(customer);
    }
    public void showInformation() {
        System.out.println("Store: " + name + " Products: ");
        for (Product product : products) {
            System.out.println(product);
        }
        System.out.println("Customers: ");
        for (Customer customer : customers) {
            System.out.println(customer);
        }
    }
}

class Main {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter the store name: ");
        String storeName = scanner.nextLine();
        Store store = new Store(storeName);
        while (true) {
            System.out.println("Choose an option:");
            System.out.println("p - Add product");
            System.out.println("c - Add customer");
            System.out.println("i - Show information");
            System.out.println("q - Quit");
        }
    }
}

```

```

        char choice = scanner.next().charAt(0);
        switch (choice) {
            case 'p':
                System.out.print("Enter product name: ");
                String productName = scanner.next();
                System.out.print("Enter product price: ");
                double productPrice = scanner.nextDouble();
                System.out.print("Enter product quantity: ");
                int productQuantity = scanner.nextInt();
                store.addProduct(new Product(productName,
productPrice, productQuantity));
                break;
            case 'c':
                System.out.print("Enter customer name: ");
                String customerName = scanner.next();
                System.out.print("Enter customer age: ");
                int customerAge = scanner.nextInt();
                store.addCustomer(new Customer(customerName,
customerAge));
                break;
            case 'i':
                store.showInformation();
                break;
            case 'q':
                System.exit(0);
                break;
            default:
                System.out.println("Invalid choice. Try
again.");
                break;
        }
    }
}
}

```