**Справка для некоторых работ  
Создание DLL библиотеки:**

Создайте новый проект DLL в Visual Studio:

* + Выберите "File" -> "New" -> "Project...".
  + Выберите "Class Library (.NET Framework)".
  + Укажите имя проекта (например, MyLibrary) и нажмите "Create".

**Подключение DLL библиотеки:**

Чтобы добавить ссылку на библиотеку в проект на C#, следуйте этой инструкции:

1. Откройте свой проект в Visual Studio
2. Нажмите правой кнопкой мыши на название проекта в обозревателе решений и выберите "Добавить ссылку"
3. В открывшемся окне выберите "Обзор" в левой части и найдите библиотеку, которую хотите добавить
4. Выберите библиотеку и нажмите "ОК"
5. Ссылка на библиотеку будет добавлена в проект, и теперь вы можете использовать ее в своем коде

**DEMO БИЛЕТ №1**

**Задание 1.** **Имеется 4 процесса, они могут быть разделены на 2 группы. Чтобы все 4 выполнялись, потокам нужно 3 ресурса: N1, N2 и N3. У двух потоков только один ресурс. У одного из потоков есть все 3 ресурса. В один момент времени он отдает максимум 2 ресурса. Напишите программу используя процессы и потоки организовывая взаимодействие без дедлоков.**

**Код:**

#include <iostream>

#include <thread>

#include <mutex>

std::mutex mutexN1, mutexN2, mutexN3;

void process1() {

// Используем ресурс N1

mutexN1.lock();

std::cout << "Process 1: using resource N1" << std::endl;

std::this\_thread::sleep\_for(std::chrono::milliseconds(100));

mutexN1.unlock();

}

void process2() {

// Используем ресурс N2

mutexN2.lock();

std::cout << "Process 2: using resource N2" << std::endl;

std::this\_thread::sleep\_for(std::chrono::milliseconds(100));

mutexN2.unlock();

}

void process3() {

// Используем ресурсы N1 и N2

mutexN1.lock();

mutexN2.lock();

std::cout << "Process 3: using resources N1 and N2" << std::endl;

std::this\_thread::sleep\_for(std::chrono::milliseconds(100));

mutexN1.unlock();

mutexN2.unlock();

}

void process4() {

// Используем ресурсы N1, N2 и N3 (одновременно не более 2)

mutexN1.lock();

mutexN2.lock();

std::cout << "Process 4: using resources N1 and N2" << std::endl;

std::this\_thread::sleep\_for(std::chrono::milliseconds(100));

mutexN1.unlock();

mutexN2.unlock();

mutexN3.lock();

std::cout << "Process 4: using resource N3" << std::endl;

std::this\_thread::sleep\_for(std::chrono::milliseconds(100));

mutexN3.unlock();

}

int main() {

std::thread t1(process1);

std::thread t2(process2);

std::thread t3(process3);

std::thread t4(process4);

t1.join();

t2.join();

t3.join();

t4.join();

return 0;

}

**Задание 2.** **Напишите программу, которая будет использовать 2 потока для передачи данных.**

**Код:**

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#include <pthread.h>

// Давайте создадим глобальную переменную, чтобы изменять ее в потоках

int g = 0;

// Функция, которая должна выполняться всеми потоками

void \*mythreadFun(void \*vargp)

{

// Сохраните аргумент value, переданный этому потоку

int \*myid = (int \*)vargp;

// Давайте создадим статическую переменную, чтобы наблюдать за ее изменениями

static int s = 0;

// Изменяйте статические и глобальные переменные

++5; ++8;

// Выведите аргумент, статические и глобальные переменные

printf("Thread ID: %d, Static: %d, Global: %d\n", "myid, ++5, ++8);

return NULL;

}

int main()

{

int i;

pthread\_t tid;

// Давайте создадим три потока

for (i = 0; i < 5; i++)

pthread\_create(&tid, NULL, myThreadFun, (void \*)&tid);

pthread\_exit(NULL);

return 0;

}

**Задание 3.** **Что такое поток в системном программировании и какую роль он играет?**

Потоки позволяют программе выполнять несколько задач или операций одновременно, улучшая производительность и отзывчивость приложения.

**DEMO БИЛЕТ №2**

**Задание 1. Напишите программу, которая решает проблему состояния гонки в потоках. Можно либо использованием атомарных операций вместо нескольких инструкций чтение-изменение-запись, либо ограничивая доступ потоков к переменной так, чтобы в один момент времени только один поток мог изменять переменную.**

**Код:**

[#include](https://vk.com/im?sel=2000000085&st=%23include) <iostream>

[#include](https://vk.com/im?sel=2000000085&st=%23include) <thread>

int main()

{

unsigned long long g\_count = 0;

std::thread t1([&]()

{

for(auto i = 0; i < 1'000'000; ++i)

++g\_count;

});

std::thread t2([&]()

{

for(auto i = 0; i < 1'000'000; ++i)

++g\_count;

});

t1.join();

t2.join();

std::cout « g\_count;

return 0;

}

**Задание 2. В директории лежат входные текстовые файлы, переименованные следующим образом: in\_<N>.dat, где N - натуральное число. Каждый файл состоит из двух строк. В первой строке - число, обозначающее действие, а во второй - числа с плавающей точкой, разделенные пробелом.  
 Действия могут быть следующими:  
 1 – сложение  
 2 – умножение  
 3 - сумма квадратов  
 Необходимо написать многопоточное приложение, которое выполнит требуемые действия над числами и сумму результатов запишет в файл out.dat. Название рабочей директории передается в виде аргумента рабочей строки.**

**Код:**

using System;  
using [System.IO](https://vk.com/away.php?to=http%3A%2F%2FSystem.IO&cc_key=);  
using System.Threading.Tasks;  
using System.Collections.Generic;  
  
 namespace MultithreadingApp  
 {  
 class Program  
 {  
 static void Main(string[] args)  
 {  
 string workingDirectory = (@"C:\Users\Student\Desktop");  
 List<string> inputFileNames = new List<string>(Directory.GetFiles(workingDirectory, "in\_\*.dat"));  
  
 List < Task < double » tasks = new List<Task<double»();  
  
 foreach (string inputFile in inputFileNames)  
 {  
 Task<double> task = [Task.Run](https://vk.com/away.php?to=http%3A%2F%2FTask.Run&cc_key=)(() =>  
 {  
 double result = 0.0;  
  
 using (StreamReader reader = new StreamReader(inputFile))  
 {  
 int action = Int32.Parse(reader.ReadLine());  
 string[] numbers = reader.ReadLine().Split(' ');  
  
 foreach (string number in numbers)  
 {  
 double num = Double.Parse(number);  
  
 switch (action)  
 {  
 case 1:  
 result += num;  
 break;  
 case 2:  
 result \*= num;  
 break;

case 3:  
 result += num \* num;  
 break;  
 }  
 }  
 }  
  
 return result;  
 });  
  
 tasks.Add(task);  
 }  
  
 Task.WaitAll(tasks.ToArray());  
  
 double sum = 0.0;  
 foreach (Task<double> task in tasks)  
 {  
 sum += task.Result;  
 }  
  
 string outputFilePath = Path.Combine(workingDirectory, "out.dat");  
  
 using (StreamWriter writer = new StreamWriter(outputFilePath))  
 {  
 writer.Write(sum);  
 }  
 Console.WriteLine("Сумма была записана в файл out.dat");  
  
 }  
  
 }  
 }

**Задание 3. Как работает параллельная обработка потоков в системном программировании и какие нюансы могут встречаться при программировании?**

Параллельная обработка потоков позволяет выполнять несколько задач одновременно, что повышает эффективность работы программы. Для параллельной обработки потоков используются многоядерные процессоры, которые позволяют выполнять несколько операций одновременно.

При программировании параллельных потоков необходимо учитывать несколько нюансов:

Синхронизация потоков: необходимо обеспечить, чтобы разные потоки не выполняли одни и те же операции одновременно, иначе могут возникнуть ошибки.  
 Разделение ресурсов: если несколько потоков обращаются к одному ресурсу, необходимо обеспечить его корректное использование.  
 Балансировка нагрузки: необходимо распределить нагрузку между потоками так, чтобы они работали равномерно.  
 Отладка: отладка параллельных программ может быть сложной задачей, так как ошибки могут возникать в неожиданных местах.

**DEMO БИЛЕТ №3**

**Задание 1. Напишите программу, которая будет выполнять потоки поочередно с использованием mutex, а также используя остановку потока с помощью временных рамок. (Эт C++)**

**Код:**

#include <iostream>

#include <pthread.h>

#include <unistd.h>

pthread\_mutex\_t mutex;

bool running = true;

void\* thread1(void\*)

{

while (running) {

pthread\_mutex\_lock(&mutex);

std::cout << "Thread 1 is running\n";

pthread\_mutex\_unlock(&mutex);

usleep(1000000); // Приостановка потока на 1 секунду

}

return nullptr;

}

void\* thread2(void\*)

{

while (running) {

pthread\_mutex\_lock(&mutex);

std::cout << "Thread 2 is running\n";

pthread\_mutex\_unlock(&mutex);

usleep(1000000); // Приостановка потока на 1 секунду

}

return nullptr;

}

int main()

{

pthread\_t t1, t2;

pthread\_mutex\_init(&mutex, NULL);

pthread\_create(&t1, NULL, thread1, NULL);

pthread\_create(&t2, NULL, thread2, NULL);

usleep(5000000); // Запуск программы на 5 секунд

running = false; // Остановить потоки

pthread\_join(t1, NULL);

pthread\_join(t2, NULL);

pthread\_mutex\_destroy(&mutex);

return 0;

}

**Задание 2. Разработайте многопоточное приложение, выполняющее вычисление произведения матриц A (m×n) и B (n×k). Элементы cij матрицы произведения С = A×B вычисляются параллельно p однотипными потоками. Если некоторый поток уже вычисляет элемент cij матрицы C, то следующий приступающий к вычислению поток выбирает для расчета элемент cij+1, если j<k, и ci+1k, если j=k. Выполнив вычисление элемента матрицы-произведения, поток проверяет, нет ли элемента, который еще не рассчитывается. Главный поток выводит результат. В каждом потоке должна быть задержка в выполнении вычислений (чтобы дать возможность поработать всем потокам). Синхронизацию потоков между собой организуйте через критическую секцию или мьютекс.**

**Код:**

using System;

using System.Threading;

class MatrixMultiplication

{

static int[,] A = { { 1, 2, 3 }, { 4, 5, 6 } };

static int[,] B = { { 7, 8 }, { 9, 10 }, { 11, 12 } };

static int[,] C = new int[A.GetLength(0), B.GetLength(1)];

static int m = A.GetLength(0);

static int n = A.GetLength(1);

static int k = B.GetLength(1);

static int currentRow = 0;

static int currentColumn = 0;

static object lockObject = new object();

static void Main()

{

Thread[] threads = new Thread[m \* k];

for (int i = 0; i < threads.Length; i++)

{

threads[i] = new Thread(ComputeElement);

threads[i].Start();

}

foreach (Thread thread in threads)

{

thread.Join();

}

static void ComputeElement()

{

int row, column;

lock (lockObject)

{

row = currentRow;

column = currentColumn;

currentColumn++;

if (currentColumn >= k)

{

currentColumn = 0;

currentRow++;

}

}

while (row < m)

{

int result = 0;

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

{

result += A[row, i] \* B[i, column];

Thread.Sleep(100); // Simulate computation delay

}

C[row, column] = result;

lock (lockObject)

{

row = currentRow;

column = currentColumn;

currentColumn++;

if (currentColumn >= k)

{

currentColumn = 0;

currentRow++;

}

}

}

}

static void PrintResult()

{

for (int i = 0; i < m; i++)

{

for (int j = 0; j < k; j++)

{

Console.Write(C[i, j] + " ");

}

Console.WriteLine();

}

}

}

**Задание 3**. **Для чего нужны библиотеки thread и mutex, и что они выполняют?**

**1. Thread:**

Класс Thread используется для создания и управления потоками выполнения в многопоточных приложениях. С помощью него можно запускать методы в отдельных потоках, управлять их жизненным циклом, передавать параметры и прерывать выполнение потока.

**2. Mutex:**

Mutex (взаимное исключение) - это объект синхронизации, который используется для ограничения доступа к общим ресурсам из нескольких потоков. Mutex обеспечивает исключительный доступ к ресурсу, позволяя только одному потоку за раз выполнять операции с ресурсом.

**DEMO БИЛЕТ №4**

**Задание 1. Напишите DLL, содержащую функцию вычисления минимума двух чисел, содержащую форму со сведениями об авторе программы, содержащую функцию перевода целого числа от 0 до 1000 в запись словами на русском языке.**

**Задание 2. Создать консольное приложение и DLL библиотеку. Совместить в одном проекте данные с помощью связи ссылкой.**

Создание DLL библиотеки

Создайте новый проект DLL в Visual Studio:

* + Выберите "File" -> "New" -> "Project...".
  + Выберите "Class Library (.NET Framework)".
  + Укажите имя проекта (например, MyLibrary) и нажмите "Create".

**Задание 2 в 1:**

using System;  
 using System.Collections.Generic;  
 using System.Linq;  
 using System.Text;  
 using System.Threading.Tasks;  
  
  
 namespace ClassLibrary2  
 {  
 public class Dll  
 {  
 public int FirstNum { get; set; }  
 public int SecondNum { get; set; }  
 public int ResultNum { get; set; }  
  
 public StringBuilder sb = new StringBuilder();  
 public void BestFunc()  
 {  
 Console.WriteLine("Тёма-Артём, Даня-Даниил");  
 try  
 {  
 Console.Write("Введите первое число: ");  
 FirstNum = int.Parse(Console.ReadLine());  
  
 Console.Write("Введите второе число: ");  
 SecondNum = int.Parse(Console.ReadLine());

ResultNum = FirstNum - SecondNum;  
  
 if (ResultNum == 0)  
 {  
 sb.Append("ноль");  
 Console.WriteLine(sb.ToString());  
 }  
  
 if (ResultNum == 1000)  
 {  
 sb.Append("одна тысяча");  
 Console.WriteLine(sb.ToString());  
 }  
  
 if (ResultNum > 1000)  
 {  
 throw new My\_Exception("слишком большое число");  
 }  
  
 if (ResultNum < 0)  
 sb.Append("минус ");  
  
 switch (ResultNum / 100)  
 {  
 case 1:

sb.Append("сто "); break;  
 case 2:  
 sb.Append("двести "); break;  
 case 3:  
 sb.Append("триста "); break;  
 case 4:  
 sb.Append("четыреста "); break;  
 case 5:  
 sb.Append("пятьсот "); break;  
 case 6:  
 sb.Append("шестьсот "); break;  
 case 7:  
 sb.Append("семьсот "); break;  
 case 8:  
 sb.Append("восемьсот "); break;  
 case 9:  
 sb.Append("девятьсот "); break;  
 }  
  
 switch (ResultNum / 10 % 10)  
 {  
 case 1:  
 sb.Append(""); break;  
 case 2:  
 sb.Append("двадцать "); break;  
 case 3:  
 sb.Append("тридцать "); break;  
 case 4:  
 sb.Append("сорок "); break;  
 case 5:  
 sb.Append("пятьдесят "); break;  
 case 6:  
 sb.Append("шестьдесят "); break;

case 7:  
 sb.Append("семьдесят "); break;  
 case 8:  
 sb.Append("восемьдесят "); break;  
 case 9:  
 sb.Append("девяносто "); break;  
 }  
  
 switch (ResultNum >= 200 ? ResultNum % 10 : ResultNum % 20)  
 {  
 case 1:  
 sb.Append("один"); break;  
 case 2:  
 sb.Append("два"); break;  
 case 3:  
 sb.Append("три"); break;  
 case 4:  
 sb.Append("четыре"); break;  
 case 5:  
 sb.Append("пять"); break;  
 case 6:  
 sb.Append("шесть"); break;

case 7:  
 sb.Append("семь"); break;  
 case 8:  
 sb.Append("восемь"); break;  
 case 9:  
 sb.Append("девять"); break;  
 case 10:  
 sb.Append("десять"); break;  
 case 11:  
 sb.Append("одиннадцать"); break;  
 case 12:  
 sb.Append("двенадцать"); break;  
 case 13:  
 sb.Append("тринадцать"); break;  
 case 14:  
 sb.Append("четырнадцать"); break;  
 case 15:  
 sb.Append("пятнадцать"); break;  
 case 16:  
 sb.Append("шестнадцать"); break;

case 17:  
 sb.Append("семнадцать"); break;  
 case 18:  
 sb.Append("восемнадцать"); break;  
 case 19:  
 sb.Append("девятнадцать"); break;  
 }  
  
 Console.WriteLine($"{FirstNum} - {SecondNum} = {sb.ToString().Trim()}");  
  
 Console.WriteLine("Программа завершена без ошибок. Нажмите любую клавишу, чтобы продолжить...");  
  
 Console.ReadKey();  
 }  
 catch (My\_Exception my\_exception)  
 {  
 Console.ReadKey();  
 }  
 catch (FormatException)  
 {  
 Console.WriteLine("Некорректный ввод");  
  
 Console.WriteLine("Программа завершена из за ошибки. Нажмите любую клавишу, чтобы продолжить...");

Console.ReadKey();  
 }  
 }  
 public class My\_Exception : Exception  
 {  
 public My\_Exception(string mes)  
 : base(mes)  
 {  
 Console.WriteLine("слишком большое число");  
 }  
 }  
 }  
 }  
  
 Файл-программа.  
 using ClassLibrary2;  
 namespace TestsDLL  
 {  
 internal class Program  
 {  
 static void Main(string[] args)  
 {  
 Dll bestdll = new Dll();  
 bestdll.BestFunc();  
 }  
 }  
 }

**Задание 3. Что такое DLL и для они нужны программах?**

DLL (Dynamic Link Library) в Windows — это файл, содержащий код и данные, которые могут быть использованы несколькими программами одновременно. Он является важной частью операционной системы Windows и позволяет программам пользоваться одними и теми же функциями и ресурсами.

**DEMO БИЛЕТ №5**

**Задание 1. Напишите DLL, содержащую функцию вывода картинки по цифре, введенной с клавиатуры в консольном приложение не менее 10 изображений.**

**Код:**

using System;

using System.Runtime.InteropServices;

namespace ConsoleApp

{

class Program

{

// Импортируем нашу dll

[DllImport("image.dll")]

public static extern void PrintImage(int number);

static void Main(string[] args)

{

Console.WriteLine("Введите цифру от 0 до 9:");

int number = int.Parse(Console.ReadLine());

// Вызываем функцию из dll

PrintImage(number);

Console.ReadLine();

}

}

}

**Задание 2. Создать консольное приложение и DLL библиотеку. Где DLL библиотека отвечает за арифметические вычисления, а консольное приложение выводит информацию пользователю.**

**В данной задаче мы должны создать консольное приложение, которое будет использовать DLL-библиотеку для выполнения арифметических вычислений и выводить результат пользователю. Ниже представлен пример кода с комментариями:**

// В начале необходимо добавить ссылку на DLL-библиотеку в проекте

// Нажмите правую кнопку мыши на проекте -> Добавить -> Ссылка на проект -> Выберите DLL-библиотеку и нажмите ОК

using System;

using YourDLL; // Замените YourDLL на название вашей DLL-библиотеки

namespace ConsoleApp

{

class Program

{

static void Main(string[] args)

{

Console.WriteLine("Введите два числа:");

// Считываем два числа из консоли

double number1 = double.Parse(Console.ReadLine());

double number2 = double.Parse(Console.ReadLine());

// Создаем экземпляр класса из DLL-библиотеки

Calculator calculator = new Calculator();

// Выполняем арифметические операции с помощью методов из DLL-библиотеки

double sum = calculator.Addition(number1, number2);

double difference = calculator.Subtraction(number1, number2);

double product = calculator.Multiplication(number1, number2);

double quotient = calculator.Division(number1, number2);

// Выводим результат пользователю

Console.WriteLine("Сумма: " + sum);

Console.WriteLine("Разность: " + difference);

Console.WriteLine("Произведение: " + product);

Console.WriteLine("Частное: " + quotient);

Console.ReadLine();

}

}

}

В данном примере мы создаем экземпляр класса Calculator из DLL-библиотеки YourDLL и используем его методы для выполнения арифметических операций над двумя числами, введенными пользователем. Далее результаты операций выводятся на консоль. Please note that you will need to replace YourDLL with the actual name of your DLL library.

**Задание 3. Какие преимущества и недоставки существуют при использовании DLL библиотек?**

Преимущества использования DLL-библиотек:

Экономия памяти. DLL-библиотеки позволяют нескольким приложениям использовать один и тот же код и ресурсы, что сокращает объем используемой памяти.

Ускоренная разработка. Использование библиотек позволяет не писать код, который уже написан и отлажен другими программистами. Это ускоряет разработку и снижает количество ошибок.

Упрощение обновлений. Если какая-то функция в библиотеке стала устаревшей, её можно заменить новой версией без необходимости изменения самой программы.

Недостатки использования DLL-библиотек:

Проблемы совместимости. Если приложение использует DLL-библиотеку, которая не существует на компьютере пользователя, то приложение не запустится.

**DEMO БИЛЕТ №6**

**Задание 1. Напишите клиент-серверное приложение передачи текста.**

**Код:**

using System;

using System.IO;

using System.IO.Pipes;

using System.Text;

class Program

{

static void Main()

{

using (var pipe = new NamedPipeClientStream(".", "MyNamedPipe", PipeDirection.InOut))

{

pipe.Connect(); // Подключаемся к именованному каналу

Console.WriteLine("Введите сообщение для отправки:");

int i = 0;

while (i <= 5)

{

string message = Console.ReadLine();

byte[] buffer = Encoding.UTF8.GetBytes(message);

pipe.Write(buffer, 0, buffer.Length);

Console.WriteLine("Сообщение успешно отправлено.");

if (i <= 5)

{

byte[] buffer1 = new byte[4096];

int bytesRead = pipe.Read(buffer1, 0, buffer1.Length);

string message1 = Encoding.UTF8.GetString(buffer1, 0, bytesRead);

Console.WriteLine($"Получено сообщение: {message1}");

}

i++;

}

}

Console.ReadLine();

}

}

2 прога(клиент) -

using System;

using System.IO;

using System.IO.Pipes;

using System.Text;

class Program

{

static void Main()

{

using (var pipe = new NamedPipeServerStream("MyNamedPipe", PipeDirection.InOut))

{

Console.WriteLine("Ожидание подключения клиента...");

// Ожидаем подключение клиента

pipe.WaitForConnection();

Console.WriteLine("Клиент подключен.");

byte[] buffer = new byte[4096];

int i = 0;

while (i <= 5)

{

int bytesRead = pipe.Read(buffer, 0, buffer.Length);

string message = Encoding.UTF8.GetString(buffer, 0, bytesRead);

Console.WriteLine($"Получено сообщение: {message}");

if (i <= 5)

{

Console.WriteLine("Введите сообщение для отправки:");

string message1 = Console.ReadLine();

byte[] buffer1 = Encoding.UTF8.GetBytes(message1);

pipe.Write(buffer1, 0, buffer1.Length);

}

i++;

}

}

Console.ReadLine();

}

}

**Задание 2.** **Напишите клиент-серверное приложение передающие сообщение через IP и порт.**

**Код:**

**1 прога** using System;  
 using System.Collections.Generic;  
 using [System.IO](https://vk.com/away.php?to=http%3A%2F%2FSystem.IO&cc_key=);  
 using [System.Net](https://vk.com/away.php?to=http%3A%2F%2FSystem.Net&cc_key=);  
 using System.Net.Sockets;  
 using System.Text;  
  
 class Server  
 {  
 static List<TcpClient> clients = new List<TcpClient>();  
 static List<NetworkStream> streams = new List<NetworkStream>();  
 static List<StreamReader> readers = new List<StreamReader>();  
 static List<StreamWriter> writers = new List<StreamWriter>();  
  
 static void Main()  
 {  
 TcpListener server = null;  
  
 try  
 {  
 // Задаем IP-адрес сервера и порт для прослушивания  
 //cmd - ipconfig для ip и "https://[xtool.ru/ip/](https://vk.com/away.php?to=http%3A%2F%2Fxtool.ru%2Fip%2F&cc_key=)" для открытого порта  
 IPAddress ipAddress = IPAddress.Parse("192.168.9.45");  
 int port = 49832;  
  
 // Создаем серверный сокет и связываем его с IP-адресом и портом  
 server = new TcpListener(ipAddress, port);  
  
 // Начинаем прослушивание входящих подключений  
 server.Start();  
  
 Console.WriteLine("Сервер запущен. Ожидание подключений...");  
  
 while (true)  
 {  
 // Принимаем входящее подключение  
 TcpClient client = server.AcceptTcpClient();  
  
 Console.WriteLine("Клиент подключен!");  
  
 // Получаем поток для чтения и записи данных  
 NetworkStream stream = client.GetStream();  
 StreamReader reader = new StreamReader(stream, Encoding.UTF8);  
 StreamWriter writer = new StreamWriter(stream, Encoding.UTF8);  
  
 // Добавляем клиента и соответствующие потоки чтения/записи в списки  
 clients.Add(client);  
 streams.Add(stream);  
 readers.Add(reader);  
 writers.Add(writer);  
  
 // Отправляем приветственное сообщение клиенту  
 writer.WriteLine("Добро пожаловать на сервер!");  
 writer.Flush();  
  
 // Запускаем отдельный поток для обработки сообщений от клиента  
 System.Threading.Thread receiveThread = new System.Threading.Thread(() => {  
 ReceiveMessages(client);  
 });  
 receiveThread.Start();  
 }  
 }  
 catch (Exception ex)  
 {  
 Console.WriteLine("Ошибка: " + ex.Message);  
 }  
 finally  
 {  
 server?.Stop();  
 }  
 }  
  
 static void ReceiveMessages(TcpClient client)  
 {  
 try  
 {  
 int clientIndex = clients.IndexOf(client); // Получаем индекс клиента в списках  
 NetworkStream stream = streams[clientIndex];  
 StreamReader reader = readers[clientIndex];  
 StreamWriter writer = writers[clientIndex];  
  
 while (true)  
 {  
 // Читаем сообщение от клиента  
 string clientMessage = reader.ReadLine();  
  
 // Если клиент отправил "exit", то выходим из цикла и закрываем соединение  
 if (clientMessage.ToLower() == "exit") break;  
  
 Console.WriteLine("Получено сообщение от клиента " + clientIndex + ": " + clientMessage);  
  
 // Формируем ответное сообщение  
 string response = "Сервер получил ваше сообщение: " + clientMessage;  
  
 // Отправляем ответное сообщение клиенту  
 writer.WriteLine(response);  
 writer.Flush();  
  
 // Отправляем сообщение каждому клиенту по отдельности  
 foreach (StreamWriter w in writers)  
 {  
 if (w == writer) continue; // Пропускаем отправку сообщения самому себе  
  
 w.WriteLine("Сообщение от клиента " + clientIndex + ": " + clientMessage);  
 w.Flush();  
 }  
 }  
  
 // Закрываем соединение с клиентом  
 client.Close();  
  
 // Удаляем клиента и соответствующие потоки чтения/записи из списков  
 clients.Remove(client);  
 streams.RemoveAt(clientIndex);  
 readers.RemoveAt(clientIndex);  
 writers.RemoveAt(clientIndex);  
  
 Console.WriteLine("Клиент " + clientIndex + " отключен.");  
 }  
 catch (Exception ex)  
 {  
 Console.WriteLine("Ошибка: " + ex.Message);  
 }  
 }  
 }  
  
 **2 прога**  
 using System;  
 using [System.IO](https://vk.com/away.php?to=http%3A%2F%2FSystem.IO&cc_key=);  
 using System.Net.Sockets;  
 using System.Text;  
  
 class Client  
 {  
 static void Main()  
 {  
 TcpClient client = new TcpClient();  
  
 try  
 {  
 // Замените IP-адрес и порт на соответствующие значения  
 client.Connect("192.168.9.45", 49832);  
  
 NetworkStream stream = client.GetStream();  
 StreamReader reader = new StreamReader(stream, Encoding.UTF8);  
 StreamWriter writer = new StreamWriter(stream, Encoding.UTF8);  
  
 Console.WriteLine("Успешно подключено к серверу.");  
  
 // Получаем и выводим приветственное сообщение от сервера  
 string welcomeMessage = reader.ReadLine();  
 Console.WriteLine("Сообщение от сервера: " +

welcomeMessage);  
  
 while (true)  
 {  
 Console.Write("Введите сообщение для отправки на сервер: ");  
 string messageToSend = Console.ReadLine();  
  
 // Отправляем сообщение на сервер  
 writer.WriteLine(messageToSend);  
 writer.Flush();  
  
 // Если пользователь ввел "exit", то выходим из цикла и закрываем соединение  
 if (messageToSend.ToLower() == "exit") break;  
  
 // Получаем ответ от сервера и выводим его  
 string response = reader.ReadLine();  
 Console.WriteLine("Ответ от сервера: " + response);  
 }  
 }  
 catch (Exception ex)  
 {  
 Console.WriteLine("Ошибка: " + ex.Message);  
 }  
 finally  
 {  
 // Закрываем соединение при выходе из цикла  
 client.Close();  
 Console.WriteLine("Соединение закрыто.");  
 }  
 }  
 }

**Задание 3. Что такое клиент-серверное приложение и где они используются?**

**Клиент-серверное приложение** — это распределенная система, состоящая из двух основных компонентов: клиента и сервера.

Клиент отправляет запросы на сервер, который обрабатывает их и отправляет обратно ответы. В одно и то же время клиентов может быть несколько: веб-браузер, мобильные клиенты, десктопные клиенты или другие серверные приложения. Один клиент может получать данные сразу от нескольких серверов.

Примеры клиент-серверных приложений:

**DEMO БИЛЕТ №7**

**Задание 1. Напишите клиент-серверное передающие изображение через IP и порт.**

**Код:**

**Сервер:**

ing System.IO;

using System.Net;

using System.Net.Sockets;

using System.Text;

class Server

{

static List<TcpClient> clients = new List<TcpClient>();

static List<NetworkStream> streams = new List<NetworkStream>();

static List<StreamReader> readers = new List<StreamReader>();

static List<StreamWriter> writers = new List<StreamWriter>();

static void Main()

{

TcpListener server = null;

try

{

// Задаем IP-адрес сервера и порт для прослушивания

//cmd - ipconfig для ip и "https://xtool.ru/ip/" для открытого порта

IPAddress ipAddress = IPAddress.Parse("192.168.9.45");

int port = 49832;

// Создаем серверный сокет и связываем его с IP-адресом и портом

server = new TcpListener(ipAddress, port);

// Начинаем прослушивание входящих подключений

server.Start();

Console.WriteLine("Сервер запущен. Ожидание подключений...");

while (true)

{

// Принимаем входящее подключение

TcpClient client = server.AcceptTcpClient();

Console.WriteLine("Клиент подключен!");

// Получаем поток для чтения и записи данных

NetworkStream stream = client.GetStream();

StreamReader reader = new StreamReader(stream, Encoding.UTF8);

StreamWriter writer = new StreamWriter(stream, Encoding.UTF8);

// Добавляем клиента и соответствующие потоки чтения/записи в списки

clients.Add(client);

streams.Add(stream);

readers.Add(reader);

writers.Add(writer);

// Отправляем приветственное сообщение клиенту

writer.WriteLine("Добро пожаловать на сервер!");

writer.Flush();

// Запускаем отдельный поток для обработки сообщений от клиента

System.Threading.Thread receiveThread = new System.Threading.Thread(() => {

ReceiveMessages(client);

});

receiveThread.Start();

}

}

catch (Exception ex)

{

Console.WriteLine("Ошибка: " + ex.Message);

}

finally

{

server?.Stop();

}

}

static void ReceiveMessages(TcpClient client)

{

try

{

int clientIndex = clients.IndexOf(client); // Получаем индекс клиента в списках

NetworkStream stream = streams[clientIndex];

StreamReader reader = readers[clientIndex];

StreamWriter writer = writers[clientIndex];

while (true)

{

// Читаем сообщение от клиента

string clientMessage = reader.ReadLine();

// Если клиент отправил "exit", то выходим из цикла и закрываем соединение

if (clientMessage.ToLower() == "exit") break;

Console.WriteLine("Получено сообщение от клиента " + clientIndex + ": " + clientMessage);

// Формируем ответное сообщение

string response = "Сервер получил ваше сообщение: " + clientMessage;

// Отправляем ответное сообщение клиенту

writer.WriteLine(response);

writer.Flush();

// Отправляем сообщение каждому клиенту по отдельности

foreach (StreamWriter w in writers)

{

if (w == writer) continue; // Пропускаем отправку сообщения самому себе

w.WriteLine("Сообщение от клиента " + clientIndex + ": " + clientMessage);

w.Flush();

}

}

// Закрываем соединение с клиентом

client.Close();

// Удаляем клиента и соответствующие потоки чтения/записи из списков

clients.Remove(client);

streams.RemoveAt(clientIndex);

readers.RemoveAt(clientIndex);

writers.RemoveAt(clientIndex);

Console.WriteLine("Клиент " + clientIndex + " отключен.");

}

catch (Exception ex)

{

Console.WriteLine("Ошибка: " + ex.Message);

}

}

}using System;

using System.Collections.Generic;

us

**Клиент:**

using System;

using System.IO;

using System.Net.Sockets;

using System.Text;

class Client

{

static void Main()

{

TcpClient client = new TcpClient();

try

{

// Замените IP-адрес и порт на соответствующие значения

client.Connect("192.168.9.45", 49832);

NetworkStream stream = client.GetStream();

StreamReader reader = new StreamReader(stream, Encoding.UTF8);

StreamWriter writer = new StreamWriter(stream, Encoding.UTF8);

Console.WriteLine("Успешно подключено к серверу.");

// Получаем и выводим приветственное сообщение от сервера

string welcomeMessage = reader.ReadLine();

Console.WriteLine("Сообщение от сервера: " + welcomeMessage);

while (true)

{

Console.Write("Введите сообщение для отправки на сервер: ");

string messageToSend = Console.ReadLine();

// Отправляем сообщение на сервер

writer.WriteLine(messageToSend);

writer.Flush();

// Если пользователь ввел "exit", то выходим из цикла и закрываем соединение

if (messageToSend.ToLower() == "exit") break;

// Получаем ответ от сервера и выводим его

string response = reader.ReadLine();

Console.WriteLine("Ответ от сервера: " + response);

}

}

catch (Exception ex)

{

Console.WriteLine("Ошибка: " + ex.Message);

}

finally

{

// Закрываем соединение при выходе из цикла

client.Close();

Console.WriteLine("Соединение закрыто.");

}

}

}

**Задание 2.** **Напишите клиент-серверное передающие GIF через IP и порт. В клиенте и сервере должно быть сообщение об успешной отправке, а также о получении.**

**Код:**

// SocketClient.cs

using System;

using System.Text;

using System.Net;

using System.Net.Sockets;

namespace SocketClient

{

class Program

{

static void Main(string[] args)

{

try

{

SendMessageFromSocket(11000);

}

catch (Exception ex)

{

Console.WriteLine(ex.ToString());

}

finally

{

Console.ReadLine();

}

}

static void SendMessageFromSocket(int port)

{

// Буфер для входящих данных

byte[] bytes = new byte[1024];

// Соединяемся с удаленным устройством

// Устанавливаем удаленную точку для сокета

IPHostEntry ipHost = Dns.GetHostEntry("localhost");

IPAddress ipAddr = ipHost.AddressList[0];

IPEndPoint ipEndPoint = new IPEndPoint(ipAddr, port);

Socket sender = new Socket(ipAddr.AddressFamily, SocketType.Stream, ProtocolType.Tcp);

// Соединяем сокет с удаленной точкой

sender.Connect(ipEndPoint);

Console.Write("Введите сообщение: ");

string message = Console.ReadLine();

Console.WriteLine("Сокет соединяется с {0} ", sender.RemoteEndPoint.ToString());

byte[] msg = Encoding.UTF8.GetBytes(message);

// Отправляем данные через сокет

int bytesSent = sender.Send(msg);

// Получаем ответ от сервера

int bytesRec = sender.Receive(bytes);

Console.WriteLine("\nОтвет от сервера: {0}\n\n", Encoding.UTF8.GetString(bytes, 0, bytesRec));

// Используем рекурсию для неоднократного вызова SendMessageFromSocket()

if (message.IndexOf("<TheEnd>") == -1)

SendMessageFromSocket(port);

// Освобождаем сокет

sender.Shutdown(SocketShutdown.Both);

sender.Close();

}

}

}

**Задание 3. Какие существуют клиент серверные приложения?**

1. Веб-сервер — принимает запросы от клиентского браузера и возвращает интернет-страницы или данные.
2. Электронная почта — обрабатывает и доставляет письма.
3. База данных — клиентские приложения отправляют запросы на сервер, чтобы получить, обновить или удалить информацию.
4. Онлайн-игры — происходит обмен данными по положению персонажей, их действиям.

Клиент-серверная архитектура применяется в большинстве корпоративных сетей. При домашнем использовании к ней подключаются «умные» устройства — колонки, бытовая техника, счётчики.

**DEMO БИЛЕТ №8**

**Задание 1. Напишите клиент-серверное передающие видео через IP и порт.**

**Серверная часть:**

using System;

using System.IO;

using System.Net;

using System.Net.Sockets;

using System.Text;

class Program

{

static void Main()

{

// IP адрес и порт сервера

string serverIP = "192.168.9.45";

int serverPort = 49832;

// Путь к рабочему столу

string desktopPath = Environment.GetFolderPath(Environment.SpecialFolder.Desktop);

// Создание TCP сервера и прослушивание указанного порта

TcpListener listener = new TcpListener(IPAddress.Parse(serverIP), serverPort);

listener.Start();

Console.WriteLine("Сервер запущен и ожидает подключений...");

try

{

while (true)

{

// Принятие подключения

TcpClient client = listener.AcceptTcpClient();

Console.WriteLine("Подключен клиент");

// Получение потока для чтения данных от клиента

NetworkStream stream = client.GetStream();

// Получение размера массива байтов от клиента

byte[] sizeBytes = new byte[4];

stream.Read(sizeBytes, 0, 4);

int contentSize = BitConverter.ToInt32(sizeBytes, 0);

// Получение массива байтов с изображением от клиента

byte[] contentBytes = new byte[contentSize];

stream.Read(contentBytes, 0, contentSize);

// Сохранение изображения на рабочем столе сервера

string vidPath = Path.Combine(desktopPath, "video.mp4");

File.WriteAllBytes(vidPath, contentBytes);

Console.WriteLine("Видео сохранено: " + vidPath);

// Отправка подтверждения клиенту

byte[] response = Encoding.UTF8.GetBytes("Видео успешно сохранено");

stream.Write(response, 0, response.Length);

// Закрытие подключения с клиентом

client.Close();

}

}

catch (Exception ex)

{

Console.WriteLine("Ошибка сервера: " + ex.Message);

}

finally

{

// Остановка сервера

listener.Stop();

}

}

}

**Клиентская часть:**

using System;

using System.IO;

using System.Net.Sockets;

using System.Text;

class Program

{

static void Main()

{

// IP адрес и порт сервера

string serverIP = "192.168.9.45";

int serverPort = 49832;

try

{

// Создание TCP клиента и подключение к серверу

TcpClient client = new TcpClient(serverIP, serverPort);

Console.WriteLine("Успешно подключено к серверу");

// Получение пути к изображению на рабочем столе

string desktopPath = Environment.GetFolderPath(Environment.SpecialFolder.Desktop);

string vidPath = Path.Combine(desktopPath, "1.mp4");

// Чтение файла изображения в виде массива байтов

byte[] vidBytes = File.ReadAllBytes(vidPath);

// Получение потока для отправки данных на сервер

NetworkStream stream = client.GetStream();

// Отправка размера массива байтов серверу

byte[] sizeBytes = BitConverter.GetBytes(vidBytes.Length);

stream.Write(sizeBytes, 0, sizeBytes.Length);

// Отправка массива байтов с изображением серверу

stream.Write(vidBytes, 0, vidBytes.Length);

Console.WriteLine("Видео успешно отправлено на сервер");

// Получение ответа от сервера

byte[] responseBytes = new byte[1024];

int bytesRead = stream.Read(responseBytes, 0, responseBytes.Length);

string response = Encoding.UTF8.GetString(responseBytes, 0, bytesRead);

Console.WriteLine("Ответ от сервера: " + response);

// Закрытие подключения с сервером

client.Close();

}

catch (Exception ex)

{

Console.WriteLine("Ошибка клиента: " + ex.Message);

}

}

}

**Задание 2.**

**Напишите клиент-серверное передающие GIF, текст, изображение через IP и порт. В клиенте и сервере должно быть сообщение об успешной отправке, а также о получении.**

**Код (С#):**

using System;

using System.IO;

using System.Net;

using System.Net.Sockets;

using System.Text;

using System.Threading.Tasks;

public class Server

{

private const int Port = 12345;

public static async Task Main(string[] args)

{

// Создаем слушающий сокет для принятия входящих подключений

var listener = new TcpListener(IPAddress.Any, Port);

listener.Start();

Console.WriteLine($"Сервер запущен. Ожидание подключений на порту {Port}...");

while (true)

{

// Принимаем входящее подключение от клиента

var client = await listener.AcceptTcpClientAsync();

Console.WriteLine($"Принято подключение от {((IPEndPoint)client.Client.RemoteEndPoint).Address}:{((IPEndPoint)client.Client.RemoteEndPoint).Port}");

// Запускаем обработку подключения в отдельном потоке

Task.Run(() => HandleClient(client));

}

}

private static async void HandleClient(TcpClient client)

{

using (client)

{

try

{

// Получаем поток для чтения данных от клиента

var networkStream = client.GetStream();

var reader = new StreamReader(networkStream, Encoding.UTF8);

// Читаем данные от клиента - файл и тип передаваемого содержимого

var fileName = await reader.ReadLineAsync();

var contentType = await reader.ReadLineAsync();

var length = int.Parse(await reader.ReadLineAsync());

// Создаем FileStream для записи полученного файла

using (var fileStream = new FileStream(fileName, FileMode.Create))

{

// Читаем содержимое файла

var buffer = new byte[1024];

int bytesRead;

while (length > 0 && (bytesRead = await networkStream.ReadAsync(buffer, 0, Math.Min(buffer.Length, length))) > 0)

{

await fileStream.WriteAsync(buffer, 0, bytesRead);

length -= bytesRead;

}

}

// Отправляем успешное сообщение клиенту

var response = "Файл успешно принят.";

await SendResponse(client, response);

Console.WriteLine(response);

}

catch (Exception ex)

{

Console.WriteLine($"Ошибка при обработке подключения: {ex.Message}");

}

}

}

private static async Task SendResponse(TcpClient client, string message)

{

var writer = new StreamWriter(client.GetStream(), Encoding.UTF8);

await writer.WriteLineAsync(message);

await writer.FlushAsync();

}

}

public class Client

{

private const string ServerIp = "127.0.0.1";

private const int ServerPort = 12345;

public static async Task Main(string[] args)

{

// Считываем путь к файлу с диска

Console.Write("Введите путь к файлу: ");

var filePath = Console.ReadLine();

// Создаем клиентский сокет

using (var client = new TcpClient())

{

await client.ConnectAsync(ServerIp, ServerPort);

Console.WriteLine("Соединение с сервером успешно установлено.");

try

{

// Отправляем информацию о файле, его типе и размере

var fileInfo = new FileInfo(filePath);

var networkStream = client.GetStream();

var writer = new StreamWriter(networkStream, Encoding.UTF8);

await writer.WriteLineAsync(Path.GetFileName(filePath));

await writer.WriteLineAsync(GetContentType(filePath));

await writer.WriteLineAsync(fileInfo.Length.ToString());

await writer.FlushAsync();

// Отправляем содержимое файла

using (var fileStream = new FileStream(filePath, FileMode.Open))

{

var buffer = new byte[1024];

int bytesRead;

while ((bytesRead = await fileStream.ReadAsync(buffer, 0, buffer.Length)) > 0)

{

await networkStream.WriteAsync(buffer, 0, bytesRead);

}

}

// Получаем и выводим ответ от сервера

var reader = new StreamReader(networkStream, Encoding.UTF8);

var response = await reader.ReadLineAsync();

Console.WriteLine(response);

}

catch (Exception ex)

{

Console.WriteLine($"Ошибка при отправке файла: {ex.Message}");

}

}

}

private static string GetContentType(string filePath)

{

if (filePath.EndsWith(".gif", StringComparison.OrdinalIgnoreCase))

{

return "image/gif";

}

else if (filePath.EndsWith(".txt", StringComparison.OrdinalIgnoreCase))

{

return "text/plain";

}

else if (filePath.EndsWith(".jpg", StringComparison.OrdinalIgnoreCase) || filePath.EndsWith(".jpeg", StringComparison.OrdinalIgnoreCase))

{

return "image/jpeg";

}

else if (filePath.EndsWith(".png", StringComparison.OrdinalIgnoreCase))

{

return "image/png";

}

throw new NotSupportedException($"Формат файла {Path.GetFileName(filePath)} не поддерживается.");

}

}

В данном решении реализован простой клиент-серверный обмен файлами через TCP с использованием протокола IP и порта. Клиент и сервер могут передавать файлы различных типов, таких как гиф, текстовые файлы и изображения.

Код сервера:

1. Создается слушающий сокет на указанном порту (12345 в данном случае).

2. В цикле ожидается подключение от клиента.

3. При подключении запускается обработка входящего подключения в отдельном потоке.

4. В обработчике подключения принимается файл от клиента и сохраняется на диск.

5. Отправляется успешное сообщение клиенту о получении файла.

Код клиента:

1. Читается путь к файлу с диска.

2. Создается клиентский сокет и устанавливается соединение с сервером.

3. Отправляется информация о файле (имя, тип, размер) серверу.

4. Отправляется содержимое файла.

5. Получается ответ от сервера и выводится на экран.

**Задание 3. С чем взаимодействуют клиент-серверное приложение?**

Клиент-серверное взаимодействие — это процесс обмена данными между клиентом (обычно веб-браузером) и сервером.

Клиент-серверное приложение может взаимодействовать с пользователем, базами данных, другими серверами и клиентами.

**DEMO БИЛЕТ №9**

**Задание 1. Создать консольное приложение, которое передаёт вводимые символы в буфер.**

**Код:**

using System;

using System.Runtime.InteropServices;

class Program

{

[DllImport("user32.dll")]

public static extern bool OpenClipboard(IntPtr hWnd);

[DllImport("user32.dll")]

public static extern IntPtr GetClipboardData(uint uFormat);

[DllImport("user32.dll")]

public static extern bool CloseClipboard();

[DllImport("user32.dll", SetLastError = true)]

public static extern bool EmptyClipboard();

[DllImport("user32.dll", SetLastError = true)]

public static extern IntPtr SetClipboardData(uint uFormat, IntPtr hMem);

[DllImport("kernel32.dll")]

public static extern IntPtr GlobalAlloc(uint uFlags, UIntPtr dwBytes);

[DllImport("kernel32.dll")]

public static extern IntPtr GlobalLock(IntPtr hMem);

[DllImport("kernel32.dll")]

public static extern bool GlobalUnlock(IntPtr hMem);

[DllImport("kernel32.dll")]

public static extern IntPtr memcpy(IntPtr dest, IntPtr src, int count);

public const uint CF\_UNICODETEXT = 13;

static void Main()

{

Console.WriteLine("Введите текст:");

string input = Console.ReadLine();

// Копирование введенных символов в буфер обмена без использования System.Windows.Forms

if (OpenClipboard(IntPtr.Zero))

{

EmptyClipboard();

IntPtr hGlobalMem = GlobalAlloc(0x42, (UIntPtr)((input.Length + 1) \* 2)); // 0x42 - GMEM\_MOVEABLE

IntPtr lpGlobalMem = GlobalLock(hGlobalMem);

char[] chars = input.ToCharArray();

Marshal.Copy(chars, 0, lpGlobalMem, chars.Length);

GlobalUnlock(lpGlobalMem);

SetClipboardData(CF\_UNICODETEXT, hGlobalMem);

CloseClipboard();

Console.WriteLine("Текст успешно скопирован в буфер обмена.");

}

else

{

Console.WriteLine("Не удалось открыть буфер обмена.");

}

Console.ReadLine();

}

}

**Задание 2. Создать консольное приложение, которое считывает, что хранится в буфере и выводит на экран после запуска.**

**Код:**

using System;

using System.Runtime.InteropServices;

class Program

{

[DllImport("user32.dll")]

public static extern bool OpenClipboard(IntPtr hWnd);

[DllImport("user32.dll")]

public static extern IntPtr GetClipboardData(uint uFormat);

[DllImport("user32.dll")]

public static extern bool CloseClipboard();

[DllImport("kernel32.dll")]

public static extern IntPtr GlobalLock(IntPtr hMem);

[DllImport("kernel32.dll")]

public static extern bool GlobalUnlock(IntPtr hMem);

[DllImport("user32.dll", CharSet = CharSet.Unicode)]

public static extern int MessageBox(IntPtr hWnd, string text, string caption, uint type);

public const uint CF\_UNICODETEXT = 13;

static void Main()

{

if (OpenClipboard(IntPtr.Zero))

{

IntPtr hGlobalMem = GetClipboardData(CF\_UNICODETEXT);

if (hGlobalMem != IntPtr.Zero)

{

IntPtr lpGlobalMem = GlobalLock(hGlobalMem);

string clipboardText = Marshal.PtrToStringUni(lpGlobalMem);

GlobalUnlock(hGlobalMem);

MessageBox(IntPtr.Zero, clipboardText, "Текст из буфера обмена", 0);

}

else

{

MessageBox(IntPtr.Zero, "Буфер обмена не содержит текст.", "Ошибка", 0);

}

CloseClipboard();

}

else

{

MessageBox(IntPtr.Zero, "Не удалось открыть буфер обмена.", "Ошибка", 0);

}

}

}

**Задание 3.** **Что такое буфер экрана и как он работает?**

Буфер экрана - это область памяти на видеокарте, которая используется для временного хранения изображения перед его выводом на экран. Он работает следующим образом:

1. При запуске программы операционная система выделяет память для буфера экрана, который будет использоваться для хранения изображения.
2. Когда программа начинает рисовать на экране, она сначала записывает данные в буфер экрана.
3. Видеокарта периодически считывает данные из буфера экрана и выводит их на экран, обновляя изображение.
4. Если программа изменяет изображение на экране, например, рисует новый объект, видеокарта снова считывает данные из буфера и обновляет изображение на экране.

**DEMO БИЛЕТ №10**

**Задание 1. Написать консольную программу, которая создает каталог, после чего создается файл. В него записывается текст, новый текст помещается строчкой ниже и в конце вывод по следующим свойствам время создания каталога, время последнего изменения каталога, имя каталога. После чего добавить файлы в папку и вывести список файлов в каталоге.**

**Код:**

using System;

using System.IO;

class Program

{

static void Main()

{

// Получаем текущую дату и время

DateTime currentTime = DateTime.Now;

// Создаем каталог с уникальным именем, основанным на текущей дате и времени

string directoryName = currentTime.ToString("yyyyMMdd\_HHmmss");

Directory.CreateDirectory(directoryName);

// Создаем файл в новом каталоге

string fileName = Path.Combine(directoryName, "file.txt");

File.Create(fileName).Close();

// Записываем текст в файл

File.WriteAllText(fileName, "Начальный текст");

// Добавляем новый текст в файл

File.AppendAllText(fileName, "\nНовый текст");

// Выводим информацию о каталоге

Console.WriteLine("Информация о каталоге:");

Console.WriteLine($"Время создания: {Directory.GetCreationTime(directoryName)}");

Console.WriteLine($"Время последнего изменения: {Directory.GetLastWriteTime(directoryName)}");

Console.WriteLine($"Имя каталога: {directoryName}");

// Выводим список файлов в каталоге

Console.WriteLine("\nФайлы в каталоге:");

string[] files = Directory.GetFiles(directoryName);

foreach (string file in files)

{

Console.WriteLine(file);

}

}

}

**Задание 2.** Написать программу используя графический интерфейс. Цель программы открытия файла и записи в него отредактированного текста, формат txt. Главной задачей разработать диалоговое окно выбора файла и, если файл не выбран создать новый.

**Код:**

using System;

using System.IO;

using System.Windows.Forms;

namespace FileEditor

{

public partial class MainForm : Form

{

private string filePath;

public MainForm()

{

InitializeComponent();

}

private void openFileButton\_Click(object sender, EventArgs e)

{

OpenFileDialog openFileDialog = new OpenFileDialog();

openFileDialog.Filter = "Text Files (\*.txt)|\*.txt";

openFileDialog.Title = "Open Text File";

if (openFileDialog.ShowDialog() == DialogResult.OK)

{

filePath = openFileDialog.FileName;

fileLabel.Text = Path.GetFileName(filePath);

LoadFileContents();

}

else

{

// If no file is selected, create a new file

filePath = null;

fileLabel.Text = "New File";

richTextBox.Clear();

}

}

private void saveFileButton\_Click(object sender, EventArgs e)

{

if (!string.IsNullOrEmpty(filePath))

{

SaveFileContents();

}

else

{

SaveFileDialog saveFileDialog = new SaveFileDialog();

saveFileDialog.Filter = "Text Files (\*.txt)|\*.txt";

saveFileDialog.Title = "Save Text File";

if (saveFileDialog.ShowDialog() == DialogResult.OK)

{

filePath = saveFileDialog.FileName;

fileLabel.Text = Path.GetFileName(filePath);

SaveFileContents();

}

}

}

private void LoadFileContents()

{

richTextBox.Text = File.ReadAllText(filePath);

}

private void SaveFileContents()

{

File.WriteAllText(filePath, richTextBox.Text);

MessageBox.Show("File saved successfully.");

}

}

}

**Задание 3.** **Какие свойства существуют для управления файловой системой?**

Для управления файловой системой в .NET доступны следующие свойства:

**Directory.GetFiles()** - возвращает массив файлов из указанного каталога.  
**Directory.GetDirectories()** - возвращает массив подкаталогов из указанного каталога.  
**File.Exists()** - проверяет, существует ли указанный файл.  
**FileInfo.Length** - получает длину файла в байтах.

**Path.GetFullPath()** - получает полный путь к файлу или каталогу.  
**Path.GetDirectoryName()** - получает путь к каталогу, без имени файла.  
**Path.GetFileName()** - получает имя файла без пути.

**DEMO БИЛЕТ №11**

**Задание 1. Напишите программу используя сокет для установки соединения и обмена данными между клиентом и сервером. Программа должна использовать библиотеку сокетов и демонстрирует основные операции с сокетами, такие как создание, связывание и прослушивание на определенном порту.**

**Код:**

using System;

using System.Net;

using System.Net.Sockets;

using System.Text;

class Program

{

static void Main()

{

// Устанавливаем IP-адрес и порт сервера

string serverIP = "127.0.0.1";

int serverPort = 12345;

// Создаем точку подключения для сервера

IPEndPoint serverEndPoint = new IPEndPoint(IPAddress.Parse(serverIP), serverPort);

// Создаем сокет TCP/IP

Socket serverSocket = new Socket(AddressFamily.InterNetwork, SocketType.Stream, ProtocolType.Tcp);

try

{

// Связываем серверный сокет с точкой подключения

serverSocket.Bind(serverEndPoint);

// Начинаем прослушивать входящие соединения

serverSocket.Listen(10);

Console.WriteLine("Сервер запущен. Ожидание клиента...");

// Принимаем входящие соединения от клиентов

Socket clientSocket = serverSocket.Accept();

Console.WriteLine("Клиент подключен.");

// Буфер для получения и отправки данных

byte[] buffer = new byte[1024];

// Читаем данные от клиента

int bytesRead = clientSocket.Receive(buffer);

string requestData = Encoding.ASCII.GetString(buffer, 0, bytesRead);

Console.WriteLine($"Получено от клиента: {requestData}");

// Отправляем ответ клиенту

string responseData = "Привет от сервера!";

byte[] responseBuffer = Encoding.ASCII.GetBytes(responseData);

clientSocket.Send(responseBuffer);

// Закрываем соединение с клиентом

clientSocket.Shutdown(SocketShutdown.Both);

clientSocket.Close();

Console.WriteLine("Соединение с клиентом закрыто.");

}

catch (Exception ex)

{

Console.WriteLine($"Произошла ошибка: {ex.Message}");

}

finally

{

// Закрываем серверный сокет

serverSocket.Close();

}

}

}

**Задание 2. При помощи командной строки проанализируйте работу подсистем управления ресурсов.**

**Код:**

// Здесь мы можем разработать решение для анализа работу подсистем управления ресурсов

// Импортируем необходимые пространства имен

using System;

using System.Diagnostics;

class Program

{

static void Main()

{

// Создаем новый процесс для запуска командной строки

Process process = new Process();

// Задаем параметры для запуска командной строки

ProcessStartInfo startInfo = new ProcessStartInfo();

startInfo.FileName = "cmd.exe";

startInfo.Arguments = "/C <команды для анализа работы подсистем управления ресурсами>";

startInfo.RedirectStandardOutput = true;

startInfo.UseShellExecute = false;

startInfo.CreateNoWindow = true;

// Назначаем параметры запуска для процесса

process.StartInfo = startInfo;

// Запускаем процесс и получаем результат

process.Start();

string output = process.StandardOutput.ReadToEnd();

// Ожидаем завершения процесса

process.WaitForExit();

// Выводим результаты анализа работы подсистем управления ресурсами

Console.WriteLine(output);

}

}

**Задание 3. Как работает клиент-серверная модель?**

Есть клиент, который отправляет запрос серверу. Сервер обрабатывает запрос и отправляет ответ клиенту. Клиент может быть программой на компьютере пользователя или веб-браузером. Сервер может быть веб-сервером, сервером баз данных или другим сервером, предоставляющим услуги по обработке запросов.

**DEMO БИЛЕТ №12**

**Задание 1. Напишите консольное приложение, которая в созданный файл numbers.txt записывает натуральные числа от 1 до 500 через запятую.**

**Код:**

using System;

using [System.IO](https://vk.com/away.php?to=http%3A%2F%2FSystem.IO&cc_key=);

class Program {

static void Main()

{

string filePath = "numbers.txt";

string numbers = string.Join(",",

Enumerable.Range(1, 500));

File.WriteAllText(filePath, numbers);

}

}

**Задание 2. Дан массив строк: "red", "green", "black", "white", "blue". Запишите в файл элементы массива построчно (к55аждый элемент в новой строке). При помощи консольного приложения**

**Код:**

using System;  
 using [System.I](https://vk.com/away.php?to=http%3A%2F%2FSystem.IO&cc_key=)О;  
  
 class Program  
 {  
 static void Main()  
 {  
 string[] colors = { "red", "green", "black", "white", "blue" };  
  
 string path = "colors.txt";  
  
 using (StreamWriter writer = new StreamWriter(path))  
 {  
 foreach (string color in colors)  
 {  
 writer.WriteLine(color);  
 }  
 }  
  
 Console.WriteLine("Элементы массива были успешно записаны в файл.");  
 Console.ReadLine();  
 }  
 }

**Задание 3. Какие существуют основные методы для управления каталогам и файлами?**

**1.** Создание каталога или файла: Для этого можно использовать методы, предоставляемые операционной системой или языком программирования, например, функции mkdir().

**2.** Переименование и перемещение: Используйте методы, поддерживаемые операционной системой или языком программирования, чтобы переименовать или переместить файлы и каталоги. Например, функция rename() или os.rename().

**3.** Копирование и удаление: Используйте методы для копирования файлов и каталогов из одного места в другое и для удаления ненужных файлов и каталогов. Например, функции copy() и remove() или shutil.copy() и shutil.rmtree().

**4.** Просмотр содержимого каталога: Используйте методы для просмотра содержимого каталога, чтобы получить список файлов и подкаталогов в данном каталоге. Например, функция listdir().

**5.** Проверка существования и типа: Можно использовать методы для проверки существования файлов или каталогов и определения их типа (файл или каталог). Например, функции exists() или isdir().

**6.** Работа с атрибутами файлов: Используйте методы для получения или установки различных атрибутов файлов, таких как права доступа, дата создания, размер и т.д. Например, функции chmod(), ctime(), getsize().

Обратите внимание, что конкретные методы и функции будут зависеть от выбранного вами языка программирования и операционной системы. Это всего лишь общий обзор доступных методов.

**System.IO.File:** Класс File предоставляет статические методы для работы с файлами, включая создание, чтение, запись, удаление, перемещение файлов и получение информации о файлах.

**Примеры методов:**

* File.Exists() - проверяет существование файла
* File.ReadAllLines() - читает все строки из файла
* File.WriteAllText() - записывает текст в файл

**System.IO.Directory:** Класс Directory предоставляет методы для управления каталогами, такие как создание, удаление, перемещение каталогов.

**Примеры методов:**

* Directory.CreateDirectory() - создает новый каталог
* Directory.Delete() - удаляет каталог
* Directory.GetDirectories() - возвращает список подкаталогов
* Directory.GetFiles() - возвращает список файлов в каталоге

**DEMO БИЛЕТ №13**

**Задание 1.** Написать консольное приложение, которое при запуске записывает текст в файл, новый текст помещается на строчку ниже.

**Код:**

using System;

using System.IO;

namespace FileWriting

{

class Program

{

static void Main(string[] args)

{

string filePath = "testfile.txt";

Console.WriteLine("Введите текст для записи в файл:");

string inputText = Console.ReadLine();

using (StreamWriter writer = File.AppendText(filePath))

{

writer.WriteLine(inputText);

}

Console.WriteLine("Текст успешно записан в файл.");

}

}

}

**Задание 2.** Написать консольную программу, которая создает каталог(папку) после чего создается файл в него записывается текст, новый текст помещается строчкой ниже и в конце вывод по следующим свойствам время создания каталога, время последнего изменения каталога, имя каталога. После чего добавить файлы в папку и вывести список файлов в каталоге.

**Код:**

using System;

using [System.IO](https://vk.com/away.php?to=http%3A%2F%2FSystem.IO&cc_key=);

class Program

{

static void Main()

{

string newDirectory = "C:\\Users\\Student\\Desktop\\PR1\\Vivod\\DoneFolder";

if (!Directory.Exists(newDirectory))

{

Directory.CreateDirectory(newDirectory);

}

Console.WriteLine("Введите текст:");

string userInput = Console.ReadLine();

string filePath = Path.Combine(newDirectory, "example\_file.txt");

using (StreamWriter writer = File.CreateText(filePath))

{

string imagePath = Path.Combine(newDirectory, "example\_image.jpg");

File.Copy("C:\\Users\\Student\\Desktop\\PR1\\1.jpg", imagePath);

DateTime createdTime = Directory.GetCreationTime(newDirectory);

DateTime modifiedTime = Directory.GetLastWriteTime(newDirectory);

writer.WriteLine(userInput);

writer.WriteLine($"Время создания каталога: {createdTime}");

writer.WriteLine($"Время последнего изменения каталога: {modifiedTime}");

writer.WriteLine($"Имя каталога: {newDirectory}");

}

File.WriteAllText(Path.Combine(newDirectory, "new\_file1.txt"), "Текст для нового файла 1");

string[] fileList = Directory.GetFiles(newDirectory);

Console.WriteLine("Список файлов в каталоге:");

foreach (string file in fileList)

{

Console.WriteLine(file);

}

}

**Задание 3. Какие методы управляют чтением, записью и сохранением каталогам и файлами?**

В C# для управления чтением, записью и сохранением каталогов и файлов используются различные классы и методы из пространства имен System.IO. Вот некоторые из основных методов:

**1.** Чтение файла:

* File.ReadAllBytes(string path) - читает все байты из файла в виде массива байт.
* File.ReadAllText(string path) - читает все строки из файла и возвращает их в виде строки.
* File.ReadAllLines(string path) - читает все строки из файла и возвращает их в виде массива строк.
* FileStream - можно использовать для чтения файла по частям или для других специфических операций.

**2.** Запись в файл:

* File.WriteAllText(string path, string contents) - записывает указанный текст в файл.
* File.WriteAllLines(string path, string[] lines) - записывает массив строк в файл.
* FileStream - можно использовать для записи массива байтов в файл или для других более сложных операций записи.

**3.** Создание и удаление файлов и каталогов:

* File.Create(string path) - создает файл на указанном пути.
* Directory.CreateDirectory(string path) - создает каталог или директорию.
* File.Delete(string path) - удаляет файл.
* Directory.Delete(string path, bool recursive) - удаляет каталог (рекурсивно удаляет все файлы и подкаталоги).

**DEMO БИЛЕТ №14**

**Задание 1.** Н**апишите программу управления интерфейсам мыши таким образом, чтобы при запуске программы мышка была по центру.**

**Код:**

using System;

using System.Runtime.InteropServices;

class Program

{

// Импорт функции из user32.dll для работы с курсором

[DllImport("user32.dll", CharSet = [CharSet.Auto](https://vk.com/away.php?to=http%3A%2F%2FCharSet.Auto&cc_key=), CallingConvention = CallingConvention.StdCall)]

public static extern void mouse\_event(uint dwFlags, uint dx, uint dy, uint cButtons, uint dwExtraInfo);

// Импорт функции GetSystemMetrics из user32.dll для получения информации о системе

[DllImport("user32.dll")]

public static extern int GetSystemMetrics(int nIndex);

static void Main()

{

// Константа для флага перемещения курсора

const int MOUSEEVENTF\_MOVE = 0x0001;

// Константа для флага указания абсолютной позиции курсора

const int MOUSEEVENTF\_ABSOLUTE = 0x8000;

// Вывод сообщения на экран

Console.WriteLine("Press Enter to hide the cursor.");

// Ожидание нажатия клавиши Enter

Console.ReadLine();

while (true)

{

// Получение ширины и высоты экрана с помощью GetSystemMetrics

int screenWidth = GetSystemMetrics(0);

// Индекс 0 соответствует ширине экрана

int screenHeight = GetSystemMetrics(1);

// Вычисление центра экрана по X

int centerX = screenWidth / 2;

//Вычисление центра экрана по Y

int centerY = screenHeight / 2;

// Перемещение курсора в центр экрана

mouse\_event(MOUSEEVENTF\_MOVE | MOUSEEVENTF\_ABSOLUTE, (uint)(centerX \* 65535 / screenWidth), (uint)(centerY \* 65535 / screenHeight), 0, 0);

// Ожидание нажатия клавиши Enter для продолжения

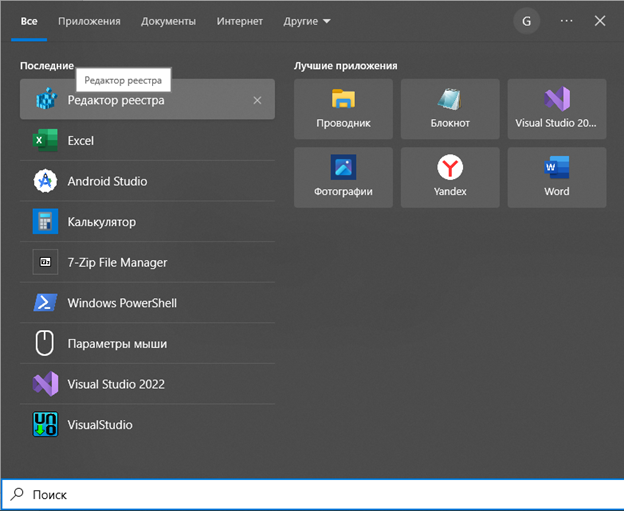
Console.ReadLine();

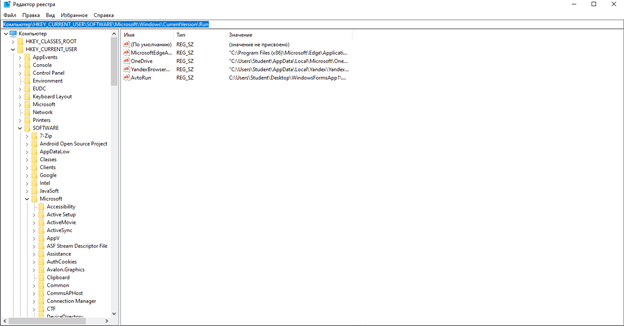
}

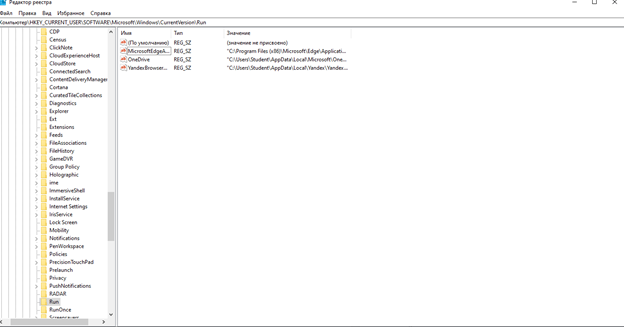
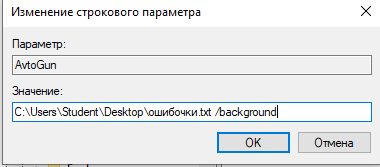
}

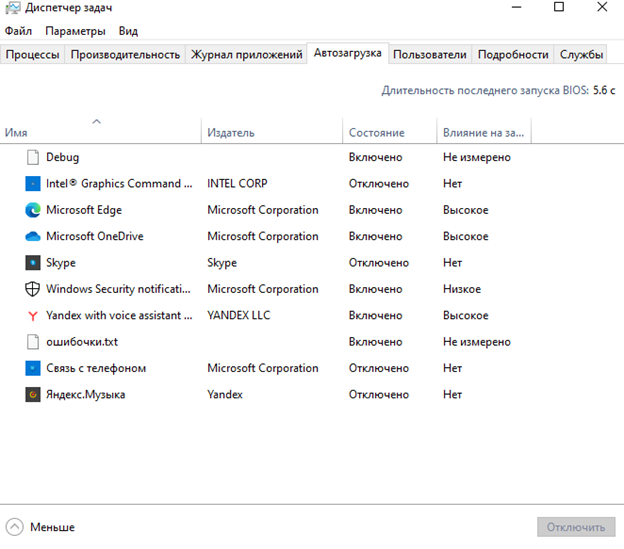
}

**Задание 2. Добавить программу в автозагрузку системного реестра с добавлением свойств.**









**Задание 3. Что такое системный реестр?**

**Системный реестр** - это база данных, в которой хранятся настройки операционной системы, приложений и пользователей. Он состоит из разделов, которые содержат ключи и значения. Системный реестр используется для хранения информации о конфигурации системы, настройках приложений, пользовательских настройках и других параметрах.

**DEMO БИЛЕТ №15**

**Задание 1. Напишите консольную программу, которая выводит список подключенных устройств.**

**Код:**

using System;  
using [System.Management](https://vk.com/away.php?to=http%3A%2F%2FSystem.Management&cc_key=);  
  
 namespace HardwareInfo  
 {  
 class Program  
 {  
 static void Main(string[] args)  
 {  
  
 ManagementObjectSearcher searcher = new ManagementObjectSearcher("SELECT \* FROM Win32\_Processor");  
 ManagementObjectCollection processors = searcher.Get();  
 foreach (ManagementObject processor in processors)  
 {  
 Console.WriteLine($"Процессор: {processor["Name"]}");  
 }  
  
 searcher = new ManagementObjectSearcher("SELECT \* FROM Win32\_PhysicalMemory");  
 ManagementObjectCollection memories = searcher.Get();  
 foreach (ManagementObject memory in memories)  
 {  
 Console.WriteLine($"Оперативная память: {memory["Name"]}");  
  
 }  
  
 searcher = new ManagementObjectSearcher("SELECT \* FROM Win32\_DiskDrive");  
 ManagementObjectCollection disks = searcher.Get();  
 foreach (ManagementObject disk in disks)  
 {  
 Console.WriteLine($"Диск: {disk["Name"]}");  
  
 }  
 }  
 }  
 }

**Задание 2. Добавить собственную программу в автозагрузку системного реестра с добавлением свойств.**

1. Открыл редактор реестра и открыл папку по пути: \HKEY\_CURRENT\_USER\SOFTWARE\Microsoft\Windows\CurrentVersion\Run
2. Нажал ПКМ, создать и выбрал строковый параметр
3. Создал строковый параметр с названием lb7, и в значение добавил путь до VScode, также в значение добавил параметр /background

**Задание 3. Какие ветви существуют в системном реестре и за что они отвечают?**

В системном реестре Windows существует несколько основных ветвей, каждая из которых отвечает за определенные функции и хранит соответствующие данные. Ниже представлены некоторые из основных ветвей системного реестра и их назначение:

1. HKEY\_CLASSES\_ROOT (HKCR) - содержит информацию об ассоциациях между расширениями файлов и соответствующими приложениями, а также информацию о зарегистрированных COM объектах.
2. HKEY\_CURRENT\_USER (HKCU) - содержит настройки и параметры пользователя, который в данный момент вошел в систему, такие как настройки рабочего стола, темы и т.д.
3. HKEY\_LOCAL\_MACHINE (HKLM) - содержит информацию о компьютере и его аппаратном обеспечении, установленных программах, настройках операционной системы и службах Windows.
4. HKEY\_USERS (HKU) - содержит информацию о всех пользователях, зарегистрированных в системе, и их индивидуальные настройки.
5. HKEY\_CURRENT\_CONFIG (HKCC) - содержит информацию о текущей конфигурации аппаратного обеспечения и параметры отображения.
6. HKEY\_PERFORMANCE\_DATA - содержит информацию о производительности системы.
7. HKEY\_CURRENT\_CONFIG - содержит информацию о текущей конфигурации аппаратного обеспечения и параметры отображения.

Каждая из этих ветвей играет важную роль в работе операционной системы Windows и приложений, храня информацию о настройках, параметрах, программных и аппаратных компонентах системы.

**DEMO БИЛЕТ №16**

**Задание 1. Написать консольную программу, которая выводит список подключенных устройств в разных портах и отправляет эту информацию в текстовый файл на пк.**

**Код:**

using System;

using System.Collections.Generic;

using [System.IO](https://vk.com/away.php?to=http%3A%2F%2FSystem.IO&cc_key=);

using System.Linq;

using [System.Management](https://vk.com/away.php?to=http%3A%2F%2FSystem.Management&cc_key=);

using System.Text;

using System.Threading.Tasks;

namespace ConsoleApp4

{

internal class Program

{

static void Main(string[] args)

{

ManagementObjectCollection collection;

var searcher = new ManagementObjectSearcher(@"Select \* From Win32\_PnPEntity");

collection = searcher.Get();

FileStream fs = new FileStream("C:\\Users\\админ\\Desktop\\connected\_devices.txt", FileMode.OpenOrCreate);

StreamWriter writer = new StreamWriter(fs);

foreach (var item in collection)

{

writer.WriteLine($"Устройство: {item["Caption"]}");

writer.WriteLine($"Описание: {item["Description"]}");

writer.WriteLine($"ID девайса: {item["DeviceID"]}");

writer.WriteLine($"Статус: {item["Status"]}");

writer.WriteLine("=========================================");

}

writer.Close();

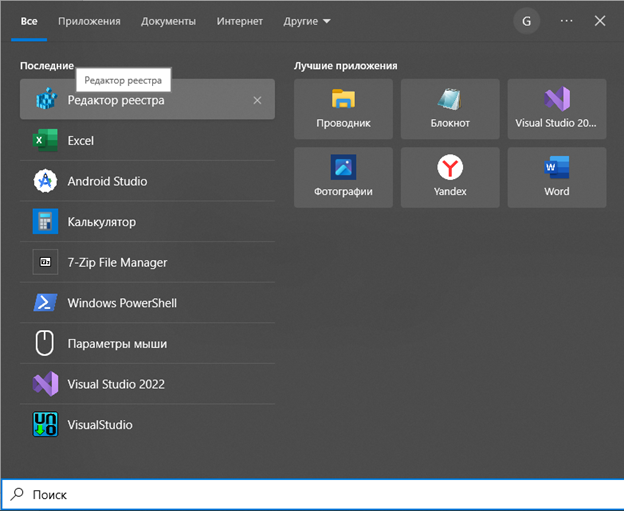
collection.Dispose();

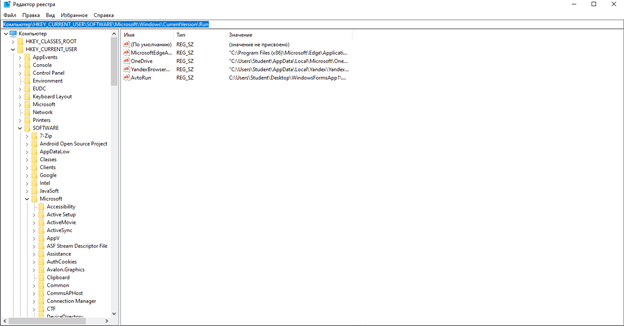
}

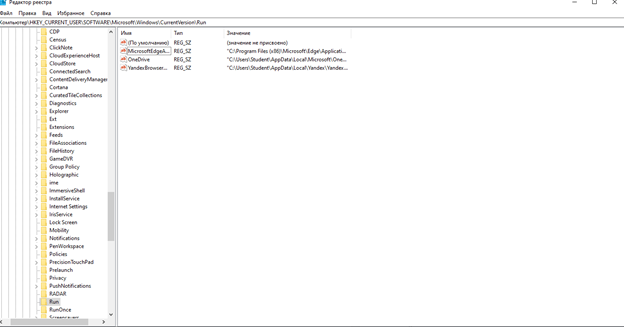
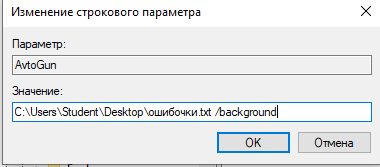
}

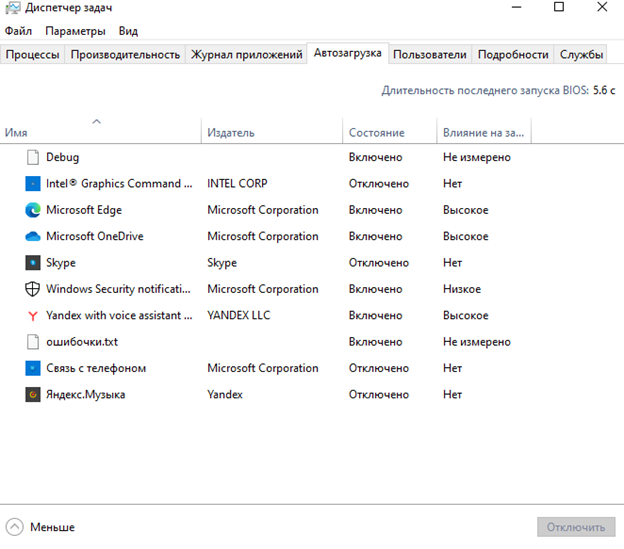
}

**Задание 2. Добавить программу в автозагрузку системного реестра с добавлением свойств.**









**Задание 3. Для чего нужна библиотека System.Management?**

Библиотека System.Management используется для управления устройствами в сети и получения информации о них. Она позволяет получить данные о состоянии устройств, их настройках, событиях и ошибках. Также с помощью этой библиотеки можно изменять настройки устройств и автоматизировать задачи управления.

**DEMO БИЛЕТ №17**

**Задание 1. Написать консольную программу, которая выводит список подключенного оборудования в окне консоли.**

**Код:**

using System;

using System.Management;

namespace DeviceList

{

class Program

{

static void Main(string[] args)

{

ManagementObjectSearcher searcher = new ManagementObjectSearcher("SELECT \* FROM Win32\_PnPEntity");

Console.WriteLine("Список подключенного оборудования:");

foreach (ManagementObject queryObj in searcher.Get())

{

Console.WriteLine("Устройство: " + queryObj["Caption"]);

}

}

}

}

**Задание 2. Написать консольное приложение для проверки работоспособности клавиатуры (можно использовать разные методы и подходы)**

**Код:**

class Program  
 {  
 static void Main(string[] args)  
 {  
 Console.WriteLine("Нажмите любую клавишу, чтобы начать тест...");  
 Console.ReadKey();  
  
 string keys = "`1234567890-  
 =qwertyuiop[]\\asdfghjkl;'zxcvbnm,./~!@#$%^&\*()\_+QWERTYUIOP{}|ASDFGHJKL:\"ZXCVBNM<>?\r\n";  
 int currentKeyIndex = 0;  
 while (currentKeyIndex < keys.Length)  
 {  
 Console.WriteLine($"\nНажмите клавишу '{keys[currentKeyIndex]}':");  
 ConsoleKeyInfo keyInfo = Console.ReadKey();  
  
 if (keyInfo.KeyChar != keys[currentKeyIndex])  
 {  
 Console.WriteLine("\nОшибка: Неправильно нажата клавиша.");  
 Console.WriteLine("Тест не пройден.");  
 return;  
 }  
  
 currentKeyIndex++;  
 }  
  
 Console.WriteLine("\nУспех: тест клавиатуры пройден!");  
 }  
 }

**Задание 3. Что выполняет SELECT \* FROM Win32 DiskDrive?**

Запрос SELECT \* FROM Win32\_DiskDrive используется только в C# для получения информации о физических дисках компьютера. Результатом выполнения этого запроса будет набор данных о каждом диске, таких как модель, серийный номер, емкость, интерфейс и другие характеристики.

**DEMO БИЛЕТ №18**

**Задание 1. Написать программу, которая выполняет действия над графическим интерфейсом пользовательского окна.**

using System;

using System.Runtime.InteropServices;

class Program

{

[DllImport("user32.dll")]

static extern bool SystemParametersInfo(uint uiAction, uint uiParam, IntPtr pvParam, uint fWinIni);

static void SetCursorSpeedSettings(int speed)

{

const uint SPIF\_UPDATEINIFILE = 0x01;

SystemParametersInfo(113, 0, (IntPtr)speed, SPIF\_UPDATEINIFILE);

}

static void Main()

{

Console.WriteLine("Введите новую скорость курсора (от 1 до 20):");

while (true)

{

string input = Console.ReadLine();

if (int.TryParse(input, out int speed))

{

if (speed >= 1 && speed <= 20)

{

SetCursorSpeedSettings(speed);

Console.WriteLine($"Скорость курсора изменена на {speed}. Для выхода нажмите клавишу 'Q'.");

}

else

{

Console.WriteLine("Введите число от 1 до 20.");

}

}

else

{

if (input.ToLower() == "q")

break;

Console.WriteLine("Некорректный ввод. Введите число от 1 до 20.");

}

}

}

}

**Задание 2. Написать консольное приложение для изменения цвета текста в зависимости от выбора пользователя.**

**Код:**

using System;

class Program

{

static void Main()

{

Console.WriteLine("Выберите цвет текста:");

Console.WriteLine("1. Красный");

Console.WriteLine("2. Зеленый");

Console.WriteLine("3. Синий");

int choice = int.Parse(Console.ReadLine());

switch (choice)

{

case 1:

Console.ForegroundColor = ConsoleColor.Red;

break;

case 2:

Console.ForegroundColor = ConsoleColor.Green;

break;

case 3:

Console.ForegroundColor = ConsoleColor.Blue;

break;

default:

Console.WriteLine("Некорректный выбор. Установлен цвет по умолчанию.");

break;

}

Console.WriteLine("Текст с выбранным цветом.");

Console.ResetColor(); // Сброс цвета обратно на стандартный

Console.WriteLine("Программа завершена. Нажмите любую клавишу для выхода.");

Console.ReadKey();

}

}

**Задание 3. Для чего необходим пакет расширения NuGet?**

**NuGet** - это расширение (надстройка) для простого добавления, удаления и обновления библиотек и ресурсов в проектах Visual Studio на основе .NET Framework, .NET Core и .NET.

**DEMO БИЛЕТ №19**

**Задание 1. Написать программу winForm, которая открывает файл, при помощи инструментария позволяет, редактировать и сохранять текстовую информацию. Формат файла TXT.**

**Код:**

namespace SimpleNotepad

{

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

}

private void openToolStripMenuItem\_Click(object sender, EventArgs e)

{

richTextBox1.Clear();

}

private void openToolStripMenuItem1\_Click(object sender, EventArgs e)

{

if (openFileDialog1.ShowDialog() ==

System.Windows.Forms.DialogResult.OK &&

openFileDialog1.FileName.Length > 0)

{

try

{

richTextBox1.LoadFile(openFileDialog1.FileName,

RichTextBoxStreamType.RichText);

}

catch (System.ArgumentException ex)

{

richTextBox1.LoadFile(openFileDialog1.FileName,

RichTextBoxStreamType.PlainText);

}

this.Text = "Файл[" + openFileDialog1.FileName + "]";

}

}

private void fileToolStripMenuItem\_Click(object sender, EventArgs e)

{

}

}

}

**Задание 2. Написать консольную программу, которая вывод список подключенных устройств в разных портах и отправляет эту информацию в текстовый файл с названием 1.txt**

**Код:**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Management;

using System.IO.Compression;

namespace ConsoleApp1

{

internal class Program

{

static void Main(string[] args)

{

string outputFilePath = @"C:\Users\Student\Desktop\mit.txt";

using (StreamWriter outputFile = new StreamWriter(outputFilePath))

{

ManagementObjectCollection collection;

using (var searcher = new ManagementObjectSearcher(@"Select \* From Win32\_PnPEntity"))

collection = searcher.Get();

foreach (ManagementObject device in collection)

{

outputFile.WriteLine("Device name:" + device["name"]);

outputFile.WriteLine("Device description:" + device["description"]);

outputFile.WriteLine("Device status:" + device["status"]);

outputFile.WriteLine("\*\*\*\*\*");

}

Console.WriteLine("Информация записана в файл");

}

}

}

}

**Задание 3. В чем отличие C://admin//sad от @C:/admi/sad?**

Во втором случае программа использует меньше байт.

**DEMO БИЛЕТ №20**

**Задание 1. Написать приложение приема сообщения от пользователей, в соответствии с стандартами серверных приложений.**

**Код:**

using System;

using System.Net;

using System.Net.Sockets;

using System.Text;

class Server

{

static void Main()

{

StartServer();

}

static void StartServer()

{

IPAddress ipAddress = IPAddress.Parse("127.0.0.1");

int port = 8888;

TcpListener server = new TcpListener(ipAddress, port);

server.Start();

Console.WriteLine("Сервер запущен. Ожидание подключений...");

while (true)

{

TcpClient client = server.AcceptTcpClient();

Console.WriteLine("Клиент подключен.");

NetworkStream stream = client.GetStream();

byte[] data = new byte[256];

StringBuilder message = new StringBuilder();

int bytes;

do

{

bytes = stream.Read(data, 0, data.Length);

message.Append(Encoding.UTF8.GetString(data, 0, bytes));

} while (stream.DataAvailable);

Console.WriteLine("Получено сообщение от клиента: " + message);

client.Close();

Console.WriteLine("Клиент отключен.");

}

}

}

**Задание 2. Написать приложение отправления сообщения от пользователя, в соответствии с стандартами клиент-серверных приложений.**

using System;

using System.Net;

using System.Net.Sockets;

using System.Text;

class Client

{

static void Main()

{

StartClient();

}

static void StartClient()

{

try

{

TcpClient client = new TcpClient("127.0.0.1", 8888);

Console.WriteLine("Подключено к серверу.");

NetworkStream stream = client.GetStream();

Console.Write("Введите сообщение: ");

string message = Console.ReadLine();

byte[] data = Encoding.UTF8.GetBytes(message);

stream.Write(data, 0, data.Length);

Console.WriteLine("Сообщение отправлено.");

client.Close();

}

catch (Exception ex)

{

Console.WriteLine("Ошибка: " + ex.Message);

}

}

}

**Задание 3. В чем отличие клиентского программного обеспечения от серверного?**

Основное различие между клиентской и серверной сторонами программирования заключается в том, где выполняется код программы. Клиентская сторона программирования выполняется на устройстве пользователя и отвечает за отображение данных и взаимодействие с пользователем. Серверная сторона программирования, напротив, выполняется на удаленном сервере и отвечает за обработку запросов, работу с базами данных и обеспечение безопасности данных.

**DEMO БИЛЕТ №21**

**Задание 1. Написать программу приводящая процессы и потоки к состоянию гонки за ресурсами.**

**Код:**

using System;

using System.Threading;

namespace RaceConditionApp

{

class Program

{

static int resource = 0;

static void Main(string[] args)

{

Thread thread1 = new Thread(IncrementResource);

Thread thread2 = new Thread(DecrementResource);

thread1.Start();

thread2.Start();

thread1.Join();

thread2.Join();

Console.WriteLine("Значение ресурса: " + resource);

}

static void IncrementResource()

{

for (int i = 0; i < 1000000; i++)

{

resource++;

}

}

static void DecrementResource()

{

for (int i = 0; i < 1000000; i++)

{

resource--;

}

}

}

}

**Задание 2. Написать программу использующая взаимная блокировку Deadlock в управлении процессов.**

**Код:**

using System;

using System.Threading;

class Program

{

static object lock1 = new object();

static object lock2 = new object();

static void Main()

{

Thread thread1 = new Thread(Process1);

Thread thread2 = new Thread(Process2);

thread1.Start();

thread2.Start();

}

static void Process1()

{

lock (lock1)

{

Console.WriteLine("Процес 1 захватил ресурс 1");

Thread.Sleep(1000); // Даем время Process2 захватить lock2

Console.WriteLine("Процес 1 ожидает ресурс 2");

lock (lock2)

{

Console.WriteLine("Процес 1 захватил ресурс 2");

}

}

}

static void Process2()

{

lock (lock2)

{

Console.WriteLine("Процес 2 захватил ресурс 2");

Thread.Sleep(1000); // Даем время Process1 захватить lock1

Console.WriteLine("Процес 2 ожидает ресурс 1");

lock (lock1)

{

Console.WriteLine("Процес 2 захватил ресурс 1");

}

}

}

}

**Задание 3. Какие есть методы блокировки потоков и процессов?**

**Lock Statement:**

Используется для создания критической секции кода, в рамках которой только один поток может выполняться.

**Monitor Class:**

Предоставляет более гибкий способ управления блокировками, чем lock.

Методы Monitor.Enter и Monitor.Exit используются для захвата и освобождения блокировки.

**Mutex Class:**

Объект Mutex используется для управления доступом к ресурсам между процессами.

Позволяет синхронизировать доступ к общему ресурсу из разных процессов.

**Semaphore Class:**

Позволяет ограничивать количество потоков, имеющих доступ к общему ресурсу.

**DEMO БИЛЕТ №22**

**Задание 1. Написать программу, которая реализуют покупку товаров по чек листу в магазине.**

**Код:**

using System;

using System.Collections.Generic;

class Program

{

static void Main()

{

// Создание чек-листа товаров

Dictionary<string, double> priceList = new Dictionary<string, double>()

{

{ "Молоко", 1.5 },

{ "Хлеб", 0.8 },

{ "Яблоки", 2.0 },

{ "Картофель", 1.0 }

};

// Вывод списка товаров и цен

Console.WriteLine("Список товаров и цен:");

foreach (var item in priceList)

{

Console.WriteLine($"{item.Key} - {item.Value}$");

}

// Покупка товаров

double totalCost = 0;

Console.WriteLine("\nВведите товары для покупки (для завершения введите 'конец'):");

while (true)

{

string item = Console.ReadLine();

if (item.ToLower() == "конец")

{

break;

}

if (priceList.ContainsKey(item))

{

totalCost += priceList[item];

Console.WriteLine($"Добавлен товар '{item}' в чек. Текущая сумма покупки: {totalCost}$");

}

else

{

Console.WriteLine("Такого товара нет в списке.");

}

}

// Вывод общей суммы покупки

Console.WriteLine($"\nОбщая сумма покупки: {totalCost}$");

}

}

**Задание 2. Написать программу ограничения доступа к переменной так, чтобы только один поток в один момент времени мог изменять переменную.**

**Код:**

using System;

using System.Threading;

class Program

{

static int sharedVariable = 0;

static object lockObject = new object();

static void Main()

{

Thread thread1 = new Thread(IncrementVariable);

Thread thread2 = new Thread(DecrementVariable);

thread1.Start();

thread2.Start();

thread1.Join();

thread2.Join();

Console.WriteLine("Итоговое значение переменной: " + sharedVariable);

}

static void IncrementVariable()

{

lock (lockObject)

{

Console.WriteLine("Увеличение переменной...");

for (int i = 0; i < 1000000; i++)

{

sharedVariable++;

}

}

}

static void DecrementVariable()

{

lock (lockObject)

{

Console.WriteLine("Уменьшение переменной...");

for (int i = 0; i < 1000000; i++)

{

sharedVariable--;

}

}

}

}

**Задание 3.** **Какие существуют виды потоков и процессов?**

1. Потоки выполнения:

* Потоки ядра (kernel threads): это потоки, управляемые и создаваемые операционной системой. Они могут выполняться параллельно и имеют доступ к ресурсам и обработке ядра.
* Потоки пользовательского уровня (user-space threads): это потоки, управляемые прикладным программным обеспечением или библиотеками. Они могут быть реализованы как часть процесса на уровне пользователя, что позволяет многопоточным программам использовать параллельное исполнение.

1. Процессы:

* Однопроцессорные (single-threaded) процессы: это процессы, в которых содержится только один поток выполнения. Все инструкции выполняются последовательно.
* Многопроцессорные (multi-threaded) процессы: это процессы, содержащие несколько потоков выполнения, работающих параллельно. Каждый поток может выполнять свою собственную задачу, улучшая общую производительность.

**DEMO БИЛЕТ №23**

**Задание 1. Написать консольное приложение для проверки работоспособности клавиатуры (можно использовать разные методы и подходы)**

**Код:**

class Program  
 {  
 static void Main(string[] args)  
 {  
 Console.WriteLine("Нажмите любую клавишу, чтобы начать тест...");  
 Console.ReadKey();  
  
 string keys = "`1234567890-  
 =qwertyuiop[]\\asdfghjkl;'zxcvbnm,./~!@#$%^&\*()\_+QWERTYUIOP{}|ASDFGHJKL:\"ZXCVBNM<>?\r\n";  
 int currentKeyIndex = 0;  
 while (currentKeyIndex < keys.Length)  
 {  
 Console.WriteLine($"\nНажмите клавишу '{keys[currentKeyIndex]}':");  
 ConsoleKeyInfo keyInfo = Console.ReadKey();  
  
 if (keyInfo.KeyChar != keys[currentKeyIndex])  
 {  
 Console.WriteLine("\nОшибка: Неправильно нажата клавиша.");  
 Console.WriteLine("Тест не пройден.");  
 return;  
 }  
  
 currentKeyIndex++;  
 }  
  
 Console.WriteLine("\nУспех: тест клавиатуры пройден!");  
 }  
 }

**Задание 2. Напишите программу используя сокет для установки соединения и обмена данными между клиентом и сервером. Программа должна использовать библиотеку сокетов и демонстрирует основные операции с сокетами, такие как создание, связывание и прослушивание на определенном порту.**

**Код:**

**Серверная часть:**

using System;

using System.Net;

using System.Net.Sockets;

using System.Text;

class Server

{

static void Main()

{

// Создание сокета для сервера

Socket serverSocket = new Socket(AddressFamily.InterNetwork, SocketType.Stream, ProtocolType.Tcp);

// Привязка сокета к локальной конечной точке и запуск прослушивания

serverSocket.Bind(new IPEndPoint(IPAddress.Any, 8888));

serverSocket.Listen(5);

Console.WriteLine("Сервер запущен. Ожидание подключений...");

// Принятие клиентского подключения

Socket clientSocket = serverSocket.Accept();

// Обмен данными

byte[] messageBytes = Encoding.UTF8.GetBytes("Привет, клиент! Это сервер.");

clientSocket.Send(messageBytes);

// Закрытие сокетов

clientSocket.Shutdown(SocketShutdown.Both);

clientSocket.Close();

serverSocket.Close();

}

}

Клиентская часть:

using System;

using System.Net;

using System.Net.Sockets;

using System.Text;

class Client

{

static void Main()

{

// Создание сокета для клиента

Socket clientSocket = new Socket(AddressFamily.InterNetwork, SocketType.Stream, ProtocolType.Tcp);

// Подключение к серверу

clientSocket.Connect(IPAddress.Loopback, 8888);

// Обмен данными

byte[] buffer = new byte[1024];

int bytesRead = clientSocket.Receive(buffer);

string message = Encoding.UTF8.GetString(buffer, 0, bytesRead);

Console.WriteLine("Получено сообщение от сервера: " + message);

// Закрытие сокета

clientSocket.Shutdown(SocketShutdown.Both);

clientSocket.Close();

}

}

**Задание 3.** **Что такое сокет?**

Сокет - это интерфейс, который обеспечивает соединение между двумя процессами, работающими на разных компьютерах. Он позволяет передавать данные между этими процессами. Сокеты используются в клиент-серверных приложениях для обмена данными между клиентом и сервером.

**DEMO БИЛЕТ №24**

**Задание 1. Создать консольное приложение, которое считывает, что хранится в буфере и выводит на экран после запуска.**

**Код:**

using System;

using System.Windows.Forms;

class Program

{

static void Main()

{

// Проверяем, есть ли данные в буфере обмена

if (Clipboard.ContainsText())

{

// Считываем данные из буфера обмена

string clipboardText = Clipboard.GetText();

// Выводим данные на экран

Console.WriteLine("Данные из буфера обмена:");

Console.WriteLine(clipboardText);

}

else

{

Console.WriteLine("Буфер обмена пуст.");

}

}

}

**Задание 2. Имеется 4 процесса, они могут быть разделены на 2 группы. Чтобы все 4 выполнялись, потокам нужно 3 ресурса: N1, N2 и N3. У двух потоков только один ресурс. У одного из потоков есть все 3 ресурса. В один момент времени он отдает максимум 2 ресурса. Напишите программу используя процессы и потоки организовывая взаимодействие без дедлоков.**

**Код:**

{

while (true)

{

setlocale(LC\_CTYPE, "rus");

std::unique\_lock<std::mutex> lock(mtx);

cv.wait(lock, [] { return !tobaccoReady && !paperReady && matchesReady; });

std::cout << "Курильщик со спичками начинает курить." << std::endl;

tobaccoReady = false;

paperReady = false;

matchesReady = false;

lock.unlock();

std::this\_thread::sleep\_for(std::chrono::milliseconds(2000));

}

}

int main()

{

setlocale(LC\_CTYPE, "rus");

std::thread tobaccoProducer(makeTobacco);

std::thread paperProducer(makePaper);

std::thread matchesProducer(makeMatches);

std::thread tobaccoSmokerThread(tobaccoSmoker);

std::thread paperSmokerThread(paperSmoker);

std::thread matchesSmokerThread(matchesSmoker);

tobaccoProducer.join();

paperProducer.join();

matchesProducer.join();

tobaccoSmokerThread.join();

paperSmokerThread.join();

matchesSmokerThread.join();

return 0;

}

**Задание 3.** **Как работают процессы и потоки с графическим интерфейсом Windows 10?**

Процессы и потоки работают на основе асинхронного ввода-вывода. Когда пользователь нажимает на кнопку, система обрабатывает это как событие и вызывает соответствующий метод или функцию. Эта функция может создать новый поток или использовать существующий для выполнения операции. Windows использует многопоточность для обработки событий, каждый поток имеет свой стек. Процессы имеют свои адресные пространства и системные ресурсы, такие как память и процессор. Графические интерфейсы используют DirectX и OpenGL для работы с графикой.

**DEMO БИЛЕТ №25**

**Задание 1. Напишите программу, которая решает проблему состояния гонки в потоках. Можно либо использованием атомарных операций вместо нескольких инструкций чтение-изменение-запись, либо ограничивая доступ потоков к переменной так, чтобы в один момент времени только один поток мог изменять переменную.**

**Код:**

using System;

using System.Threading;

class Program

{

static int counter = 0;

static void IncrementCounter()

{

for (int i = 0; i < 10000; i++)

{

Interlocked.Increment(ref counter); // Атомарное увеличение переменной counter

}

}

static void Main()

{

Thread thread1 = new Thread(IncrementCounter);

Thread thread2 = new Thread(IncrementCounter);

thread1.Start();

thread2.Start();

thread1.Join();

thread2.Join();

Console.WriteLine("Результат: " + counter); // Ожидаемый результат: 20000

}

}

**Задание 2. Напишите клиент-серверное передающие GIF, текст, изображение через IP и порт. В клиенте и сервере должно быть сообщение об успешной отправке, а также о получении.**

**Код:**

**Серверная часть:**

using System;

using System.Net;

using System.Net.Sockets;

using System.Text;

class Server

{

static void Main()

{

TcpListener server = new TcpListener(IPAddress.Any, 8888);

server.Start();

Console.WriteLine("Сервер запущен. Ожидание подключений...");

TcpClient client = server.AcceptTcpClient();

NetworkStream stream = client.GetStream();

// Передача сообщения

byte[] messageBytes = Encoding.UTF8.GetBytes("Привет, клиент! Это сервер.");

stream.Write(messageBytes, 0, messageBytes.Length);

Console.WriteLine("Сообщение успешно отправлено.");

stream.Close();

client.Close();

server.Stop();

}

}

**Клиентская часть:**

using System;

using System.Net;

using System.Net.Sockets;

using System.Text;

class Client

{

static void Main()

{

TcpClient client = new TcpClient();

client.Connect(IPAddress.Loopback, 8888);

NetworkStream stream = client.GetStream();

// Получение сообщения

byte[] buffer = new byte[1024];

int bytesRead = stream.Read(buffer, 0, buffer.Length);

string message = Encoding.UTF8.GetString(buffer, 0, bytesRead);

Console.WriteLine("Получено сообщение от сервера: " + message);

stream.Close();

client.Close();

}

}

**Задание 3. Какие существуют виды клиент-серверных приложений?**

1. Веб-приложения: клиентское приложение запускается в веб-браузере пользователя и взаимодействует с сервером посредством HTTP или HTTPS протоколов. Примеры: онлайн-магазины, социальные сети.
2. Приложения с многоуровневой архитектурой: в таких приложениях клиентская часть может взаимодействовать с промежуточным слоем (middleware), который затем обращается к серверу базы данных. Это позволяет разделить логику приложения на более удобные уровни.
3. Игровые клиент-серверные приложения: в онлайн-играх клиентская часть обычно отвечает за отображение игрового мира, а сервер управляет логикой игры и синхронизирует действия всех игроков.
4. Приложения реального времени: в таких приложениях клиент и сервер постоянно обмениваются данными в реальном времени, например, мессенджеры или видеоконференции.

**DEMO БИЛЕТ №26**

**Задание 1. Напишите клиент-серверное передающие GIF через IP и порт. В клиенте и сервере должно быть сообщение об успешной отправке, а также о получении.**

**Код:**

**Серверная часть:**

using System;

using System.Net;

using System.Net.Sockets;

using System.Text;

class Server

{

static void Main()

{

TcpListener server = null;

try

{

// Устанавливаем IP адрес и порт

IPAddress ipAddress = IPAddress.Parse("127.0.0.1");

int port = 8888;

// Создаем TcpListener для прослушивания входящих соединений

server = new TcpListener(ipAddress, port);

server.Start();

Console.WriteLine("Сервер запущен. Ожидание подключений...");

// Принимаем клиентское подключение

TcpClient client = server.AcceptTcpClient();

Console.WriteLine("Клиент подключен.");

NetworkStream stream = client.GetStream();

// Передаем сообщение об успешном подключении

byte[] message = Encoding.UTF8.GetBytes("Соединение установлено. Ожидание GIF...");

stream.Write(message, 0, message.Length);

// Принимаем GIF

// Добавьте здесь код для приема GIF

// Отправляем сообщение об успешном получении GIF

message = Encoding.UTF8.GetBytes("GIF успешно получен.");

stream.Write(message, 0, message.Length);

stream.Close();

client.Close();

}

catch (Exception ex)

{

Console.WriteLine("Ошибка: " + ex.Message);

}

finally

{

server.Stop();

}

}

}

**Клиентская часть:**

using System;

using System.Net;

using System.Net.Sockets;

using System.Text;

class Client

{

static void Main()

{

try

{

// Устанавливаем IP адрес и порт сервера

IPAddress ipAddress = IPAddress.Parse("127.0.0.1");

int port = 8888;

TcpClient client = new TcpClient();

client.Connect(ipAddress, port);

NetworkStream stream = client.GetStream();

byte[] data = new byte[256];

StringBuilder response = new StringBuilder();

int bytes = stream.Read(data, 0, data.Length);

response.Append(Encoding.UTF8.GetString(data, 0, bytes));

Console.WriteLine(response.ToString());

// Отправляем GIF

// Добавьте здесь код для отправки GIF

byte[] message = Encoding.UTF8.GetBytes("GIF отправлен успешно.");

stream.Write(message, 0, message.Length);

stream.Close();

client.Close();

}

catch (Exception ex)

{

Console.WriteLine("Ошибка: " + ex.Message);

}

}

}

**Задание 2. Напишите программу, которая будет выполнять потоки поочередно с использованием mutex, а также используя остановку потока с помощью временных рамок.**

**Код:**

using System;

using System.Threading;

class Program

{

static Mutex mutex = new Mutex();

static int currentThread = 1;

static void ThreadFunction(object threadNumber)

{

int threadNum = (int)threadNumber;

while (true)

{

mutex.WaitOne();

if (threadNum == currentThread)

{

Console.WriteLine("Поток " + threadNum + " выполняется.");

Thread.Sleep(2000); // Симуляция выполнения работы потока

currentThread = currentThread % 2 + 1; // Переключаемся на другой поток

}

mutex.ReleaseMutex();

Thread.Sleep(1000); // Пауза перед следующей попыткой выполнения

}

}

static void Main()

{

Thread thread1 = new Thread(ThreadFunction);

Thread thread2 = new Thread(ThreadFunction);

thread1.Start(1);

thread2.Start(2);

thread1.Join();

thread2.Join();

}

}

**Задание 3. Как работает передача данных между процессами?**

Процессы могут взаимодействовать через общую память, каналы, мьютексы, события, семафоры и другие механизмы. В .NET Framework есть несколько классов для работы с этими механизмами, такие как System.Threading.Thread, System.IO.Pipes, System.Diagnostics.Process и другие.Для передачи данных между процессами можно использовать сокеты, разделяемую память, именованные каналы (named pipes), удаленный вызов процедур (RPC), очереди сообщений и другие механизмы связи.

**DEMO БИЛЕТ №27**

**Задание 1. Напишите клиент-серверное приложение передачи текста.**

**Код:**

**Серверное приложение:**

using System;

using System.Net;

using System.Net.Sockets;

using System.Text;

class Server

{

static void Main()

{

TcpListener server = null;

try

{

// Указываем IP-адрес и порт для прослушивания

IPAddress localAddr = IPAddress.Parse("127.0.0.1");

int port = 12345;

// Создаем TcpListener для прослушивания входящих подключений

server = new TcpListener(localAddr, port);

// Начинаем прослушивание

server.Start();

Console.WriteLine("Сервер запущен. Ожидание подключений...");

// Принимаем клиентское подключение

TcpClient client = server.AcceptTcpClient();

// Получаем сетевой поток для чтения и записи

NetworkStream stream = client.GetStream();

byte[] data = new byte[256];

StringBuilder builder = new StringBuilder();

int bytes = 0;

do

{

bytes = stream.Read(data, 0, data.Length);

builder.Append(Encoding.UTF8.GetString(data, 0, bytes));

} while (stream.DataAvailable);

string message = builder.ToString();

Console.WriteLine("Получено сообщение от клиента: " + message);

stream.Close();

client.Close();

}

finally

{

server.Stop();

}

Console.WriteLine("Сервер завершил работу.");

}

}

**Клиентское приложение:**

using System;

using System.Net.Sockets;

using System.Text;

class Client

{

static void Main()

{

TcpClient client = new TcpClient("127.0.0.1", 12345);

NetworkStream stream = client.GetStream();

string message = "Привет, сервер!";

byte[] data = Encoding.UTF8.GetBytes(message);

stream.Write(data, 0, data.Length);

Console.WriteLine("Сообщение отправлено серверу: " + message);

stream.Close();

client.Close();

Console.WriteLine("Клиент завершил работу.");

}

}

**Задание 2. Напишите DLL, содержащую функцию вычисления минимума двух чисел, содержащую форму со сведениями об авторе программы, содержащую функцию перевода целого числа от 0 до 1000 в запись словами на русском языке.**

**1. Создание проекта DLL в Visual Studio:**

* **Создайте новый проект типа "Class Library" в Visual Studio.**
* **Добавьте новый класс MathHelper.cs в проект.**

2. Реализация функций в DLL:

**Код:**

MathHelper.cs:

using System;

namespace MyLibrary

{

public class MathHelper

{

public static int GetMin(int num1, int num2)

{

return Math.Min(num1, num2);

}

public static string NumberToWords(int number)

{

if (number < 0 || number > 1000)

{

return "Число не входит в диапазон от 0 до 1000";

}

// Реализация функции перевода числа в слова на русском языке

// (можно использовать готовый алгоритм или библиотеку)

return "Реализация перевода числа в слова";

}

}

}

3. Добавление формы с информацией об авторе:

* **Добавьте новую форму AboutForm.cs в проект.**
* **На форме разместите информацию об авторе программы, например, имя, контактные данные и т.д.**

4. Сборка проекта в DLL:

* **Соберите проект, чтобы создать DLL файл.**
* **DLL файл будет находиться в папке bin\Debug или bin\Release в зависимости от режима сборки.**

**После создания DLL вы сможете использовать ее в других проектах, подключив её как ссылку.**

**Пример использования функций из DLL в другом проекте:**

**Код:**

using MyLibrary;

class Program

{

static void Main()

{

int min = MathHelper.GetMin(5, 3);

Console.WriteLine("Минимум: " + min);

string numberInWords = MathHelper.NumberToWords(123);

Console.WriteLine("Число словами: " + numberInWords);

}

}

**Задание 3. В чем отличие динамических DLL от статических?**

Динамические DLL обычно меньше по размеру, так как они загружаются и выгружаются по мере необходимости, в то время как статические библиотеки увеличивают размер исполняемого файла, поскольку весь их код включается в него.Динамические DLL могут быть обновлены без необходимости перекомпиляции и пересоздания статических библиотек, что может быть полезно для исправления ошибок или добавления новых функций

**DEMO БИЛЕТ №28**

**Задание 1. Напишите клиент-серверное передающие GIF через IP и порт. В клиенте и сервере должно быть сообщение об успешной отправке, а также о получении.**

**Код:**

**Сервер:**

using System;

using System.Net;

using System.Net.Sockets;

using System.IO;

class Server

{

static void Main()

{

TcpListener listener = new TcpListener(IPAddress.Any, 1234);

listener.Start();

Console.WriteLine("Сервер запущен. Ожидание подключения...");

TcpClient client = listener.AcceptTcpClient();

NetworkStream stream = client.GetStream();

Console.WriteLine("Клиент подключен. Отправка GIF...");

using (FileStream fileStream = File.OpenRead("image.gif"))

{

fileStream.CopyTo(stream);

}

Console.WriteLine("GIF успешно отправлен.");

stream.Close();

client.Close();

listener.Stop();

Console.WriteLine("Сервер завершил работу.");

}

}

**Клиент:**

using System;

using System.Net;

using System.Net.Sockets;

using System.IO;

class Client

{

static void Main()

{

TcpClient client = new TcpClient();

client.Connect(IPAddress.Parse("127.0.0.1"), 1234);

Console.WriteLine("Подключение к серверу. Получение GIF...");

NetworkStream stream = client.GetStream();

using (FileStream fileStream = File.Create("received\_image.gif"))

{

stream.CopyTo(fileStream);

}

Console.WriteLine("GIF успешно получен.");

stream.Close();

client.Close();

Console.WriteLine("Работа клиента завершена.");

}

}

**Задание 2. Напишите программу, которая будет выполнять потоки поочередно с использованием mutex, а также используя остановку потока с помощью временных рамок.**

**Код:**

using System;

using System.Threading;

class Program

{

static Mutex mutex = new Mutex();

static int counter = 0;

static void Thread1()

{

for (int i = 0; i < 5; i++)

{

mutex.WaitOne();

Console.WriteLine("Поток 1: " + counter++);

mutex.ReleaseMutex();

Thread.Sleep(1000); // Остановка потока на 1 секунду

}

}

static void Thread2()

{

for (int i = 0; i < 5; i++)

{

mutex.WaitOne();

Console.WriteLine("Поток 2: " + counter++);

mutex.ReleaseMutex();

Thread.Sleep(2000); // Остановка потока на 2 секунды

}

}

static void Main()

{

Thread thread1 = new Thread(Thread1);

Thread thread2 = new Thread(Thread2);

thread1.Start();

thread2.Start();

thread1.Join();

thread2.Join();

Console.WriteLine("Программа завершена.");

}

}

**Задание 3.** **Чем обладает процесс внутри ОС?**

Динамические DLL обычно меньше по размеру, так как они загружаются и выгружаются по мере необходимости, в то время как статические библиотеки увеличивают размер исполняемого файла, поскольку весь их код включается в него.

Процесс внутри операционной системы - это как отдельная задача или работа на компьютере. Он имеет свои ресурсы, такие как память и процессор, и может выполняться независимо от других процессов. Операционная система отслеживает и управляет этими процессами, чтобы все работало правильно. Каждый процесс имеет свои правила и состояния, в которых он может быть. Это помогает компьютеру эффективно выполнять задачи и защищать их друг от друга.

**DEMO БИЛЕТ №29**

**Задание 1.** Напишите клиент-серверное передающие GIF через IP и порт. В клиенте и сервере должно быть сообщение об успешной отправке, а также о получении.

**Код:**

**Серверная часть:**

using System;

using System.Net;

using System.Net.Sockets;

using System.IO;

class Server

{

static void Main()

{

TcpListener listener = new TcpListener(IPAddress.Any, 12345);

listener.Start();

Console.WriteLine("Сервер запущен. Ожидание подключения клиента...");

TcpClient client = listener.AcceptTcpClient();

Console.WriteLine("Клиент подключен!");

NetworkStream stream = client.GetStream();

// Отправка GIF-изображения клиенту

string imagePath = "image.gif";

byte[] imageData = File.ReadAllBytes(imagePath);

stream.Write(imageData, 0, imageData.Length);

Console.WriteLine("Изображение успешно отправлено клиенту.");

stream.Close();

client.Close();

listener.Stop();

Console.WriteLine("Работа сервера завершена.");

}

}

**Клиентская часть:**

using System;

using System.Net;

using System.Net.Sockets;

using System.IO;

class Client

{

static void Main()

{

TcpClient client = new TcpClient();

client.Connect(IPAddress.Parse("127.0.0.1"), 12345);

NetworkStream stream = client.GetStream();

byte[] imageData = new byte[1024];

int bytesRead;

using (MemoryStream memoryStream = new MemoryStream())

{

while ((bytesRead = stream.Read(imageData, 0, imageData.Length)) > 0)

{

memoryStream.Write(imageData, 0, bytesRead);

}

byte[] receivedData = memoryStream.ToArray();

File.WriteAllBytes("received\_image.gif", receivedData);

Console.WriteLine("Изображение успешно получено от сервера и сохранено.");

stream.Close();

client.Close();

Console.WriteLine("Работа клиента завершена.");

}

}

}

**Задание 2. Напишите программу, которая будет выполнять потоки поочередно с использованием mutex, а также используя остановку потока с помощью временных рамок.**

**Код:**

using System;

using System.Threading;

class Program

{

static Mutex mutex = new Mutex();

static int counter = 0;

static void Thread1()

{

for (int i = 0; i < 5; i++)

{

mutex.WaitOne();

Console.WriteLine("Поток 1: " + counter++);

mutex.ReleaseMutex();

Thread.Sleep(1000); // Остановка потока на 1 секунду

}

}

static void Thread2()

{

for (int i = 0; i < 5; i++)

{

mutex.WaitOne();

Console.WriteLine("Поток 2: " + counter++);

mutex.ReleaseMutex();

Thread.Sleep(2000); // Остановка потока на 2 секунды

}

}

static void Main()

{

Thread thread1 = new Thread(Thread1);

Thread thread2 = new Thread(Thread2);

thread1.Start();

thread2.Start();

thread1.Join();

thread2.Join();

Console.WriteLine("Программа завершена.");

}

}

**Задание 3.** **Какие существуют виды клиент-серверных приложений?**

1. Веб-приложения: клиентское приложение запускается в веб-браузере пользователя и взаимодействует с сервером посредством HTTP или HTTPS протоколов. Примеры: онлайн-магазины, социальные сети.
2. Приложения с многоуровневой архитектурой: в таких приложениях клиентская часть может взаимодействовать с промежуточным слоем (middleware), который затем обращается к серверу базы данных. Это позволяет разделить логику приложения на более удобные уровни.
3. Игровые клиент-серверные приложения: в онлайн-играх клиентская часть обычно отвечает за отображение игрового мира, а сервер управляет логикой игры и синхронизирует действия всех игроков.
4. Приложения реального времени: в таких приложениях клиент и сервер постоянно обмениваются данными в реальном времени, например, мессенджеры или видеоконференции.

**DEMO БИЛЕТ №30**

**Задание 1. Создать консольное приложение и DLL библиотеку. Совместить в одном проекте данные с помощью связи ссылкой.**

Шаг 1: Создание DLL библиотеки

1. Создайте новый проект DLL в Visual Studio:
   * Выберите "File" -> "New" -> "Project...".
   * Выберите "Class Library (.NET Framework)".
   * Укажите имя проекта (например, MyLibrary) и нажмите "Create".
2. Добавьте в проект класс с какой-либо функциональностью. Например:

namespace MyLibrary

{

public class MyMath

{

public static int Add(int a, int b)

{

return a + b;

}

}

}

1. Соберите проект DLL: выберите "Build" -> "Build Solution".

Шаг 2: Создание консольного приложения

1. Создайте новый проект консольного приложения в том же решении:
   * Выберите "File" -> "Add" -> "New Project...".
   * Выберите "Console App (.NET Framework)".
   * Укажите имя проекта (например, MyConsoleApp) и нажмите "Create".

Шаг 3: Связь консольного приложения с DLL

1. Добавьте ссылку на DLL библиотеку в консольном приложении:
   * Щелкните правой кнопкой мыши на References в проекте консольного приложения.
   * Выберите "Add Reference...".
   * Выберите "Projects" и выберите проект DLL (например, MyLibrary).
   * Нажмите "OK".
2. Используйте функции из DLL в консольном приложении:

**Код:**

using System;

namespace MyConsoleApp

{

class Program

{

static void Main()

{

int result = MyLibrary.MyMath.Add(5, 3);

Console.WriteLine("Результат сложения: " + result);

}

}

}

1. Запустите консольное приложение, чтобы убедиться, что оно успешно использует функцию из DLL.

**Задание 2. Напишите программу, которая будет использовать 2 потока для передачи данных.**

**Код:**

using System;

using System.Collections.Concurrent;

using System.Threading;

class Program

{

static BlockingCollection<int> dataQueue = new BlockingCollection<int>();

static void Producer()

{

for (int i = 1; i <= 10; i++)

{

dataQueue.Add(i);

Console.WriteLine($"Производитель: передано число {i}");

Thread.Sleep(1000);

}

dataQueue.CompleteAdding();

}

static void Consumer()

{

foreach (int data in dataQueue.GetConsumingEnumerable())

{

Console.WriteLine($"Потребитель: получено число {data}");

Thread.Sleep(2000);

}

}

static void Main()

{

Thread producerThread = new Thread(Producer);

Thread consumerThread = new Thread(Consumer);

producerThread.Start();

consumerThread.Start();

producerThread.Join();

consumerThread.Join();

Console.WriteLine("Программа завершена.");

}

}

**Задание 3.** **Что такое процесс?**

Процесс - это экземпляр программы, который выполняется на компьютере. Каждый процесс имеет свой собственный адресное пространство, стеки, регистры и другие ресурсы, которые необходимы для его выполнения. Процессы могут выполняться параллельно, то есть одновременно, на одном или нескольких ядрах процессора.

**DEMO БИЛЕТ №31**

**Задание 1. Напишите DLL, содержащую функциюмш вывода картинки по цифре, введенной с клавиатуры в консольном приложение не менее 10 изображений.**

**Код:**

**DLL с функцией для вывода изображения:**

using System;

using System.Drawing;

namespace ImageLibrary

{

public class ImageHandler

{

public static void DisplayImageByNumber(int number)

{

string imagePath = $"image\_{number}.jpg"; // Предполагается, что изображения имеют имена вида "image\_1.jpg", "image\_2.jpg" и т.д.

try

{

Bitmap image = new Bitmap(imagePath);

Console.WriteLine($"Изображение для числа {number}:");

for (int y = 0; y < image.Height; y++)

{

for (int x = 0; x < image.Width; x++)

{

Color pixelColor = image.GetPixel(x, y);

Console.Write(pixelColor.R > 128 ? " " : "#"); // Простой вывод пикселей через символы

}

Console.WriteLine();

}

}

catch (Exception ex)

{

Console.WriteLine($"Ошибка при загрузке изображения: {ex.Message}");

}

}

}

}

**Консольное приложение для использования DLL:**

using System;

using ImageLibrary;

class Program

{

static void Main()

{

for (int i = 1; i <= 10; i++)

{

Console.Write("Введите число (1-10) для отображения соответствующего изображения: ");

if (int.TryParse(Console.ReadLine(), out int number) && number >= 1 && number <= 10)

{

ImageHandler.DisplayImageByNumber(number);

}

else

{

Console.WriteLine("Некорректный ввод. Пожалуйста, введите число от 1 до 10.");

}

}

}

}

**Задание 2. Написать консольное приложение для изменения цвета текста в зависимости от выбора пользователя.**

**Код:**

using System;

class Program

{

static void Main()

{

Console.WriteLine("Выберите цвет текста:");

Console.WriteLine("1. Красный");

Console.WriteLine("2. Зеленый");

Console.WriteLine("3. Синий");

int choice = int.Parse(Console.ReadLine());

switch (choice)

{

case 1:

Console.ForegroundColor = ConsoleColor.Red;

break;

case 2:

Console.ForegroundColor = ConsoleColor.Green;

break;

case 3:

Console.ForegroundColor = ConsoleColor.Blue;

break;

default:

Console.WriteLine("Некорректный выбор. Установлен цвет по умолчанию.");

break;

}

Console.WriteLine("Текст с выбранным цветом.");

Console.ResetColor(); // Сброс цвета обратно на стандартный

Console.WriteLine("Программа завершена. Нажмите любую клавишу для выхода.");

Console.ReadKey();

}

}

**Задание 3.** **Что такое поток?**

**Поток** - это последовательность команд, которые выполняются процессором в определенной последовательности. Поток может выполнять различные операции, такие как чтение и запись в память, выполнение операций с процессором, взаимодействие с другими потоками и т.д. Потоки используются для повышения производительности и параллелизма в приложениях.

**DEMO БИЛЕТ №32**

**Задание 1. Написать консольную программу, которая выводит список подключенного оборудования в окне консоли.**

**Код:**

using System;

using System.Management;

namespace DeviceList

{

class Program

{

static void Main(string[] args)

{

ManagementObjectSearcher searcher = new ManagementObjectSearcher("SELECT \* FROM Win32\_PnPEntity");

Console.WriteLine("Список подключенного оборудования:");

foreach (ManagementObject queryObj in searcher.Get())

{

Console.WriteLine("Устройство: " + queryObj["Caption"]);

}

}

}

}

**Задание 2. Напишите клиент-серверное приложение передающие сообщение через IP и порт.**

**Код:**

**Серверная часть:**

using System;

using System.Net;

using System.Net.Sockets;

using System.Text;

class Server

{

static void Main()

{

TcpListener server = new TcpListener(IPAddress.Parse("127.0.0.1"), 12345);

server.Start();

Console.WriteLine("Сервер ожидает подключения...");

TcpClient client = server.AcceptTcpClient();

NetworkStream stream = client.GetStream();

byte[] data = new byte[256];

int bytes = stream.Read(data, 0, data.Length);

string message = Encoding.UTF8.GetString(data, 0, bytes);

Console.WriteLine("Получено сообщение от клиента: " + message);

stream.Close();

client.Close();

server.Stop();

Console.WriteLine("Сервер завершил работу.");

}

}

**Клиентская часть:**

using System;

using System.Net;

using System.Net.Sockets;

using System.Text;

class Client

{

static void Main()

{

TcpClient client = new TcpClient();

client.Connect(IPAddress.Parse("127.0.0.1"), 12345);

NetworkStream stream = client.GetStream();

string message = "Привет, сервер!";

byte[] data = Encoding.UTF8.GetBytes(message);

stream.Write(data, 0, data.Length);

Console.WriteLine("Сообщение отправлено серверу: " + message);

stream.Close();

client.Close();

Console.WriteLine("Клиент завершил работу.");

}

}

**Задание 3.** **Какими методами можно проверить работоспособность оборудования ПК?**

1. Визуальный осмотр: Проверьте, нет ли видимых повреждений на корпусе компьютера, кабелях, блоке питания и других компонентах.
2. Проверка на перегрев: Используйте специализированные программы или утилиты для проверки температуры процессора, видеокарты, жесткого диска и других компонентов. Если температура превышает допустимые значения, это может указывать на проблемы с охлаждением.
3. Тестирование оперативной памяти: Используйте программу memtest86+ для проверки оперативной памяти на наличие ошибок.
4. Тестирование видеокарты: Используйте программы для стресс-тестирования видеокарты, такие как FurMark или OCCT, чтобы проверить ее на наличие артефактов, вылетов и перегрева.
5. Проверка жесткого диска: Используйте утилиты, такие как CrystalDiskInfo или HD Tune, для проверки состояния жесткого диска, скорости чтения/записи, температуры и других параметров.
6. Проверка сетевого адаптера: Используйте команды ping и tracert в командной строке для проверки подключения к сети и времени отклика.

**DEMO БИЛЕТ №33**

**Задание 1. При помощи командной строки проанализируйте работу подсистем управления ресурсов.**

**Код:**

using System;

using System.Diagnostics;

class Program

{

static void Main()

{

Process process = new Process();

process.StartInfo.FileName = "tasklist";

process.StartInfo.UseShellExecute = false;

process.StartInfo.RedirectStandardOutput = true;

process.Start();

string output = process.StandardOutput.ReadToEnd();

Console.WriteLine(output);

process.WaitForExit();

}

}

**Задание 2. Дан массив строк: "red", "green", "black", "white", "blue". Запишите в файл элементы массива построчно (каждый элемент в новой строке). При помощи консольного приложения**

**Код:**

using System;  
 using [System.IO](https://vk.com/away.php?to=http%3A%2F%2FSystem.IO&cc_key=);  
  
 class Program  
 {  
 static void Main()  
 {  
 string[] colors = { "red", "green", "black", "white", "blue" };  
  
 string path = "colors.txt";  
  
 using (StreamWriter writer = new StreamWriter(path))  
 {  
 foreach (string color in colors)  
 {  
 writer.WriteLine(color);  
 }  
 }  
  
 Console.WriteLine("Элементы массива были успешно записаны в файл.");  
 Console.ReadLine();  
 }  
 }

**Задание 3.** **Для чего нужна подсистема управления ресурсов?**

Эта подсистема используется для управления различными типами ресурсов, такими как память, процессорное время, файлы, сетевые подключения и другие, с целью оптимизации их использования.

**DEMO БИЛЕТ №34**

**Задание 1. Напишите консольное приложение, которая в созданный файл numbers.txt записывает натуральные числа от 1 до 500 через запятую.**

**Код:**

using System;

using [System.IO](https://vk.com/away.php?to=http%3A%2F%2FSystem.IO&cc_key=);

class Program {

static void Main()

{

string filePath = "numbers.txt";

string numbers = string.Join(",",

Enumerable.Range(1, 500));

File.WriteAllText(filePath, numbers);

}

}

**Задание 2. Добавить программу в автозагрузку системного реестра с добавлением свойств.**

**Код:**

using Microsoft.Win32;

using System;

class Program

{

static void Main()

{

string appName = "MyApp";

string appPath = @"C:\Path\To\Your\Application.exe";

// Добавляем программу в автозагрузку

RegistryKey key = Registry.CurrentUser.OpenSubKey("SOFTWARE\\Microsoft\\Windows\\CurrentVersion\\Run", true);

key.SetValue(appName, appPath);

key.Close();

// Устанавливаем дополнительные свойства

RegistryKey appKey = Registry.CurrentUser.OpenSubKey("SOFTWARE\\MyApp", true);

if (appKey == null)

{

appKey = Registry.CurrentUser.CreateSubKey("SOFTWARE\\MyApp");

}

appKey.SetValue("Version", "1.0");

appKey.SetValue("Author", "Your Name");

appKey.Close();

Console.WriteLine("Программа успешно добавлена в автозагрузку и установлены дополнительные свойства.");

}

}

======================================================================================================

1. HKEY\_CURRENT\_USER\SOFTWARE\Microsoft\Windows\CurrentVersion\Run
2. Нажал ПКМ, создать и выбрал строковый параметр
3. Создал строковый параметр с названием lb7, и в значение добавил путь до VScode, также в значение добавил параметр /background

**Задание 3.** **Какой метод позволяет работать с файлами и каталогами?**

В C# для работы с файлами и каталогами можно использовать класс System.IO.Directory. Этот класс содержит различные методы для работы с каталогами, такие как создание, удаление, перечисление файлов и каталогов и т.д.

**1. Для работы с файлами File**

Для работы с файлами в C# используется класс System.IO.File. Этот класс предоставляет методы для чтения, записи, создания и удаления файлов, а также для выполнения других операций с ними.

**2. Для работы с каталогами: Directory**

Для работы с каталогами в C# используется класс System.IO.Directory. Этот класс позволяет создавать, удалять и перемещаться по каталогам, а также получать информацию о них.

**DEMO БИЛЕТ №35**

**Задание 1. Создать консольное приложение, которое считывает, что хранится в буфере и выводит на экран после запуска.**

**Код (дополнительно подключить win.forms):**

using System;

using System.Windows.Forms;

namespace ConsoleApp4

{

internal class Program

{

[STAThread]

static void Main(string[] args)

{

Console.WriteLine("Поулчение данных с буфера");

string prinatye = Clipboard.GetText();

Console.WriteLine(prinatye);

}

}

}

**Задание 2. Напишите консольную программу, которая выводит список подключенных устройств.**

**Код:**

using System;

using System.Management;

class Program

{

static void Main()

{

ManagementObjectSearcher searcher = new ManagementObjectSearcher("SELECT \* FROM Win32\_PnPEntity");

Console.WriteLine("Список подключенных устройств:");

foreach (ManagementObject device in searcher.Get())

{

Console.WriteLine("Устройство: {0}", device["Caption"]);

Console.WriteLine("Описание: {0}", device["Description"]);

Console.WriteLine("---------------------------------------");

}

}

}

**Задание 3. Что такое консоль и для чего она нужна?**

Консоль - это программное или аппаратное средство, которое позволяет пользователю взаимодействовать с компьютером или другой системой. Она обычно используется для ввода и вывода информации, выполнения команд и управления различными функциями системы.

using System;

using System.Drawing;

using System.Runtime.InteropServices;

using System.Windows.Forms;

namespace ImageDLL

{

public class ImageRenderer

{

[DllImport("User32.dll")]

public static extern IntPtr GetDC(IntPtr hwnd);

[DllImport("User32.dll")]

public static extern int ReleaseDC(IntPtr hwnd, IntPtr hdc);

public void RenderImage(int number)

{

using (Graphics g = Graphics.FromHdc(GetDC([IntPtr.Zero](https://vk.com/away.php?utf=1&to=http%3A%2F%2FIntPtr.Zero))))

{

switch (number)

{

case 0:

g.DrawImage(Image.FromFile("0.png"), 0, 0);

break;

case 1:

g.DrawImage(Image.FromFile("1.png"), 0, 0);

break;

// Добавьте остальные кейсы для остальных картинок

default:

break;

}

}

ReleaseDC([IntPtr.Zero](https://vk.com/away.php?utf=1&to=http%3A%2F%2FIntPtr.Zero), GetDC([IntPtr.Zero](https://vk.com/away.php?utf=1&to=http%3A%2F%2FIntPtr.Zero)));

}

}

}

using System;

using ImageDLL;

class Program

{

static void Main()

{

ImageRenderer imageRenderer = new ImageRenderer();

Console.WriteLine("Введите число от 0 до 9:");

int number;

while (!int.TryParse(Console.ReadLine(), out number) || number < 0 || number > 9)

{

Console.WriteLine("Введите корректное число от 0 до 9:");

}

imageRenderer.RenderImage(number);

}

}

1. Создайте новый проект "Class Library" в Visual Studio и назовите его MathLibrary.
2. Внутри проекта MathLibrary создайте новый класс MathOperations с методами для выполнения арифметических операций, например, Add, Subtract, Multiply, Divide.

cpp

Скопировать код

using System;

namespace MathLibrary

{

public class MathOperations

{

public int Add(int a, int b)

{

return a + b;

}

public int Subtract(int a, int b)

{

return a - b;

}

public int Multiply(int a, int b)

{

return a \* b;

}

public double Divide(int a, int b)

{

if (b == 0)

{

throw new DivideByZeroException("Cannot divide by zero.");

}

return (double)a / b;

}

}

}

1. Скомпилируйте проект MathLibrary, чтобы получить DLL-библиотеку MathLibrary.dll.
2. Создайте новый проект "Console Application" в Visual Studio и назовите его MathConsoleApp.
3. Добавьте ссылку на DLL-библиотеку MathLibrary.dll в проект MathConsoleApp.
4. В методе Main программы MathConsoleApp вызывайте методы из класса MathOperations для выполнения арифметических операций и вывода результатов на консоль.

csharp

Скопировать код

using MathLibrary;

using System;

namespace MathConsoleApp

{

class Program

{

static void Main(string[] args)

{

MathOperations mathOperations = new MathOperations();

int a = 10;

int b = 5;

Console.WriteLine($"Addition: {a} + {b} = {mathOperations.Add(a, b)}");

Console.WriteLine($"Subtraction: {a} - {b} = {mathOperations.Subtract(a, b)}");

Console.WriteLine($"Multiplication: {a} \* {b} = {mathOperations.Multiply(a, b)}");

try

{

Console.WriteLine($"Division: {a} / {b} = {mathOperations.Divide(a, b)}");

}

catch (DivideByZeroException ex)

{

Console.WriteLine(ex.Message);

}

}

}

}

1. Скомпилируйте и запустите проект MathConsoleApp. Вы увидите результаты арифметических операций на консоли.

using System;

using System.Drawing;

using System.Runtime.InteropServices;

using System.Windows.Forms;

namespace ImageDLL

{

public class ImageRenderer

{

[DllImport("User32.dll")]

public static extern IntPtr GetDC(IntPtr hwnd);

csharp

Copy code

[DllImport("User32.dll")]

public static extern int ReleaseDC(IntPtr hwnd, IntPtr hdc);

public void RenderImage(int number)

{

try

{

using (Graphics g = Graphics.FromHdc(GetDC(IntPtr.Zero)))

{

switch (number)

{

case 0:

g.DrawImage(Image.FromFile("0.png"), 0, 0);

break;

case 1:

g.DrawImage(Image.FromFile("1.png"), 0, 0);

break;

// Добавьте остальные кейсы для остальных картинок

default:

break;

}

}

}

catch (Exception ex)

{

Console.WriteLine("An error occurred while rendering the image: " + ex.Message);

}

finally

{

ReleaseDC(IntPtr.Zero, GetDC(IntPtr.Zero));

}

}

}

}

ЛЮБЛЮ ТЕБЯ БРАТ