МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«СЕВАСТОПОЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»
МОРСКОЙ КОЛЛЕДЖ

МЕТОДИЧЕСКИЕ РЕКОМЕНДАЦИИ

к практическим занятиям студентов
09.02.07 Информационные системы и программирование
Общепрофессиональный цикл

Разработали: преподаватели:
Батыргареева С.Ф.,
Корепанова Н.Л.,
Лебедева М.А.,
Сушкова Ю.А.

2022г

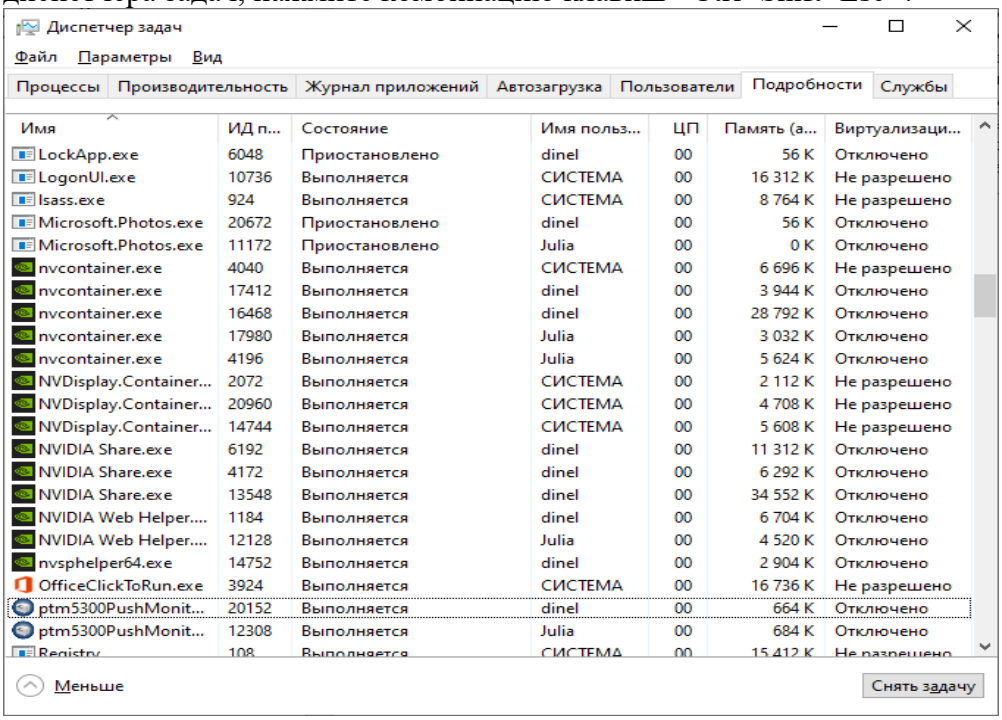
Практическое занятие №1
МОНИТОРИНГ ПРОЦЕССОВ, ПОТОКОВ

Цель работы: знакомство с методами классов Process и Thread, обеспечивающими получение информации о выполняющихся в данный момент процессах MS Windows и используемых ими потоках.

Теоретические сведения:

C#: Процессы Windows

Понятие "процесса" существовало в операционных системах Windows задолго до появления платформы .NET. Попросту говоря, под процессом понимается выполняющаяся программа. Однако формально процесс — это концепция уровня операционной системы, которая используется для описания набора ресурсов (таких как внешние библиотеки кода и главный поток) и необходимой памяти, используемой выполняющимся приложением. Для каждого загружаемого в память файла *.exe в операционной системе создается отдельный изолированный процесс, который используется на протяжении всего времени его существования. Благодаря такой изоляции приложений, исполняющая среда получается гораздо более надежной и стабильной, поскольку выход из строя одного процесса никак не сказывается на работе других процессов.
Более того, доступ напрямую к данным в одном процессе из другого процесса невозможен, если только не применяется API-интерфейс распределенных вычислений, такой как Windows Communication Foundation. Из-за всех этих моментов процесс может считаться фиксированной и безопасной границей выполняющегося приложения.
Каждый процесс Windows получает уникальный идентификатор процесса (Process ID — PID) и может независимо загружаться и выгружаться операционной системой (в том числе программно). Как уже наверняка известно, в окне WindowsTask Manager (Диспетчер задач) имеется вкладка Processes (Процессы), на которой можно просматривать различные статические данные о выполняющихся на данной машине процессах, в том числе их PID-идентификаторы и имена образов. Чтобы открыть окно диспетчера задач, нажмите комбинацию клавиш <Ctrl+Shift+Esc>:



Имя	ИД п...	Состояние	Имя польз...	ЦП	Память (а...	Виртуализаци...
LockApp.exe	6048	Приостановлено	dinel	00	56 K	Отключено
LogonUI.exe	10736	Выполняется	СИСТЕМА	00	16 312 K	Не разрешено
lsass.exe	924	Выполняется	СИСТЕМА	00	8 764 K	Не разрешено
Microsoft.Photos.exe	20672	Приостановлено	dinel	00	56 K	Отключено
Microsoft.Photos.exe	11172	Приостановлено	Julia	00	0 K	Отключено
nvcontainer.exe	4040	Выполняется	СИСТЕМА	00	6 696 K	Не разрешено
nvcontainer.exe	17412	Выполняется	dinel	00	3 944 K	Отключено
nvcontainer.exe	16468	Выполняется	dinel	00	28 792 K	Отключено
nvcontainer.exe	17980	Выполняется	Julia	00	3 032 K	Отключено
nvcontainer.exe	4196	Выполняется	Julia	00	5 624 K	Отключено
NVDisplay.Container...	2072	Выполняется	СИСТЕМА	00	2 112 K	Не разрешено
NVDisplay.Container...	20960	Выполняется	СИСТЕМА	00	4 708 K	Не разрешено
NVDisplay.Container...	14744	Выполняется	СИСТЕМА	00	5 608 K	Не разрешено
NVIDIA Share.exe	6192	Выполняется	dinel	00	11 312 K	Отключено
NVIDIA Share.exe	4172	Выполняется	dinel	00	6 292 K	Отключено
NVIDIA Share.exe	13548	Выполняется	dinel	00	34 552 K	Отключено
NVIDIA Web Helper....	1184	Выполняется	dinel	00	6 704 K	Отключено
NVIDIA Web Helper....	12128	Выполняется	Julia	00	4 520 K	Отключено
nvsphelper64.exe	14752	Выполняется	dinel	00	2 904 K	Отключено
OfficeClickToRun.exe	3924	Выполняется	СИСТЕМА	00	16 736 K	Не разрешено
ptm5300PushMonit...	20152	Выполняется	dinel	00	664 K	Отключено
ptm5300PushMonit...	12308	Выполняется	Julia	00	684 K	Отключено
Registry	108	Выполняется	СИСТЕМА	00	15 412 K	Не разрешено

Потоки

В каждом процессе Windows содержится первоначальный "поток", который является входной точкой для приложения. Потоком называется используемый внутри процесса путь выполнения. Формально поток, который создается первым во входной точке процесса, называется главным потоком (primary thread). В любой исполняемой программе .NET (консольном приложении, приложении Windows Forms, приложении WPF и т.д.) входная точка обозначается как метод Main(). При вызове этого метода главный поток создается автоматически.

Процессы, в которых содержится единственный главный поток выполнения, изначально являются безопасными к потокам (thread safe), поскольку в каждый отдельный момент времени доступ к данным приложения в них может получать только один поток. Однако подобные однопоточные процессы (особенно с графическим пользовательским интерфейсом) часто замедленно реагируют на действия пользователя, когда их единственный поток выполняет какую-то сложную операцию (вроде вывода на печать длинного текстового файла, сложных математических вычислений или подключения к удаленному серверу).

Из-за такого потенциального недостатка однопоточных приложений, API-интерфейс Windows (а также платформа .NET) предоставляет возможность для главного потока порождать дополнительные вторичные потоки (также называемые рабочими потоками). Это делается с применением набора функций из API-интерфейса Windows, таких как CreateThread() . Каждый поток (первичный или вторичный) в процессе становится уникальным путем выполнения и может параллельно получать доступ ко всем разделяемым элементам данных внутри соответствующего процесса.

Как нетрудно догадаться, разработчики обычно создают дополнительные потоки для улучшения общей степени восприимчивости программы к действиям пользователя. Многопоточные процессы обеспечивают иллюзию того, что выполнение многочисленных действий происходит примерно в одно и то же время. Например, дополнительный рабочий поток может порождаться в приложении для выполнения какой-нибудь трудоемкой задачи (подобной выводу на печать большого текстового файла). После начала выполнения задачи вторичным потоком основной поток все равно не утрачивает способности реагировать на действия пользователя, что дает всему процессу возможность сопровождаться куда более высокой производительностью.

Взаимодействие с процессами в .NET

Хотя в самих процессах и потоках нет ничего нового, способ, которым с ними можно взаимодействовать в рамках платформы .NET, довольно прилично изменился (в лучшую сторону). Чтобы проложить себе путь к пониманию приемов построения многопоточных сборок, начнем с того, что посмотрим, каким образом можно взаимодействовать с процессами за счет применения библиотек базовых классов .NET.

В пространстве имен System.Diagnostics поставляется набор типов, которые позволяют программно взаимодействовать с процессами и различными связанными с диагностикой средствами вроде системного журнала событий и счетчиков производительности. Нас интересуют только те типы, которые позволяют взаимодействовать с процессами. Некоторые наиболее важные из них перечислены ниже:

Process

Предоставляет доступ к локальным и удаленным процессам, а также позволяет запускать и останавливать процессы программным образом

ProcessModule

Представляет модуль (*.dll или *.exe), который должен загружаться в определенный процесс. Важно понимать, что этот тип может применяться для представления любого модуля — COM, .NET или традиционного двоичного на базе C

ProcessModuleCollection

Позволяет создавать строго типизированную коллекцию объектов ProcessModule

ProcessStartInfo

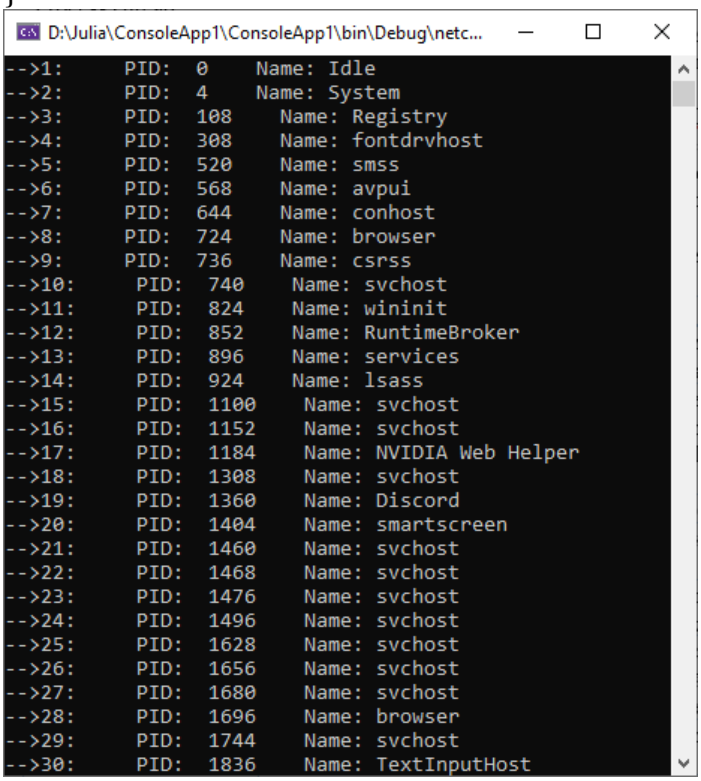
Позволяет указывать набор значений, которые должны использоваться при запуске процесса посредством метода Process.Start()
ProcessThread
Представляет поток внутри определенного процесса. Следует иметь в виду, что этот тип применяется для диагностики набора потоков в процессе, но не для ответвления внутри него новых потоков
ProcessThreadCollection
Позволяет создавать строго типизованную коллекцию объектов ProcessThread
Класс System.Diagnostics.Process позволяет анализировать процессы, которые выполняются на какой-то определенной машине (локальной или удаленной). Кроме того, в нем есть члены, которые позволяют программно запускать и останавливать процессы, просматривать приоритет процесса, а также получать список активных потоков или модулей, которые были загружены в данный процесс. В таблице перечислены некоторые наиболее важные свойства и методы класса System.Diagnostics.Process:

Свойство или метод	Описание
ExitTime	Позволяет извлекать значение даты и времени, ассоциируемое с процессом, который завершил свою работу (и представленное типом DateTime).
Handle	Это свойство возвращает дескриптор (представляемый с помощью IntPtr), который был назначен процессу операционной системой. Может оказаться полезным при создании приложений .NET, нуждающихся во взаимодействии с неуправляемым кодом.
Id	Позволяет получать идентификатор (PID) соответствующего процесса.
MachineName	Позволяет получать имя компьютера, на котором выполняется соответствующий процесс.
MainWindowTitle	Позволяет получать заголовок главного окна процесса (если у процесса нет главного окна, возвращается пустая строка).
Modules	Позволяет получать доступ к строго типизованной коллекции ProcessModuleCollection, представляющей набор модулей (*.dll или *.exe), которые были загружены в рамках текущего процесса.
ProcessName	Позволяет получать имя процесса (которое совпадает с именем самого приложения).
Responding	Позволяет получать значение, показывающее, реагирует ли пользовательский интерфейс соответствующего процесса на действия пользователя (и не находится ли он в текущий момент в "зависшем" состоянии).
StartTime	Позволяет получать информацию о том, когда был запущен соответствующий процесс (представленную типом DateTime).
Threads	Позволяет получать информацию о том, какие потоки выполняются в рамках соответствующего процесса (в виде коллекции объектов ProcessThread).
CloseMainWindow()	Этот метод позволяет завершать процесс, обладающий пользовательским интерфейсом, за счет отправки в его главное окно сообщения о закрытии.
GetCurrentProcess()	Этот статический метод возвращает новый объект Process, представляющий процесс, который является активным в текущий момент.
GetProcesses()	Этот статический метод возвращает массив новых объектов Process, которые выполняются на текущей машине.
Kill()	Этот метод позволяет немедленно останавливать соответствующий процесс.
Start()	Этот метод позволяет запускать процесс.

Давайте рассмотрем пример:

```
using System;
using System.Linq;
using System.Diagnostics;

namespace ConsoleApplication1
```

```
{
   class Program
   {
      static void Main()
      {
         // Используем LINQ-запрос для получения
         // списка всех используемых процессов
         var allProcess = from pr in Process.GetProcesses(".")
                          orderby pr.Id select pr;
         int i = 0;
         foreach (var proc in allProcess)
         {
            i++;
            string infoproc = string.Format(@"-->{0}:     PID:  {1}    Name: {2} ",i,proc.Id,proc.ProcessName);
            Console.WriteLine(infoproc);
         }
         Console.ReadLine();
      }
   }
}
```



ВЫПОЛНЕНИЕ РАБОТЫ:

Помимо полного списка всех выполняющихся методов на конкретной машине процессов, статический Process.GetProcessById () позволяет получать информацию и по конкретному объекту Процесс за счет указаний ассоциируемого с ним идентификатора (PID). В случае запроса несуществующего PID генерируется исключение ArgumentException .

Набор потоков представляется в виде строго типизованной коллекции ProcessThreadCollection, в которой содержится определенное количество отдельных объектов ProcessThread. Свойство Threads в System.Diagnostics.Process предоставляет доступ к классу ProcessThreadCollection.
CurrentPriority Позволяет получить информацию о текущем приоритете потока
Идентификатор Позволяет получить уникальный идентификатор потока
IdealProcessor Позволяет указать предпочтительный процессор для выполнения данного потока
PriorityLevel Позволяет получить или задать уровень приоритета потока
ПроцессорAffinity Позволяет указать процессоры.
StartAddress Позволяет, по какому адресу в памяти операционная система вызывала функцию, приведшую к запуску данного потока
Начальное время Позволяет узнать, когда операционная система запустила поток
ThreadState Позволяет узнать текущее состояние данного потока
TotalProcessorTime Позволяет узнать, сколько всего времени данный поток использует процессор
WaitReason Позволяет узнать причину, по которой поток находится в состоянии ожидания

Тип ProcessThread не является сущностью, применяемой для создания, приостановки и уничтожения потоков в .NET. Он скорее представляет собой средство, позволяющее получать диагностическую информацию по активным потокам Windows внутри выполняющегося процесса.
Теперь давайте посмотрим, как реализовать проход по загруженным модулям, которые обслуживаются в конкретном процессе. При обсуждении модуля - это общий инструмент, используемый для описания заданной сборки * .dll (или * .exe), который обслуживается в определенном процессе. При получении доступа к ProcessModuleCollection через свойство Process.Modules можно извлечь список всех модулей, которые обслуживаются внутри: .NET, COM и основанных на C библиотек.
И наконец, последними членами класса System.Diagnostics.Process, которые остались рассмотреть, являются методы Start () и Kill (). Эти методы позволяют, соответственно, программно запускать и завершать процесс. Метод Start () может принимать тип System.Diagnostics.ProcessStartInfo и дополнительные фрагменты информации относительно запуска определенного процесса.
Давайте рассмотрим пример глобального изучения процессов:

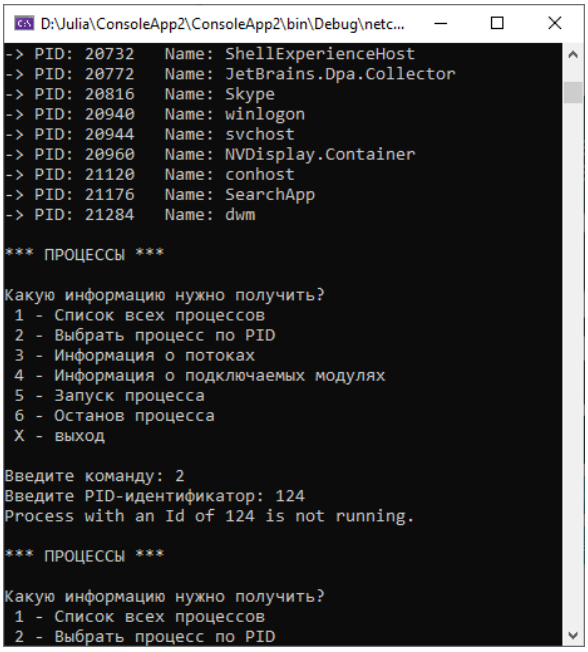
```
using System;
using System.Linq;
using System.Diagnostics;

namespace ConsoleApplication1
{
   class Program
   {
      static void Main()
      {
         Console.WriteLine("*** ПРОЦЕССЫ ***\n");
         string comm = Command();
         UseMyCommand(comm);
         // Используем рекурсию для вызова новых команд
         if (comm != "X")
              Main();
      }

      static string Command()
      {
         Console.WriteLine("Какую информацию нужно получить? \n"+
            " 1 - Список всех процессов\n 2 - Выбрать процесс по PID\n"+
            " 3 - Информация о потоках\n 4 - Информация о подключаемых модулях\n"+
```

```
        " 5 - Запуск процесса\n 6 - Останов процесса\n X - выход\n");
    Console.Write("Введите команду: ");
    string comm = Console.ReadLine();
    return comm;
}

static void UseMyCommand(string str)
{
    switch (str)
    {
        case "X":
            break;
        case "1":
            AllInfoProcess();
            break;
        case "2":
            ProcInMyPid();
            break;
        case "3":
            Threads();
            break;
        case "4":
            InfoByModuleProc();
            break;
        case "5":
            StartProcess();
            break;
        case "6":
            StopProcess();
            break;
        default:
            Console.ForegroundColor = ConsoleColor.Red;
            Console.WriteLine("Команда не распознана!");
            Console.ForegroundColor = ConsoleColor.Gray;
            break;
    }
    Console.WriteLine();
}

static void AllInfoProcess()
{
    var myProcess = from proc in Process.GetProcesses(".")
                    orderby proc.Id select proc;
    Console.WriteLine("\n*** Текущие процессы ***\n");
    foreach (var p in myProcess)
        Console.WriteLine("-> PID: {0}\tName: {1}",p.Id,p.ProcessName);
}

static void ProcInMyPid()
{
    Console.Write("Введите PID-идентификатор: ");
    string pid = Console.ReadLine();
```

```
    Process myProc = null;
    try
    {
        int i = int.Parse(pid);
        myProc = Process.GetProcessById(i);
        Console.WriteLine("\n-> PID: {0}\tName: {1}\n",myProc.Id,myProc.ProcessName);
    }
    catch (Exception ex)
    {
        Console.WriteLine(ex.Message);
    }
}

static void Threads()
{
    Console.Write("Введите PID-идентификатор: ");
    string pid = Console.ReadLine();
    Process myProc = null;
    try
    {
        int i = int.Parse(pid);
        myProc = Process.GetProcessById(i);
        // Получаем коллекцию потоков процесса
        ProcessThreadCollection threads = myProc.Threads;
        Console.WriteLine("Потоки процесса {0}:\n", myProc.ProcessName);
        foreach (ProcessThread pt in threads)
            Console.WriteLine("-> Thread ID: {0}\tВремя: {1}\tПриоритет: {2}"
                , pt.Id, pt.StartTime.ToShortTimeString(), pt.PriorityLevel);
    }
    catch (Exception ex)
    {
        Console.WriteLine(ex.Message);
    }
    finally
    {
        Console.WriteLine();
    }
}

static void InfoByModuleProc()
{
    Console.Write("Введите PID-идентификатор: ");
    string pid = Console.ReadLine();
    Process myProc = null;
    try
    {
        int i = int.Parse(pid);
        myProc = Process.GetProcessById(i);
        Console.WriteLine("Подключаемые модули процесса {0}:\n", myProc.ProcessName);
        // Получаем коллекцию модулей
        ProcessModuleCollection mods = myProc.Modules;
        foreach (ProcessModule pm in mods)
```

```
                Console.WriteLine("-> Имя модуля: "+pm.ModuleName);
            }
            catch (Exception ex)
            {
               Console.WriteLine(ex.Message);
            }
            finally
            {
               Console.WriteLine();
            }
        }

        static void StartProcess()
        {
            Process myProc = null;
            // Запустить сайт sevsu.ru на машине пользователя
            try
            {               // Будет работать только для пользавателей у которых установлен браузер Google
Chrome
               // Чтобы запустить IE, нужно запустить процесс IExplore.exe
               myProc = Process.Start("chrome.exe", "sevsu.ru");
            }
            catch (Exception ex)
            {               Console.WriteLine(ex);
            }
        }
        static void StopProcess()
        {           Console.Write("Введите PID-идентификатор процесса, который нужно остановить: ");
            string pid = Console.ReadLine();
            Process myProc = null;
            try
            {               int i = int.Parse(pid);
               myProc = Process.GetProcessById(i);
               myProc.Kill();
            }
            catch (Exception ex)
            {
               Console.WriteLine(ex.Message);
            }
        }
    }
}
```

Это приложение является многофункциональным и позволяет просмотреть все текущие процессы на машине пользователя, найти процесс по PID-системе, получить информацию о потоках и подключаемых модулях процесса, а также запустить его выполнить (в качестве примера запуска процесса приведен запуск Google Chrome и передача URL sevsu.ru).



Оформить отчет по всем командам меню со скриншотами выполнения и с пояснениями в коде по методам и свойствам процессов и потоков.

ЛИТЕРАТУРА (рекомендуемая)

1. Руководство по языку C#
   https://docs.microsoft.com/ru-ru/dotnet/api/system.diagnostics.process?view=net-6.0
   https://docs.microsoft.com/ru-ru/dotnet/api/system.threading.thread?view=net-6.0

2. *Гостев, И. М.* Операционные системы : учебник и практикум для среднего профессионального образования / И. М. Гостев. — 2-е изд., испр. и доп. — Москва : Издательство Юрайт, 2019. — 164 с. — (Профессиональное образование). — ISBN 978-5-534-04951-0. — Текст : электронный // ЭБС Юрайт [сайт]. — URL: https://biblio-online.ru/bcode/438283 с.45-59

Практическое занятие №2

Исследование работы планировщиков процессов в мультипрограммных операционных системах

ЦЕЛЬ РАБОТЫ:

1) знакомство с программой имитирующей модель управления работой планировщика процессов в операционной системе;
2) сравнение методов планирования по различным критериям.

ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ:

Операционные системы различаются

- особенностями реализации внутренних алгоритмов управления основными ресурсами компьютера (процессорами, памятью, устройствами),
- особенностями использованных методов проектирования,
- типами аппаратных платформ,
- критериями эффективности,
- особенностями реализации сетевых решений
- и многими другими свойствами.

Мультипрограммирование, или многозадачность, — это способ организации вычислительного процесса, при котором на одном процессоре выполняются сразу несколько программ (создается видимость одновременного выполнения нескольких программ).

Мультипрограммирование применяется для повышения эффективности вычислительной системы.

Эффективность же может пониматься как

- общая пропускная способность вычислительной системы;
- удобство работы пользователей, например, возможность интерактивной работы для нескольких пользователей или возможность работы одного пользователя с несколькими приложениями на одной машине;
- реактивность системы – то есть способность системы выдерживать заранее заданные (возможно, очень короткие) интервалы времени между запуском программы и получением результата.

В зависимости от выбранного критерия эффективности операционные системы делятся на:

- системы пакетной обработки (например, ОС ЕС),
- системы разделения времени (UNIX, VMS),
- системы реального времени (QNX, RT/11).

Системы пакетной обработки служат для решения задач в основном вычислительного характера, не требующих быстрого получения результатов.

Главной целью и критерием эффективности таких систем является максимальная пропускная способность, то есть решение максимального числа задач в единицу времени.

Схема функционирования систем пакетной обработки данных:

- в начале работы формируется пакет заданий, каждое задание содержит требование к системным ресурсам;
- из этого пакета заданий формируется мультипрограммная смесь, то есть множество одновременно выполняемых задач. Для одновременного выполнения выбираются задачи, предъявляющие отличающиеся требования к ресурсам, так, чтобы обеспечивалась сбалансированная загрузка всех устройств вычислительной машины; (в мультипрограммной смеси желательно одновременное присутствие вычислительных задач и задач с интенсивным вводом-выводом).

Таким образом, выбор нового задания из пакета заданий зависит от внутренней ситуации, складывающейся в системе, то есть выбирается "выгодное" задание.

В таких ОС невозможно гарантировать выполнение того или иного задания в течение определенного периода времени.

В системах пакетной обработки переключение процессора с выполнения одной задачи на выполнение другой происходит только в случае, если активная задача сама отказывается от процессора, например, из-за необходимости выполнить операцию ввода-вывода. Поэтому одна задача может надолго занять процессор, что делает невозможным выполнение интерактивных задач.

Взаимодействие пользователя с вычислительной машиной, на которой установлена система пакетной обработки, сводится к тому, что он приносит задание, отдает его диспетчеру-оператору, а в конце дня после выполнения всего пакета заданий получает результат. Очевидно, что такой порядок снижает эффективность работы пользователя.

Основной недостаток систем пакетной обработки - изоляция пользователя-программиста от процесса выполнения его задач.

Системы разделения времени (интерактивные) призваны исправить недостаток систем пакетной обработки данных.

Каждому пользователю системы разделения времени предоставляется терминал, с которого он может вести диалог со своей программой.

Каждой задаче выделяется только квант процессорного времени,

Þ ни одна задача не занимает процессор надолго, и время ответа оказывается приемлемым.

Если квант выбран достаточно небольшим, то у всех пользователей, одновременно работающих на одной и той же машине, складывается впечатление, что каждый из них единолично использует машину.

Системы разделения времени обладают меньшей пропускной способностью, чем системы пакетной обработки, так как

- на выполнение принимается каждая запущенная пользователем задача, а не та, которая "выгодна" системе,
- увеличивается время работы, так как выполняется более частое переключение процессора с задачи на задачу.

Таким образом, критерием эффективности систем разделения времени является не максимальная пропускная способность, а удобство и эффективность работы пользователя.

Системы реального времени применяются для управления различными техническими объектами, такими, например, как станок, спутник, научная экспериментальная установка или технологическими процессами, такими, как гальваническая линия, доменный процесс и т.п.

Существует предельно допустимое время, в течение которого должна быть выполнена та или иная программа, управляющая объектом, в противном случае может произойти авария: спутник выйдет из зоны видимости, экспериментальные данные, поступающие с датчиков, будут потеряны, толщина гальванического покрытия не будет соответствовать норме.

Таким образом, критерием эффективности для систем реального времени является их способность выдерживать заранее заданные интервалы времени между запуском программы и получением результата (управляющего воздействия).

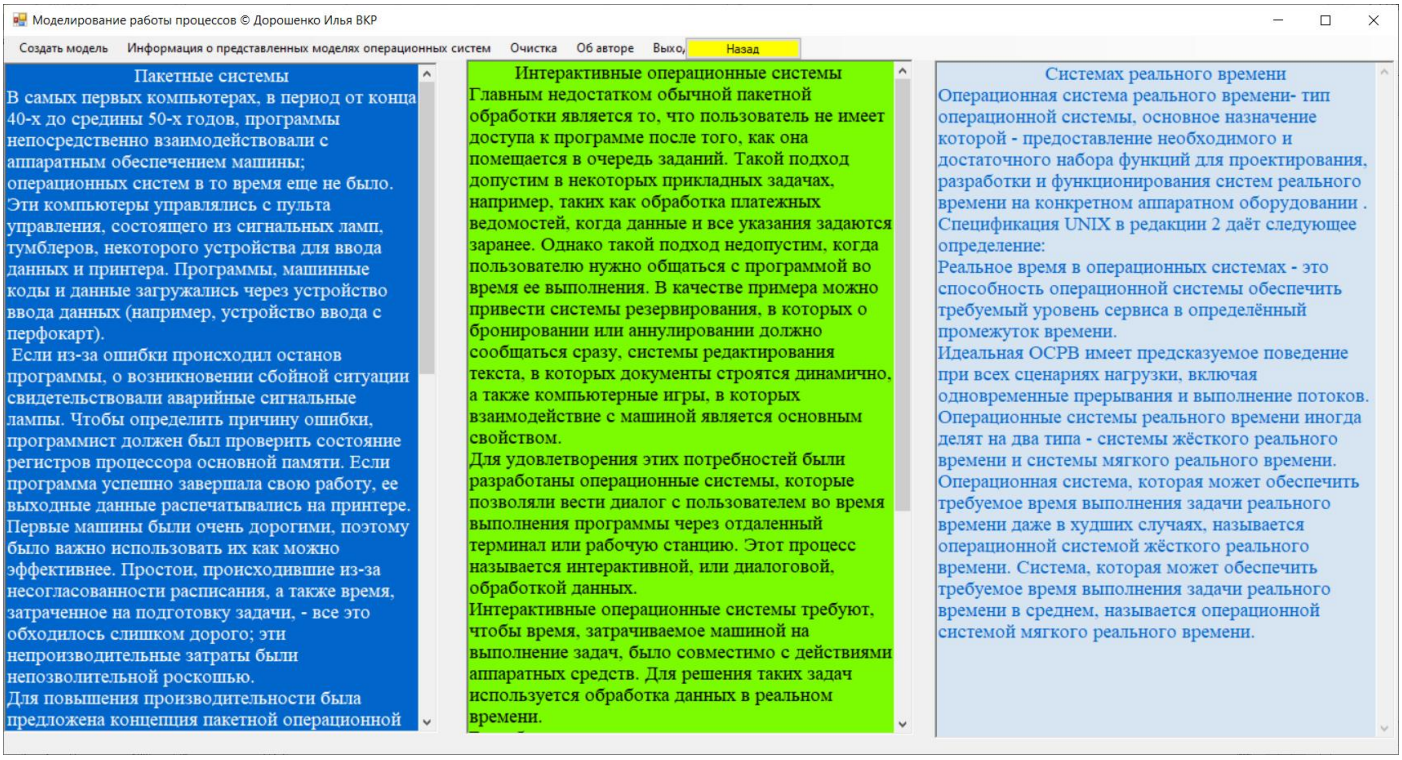
Это время называется временем реакции системы, а соответствующее свойство системы - реактивностью.

Для этих систем мультипрограммная смесь представляет собой фиксированный набор заранее разработанных программ, а выбор программы на выполнение осуществляется исходя из текущего состояния объекта или в соответствии с расписанием плановых работ.

Замечание. Некоторые операционные системы могут совмещать в себе свойства систем разных типов, например, часть задач может выполняться в режиме пакетной обработки, а часть - в режиме реального времени или в режиме разделения времени. В таких случаях режим пакетной обработки часто называют фоновым режимом.

ХОД РАБОТЫ:

1. Скачать с диска приложение planprocess для исследования алгоритмов планирования процессов;
2. Выбрать из таблицы вариантов свой вариант по списку алгоритм по каждому виду систем;
3. Ввести для каждого алгоритма данные о пяти процессах в системе;
4. Сравнить эффективность работы алгоритмов и выявить процесс, который выполнится самым первым (почему?), последним.



СОСТАВ ОТЧЕТА:

Титульный лист

Цель работы

Ход работы: сведения о типе системы по критериям эффективности, сведения об алгоритме планирования процессов, скриншоты введенных данных о процессах, скриншоты диаграммы/таблицы выполнения процессов

Выводы.

Отчет предъявить преподавателю для оценивания прикрепить в системе testmoodle.sevsu.ru в виде отдельного файла

Таблица вариантов

№	Алгоритм систем пакетной обработки	Алгоритм систем разделения времени	Алгоритм систем реального времени
1.	Неприоритетные алгоритмы FIFO	Циклическое	статические
2.	Неприоритетные алгоритмы SJF	Приоритетное	динамические
3.	Приоритетные алгоритмы – по наименьшему алгоритму планирования	Циклическое	динамические
4.	Неприоритетные алгоритмы FIFO	Приоритетное	статические
5.	Неприоритетные алгоритмы SJF	Циклическое	статические
6.	Приоритетные алгоритмы – по наименьшему алгоритму планирования	Приоритетное	динамические
7.	Неприоритетные алгоритмы SJF	Циклическое	динамические
8.	Неприоритетные алгоритмы FIFO	Приоритетное	динамические
9.	Неприоритетные алгоритмы SJF	Циклическое	статические
10.	Приоритетные алгоритмы – по наименьшему алгоритму планирования	Приоритетное	статические
11.	Неприоритетные алгоритмы FIFO	Циклическое	динамические
12.	Приоритетные алгоритмы – по наименьшему алгоритму планирования	Приоритетное	динамические

13.	Неприоритетные алгоритмы FIFO	Циклическое	динамические
14.	Неприоритетные алгоритмы SJF	Приоритетное	статические
15.	Приоритетные алгоритмы – по наименьшему алгоритму планирования	Циклическое	статические
16.	Приоритетные алгоритмы – по наименьшему алгоритму планирования	Приоритетное	динамические
17.	Неприоритетные алгоритмы SJF	Циклическое	статические
18.	Неприоритетные алгоритмы FIFO	Приоритетное	статические
19.	Неприоритетные алгоритмы SJF	Циклическое	динамические
20.	Приоритетные алгоритмы – по наименьшему алгоритму планирования	Приоритетное	статические
21.	Приоритетные алгоритмы – по наименьшему алгоритму планирования	Циклическое	статические
22.	Неприоритетные алгоритмы SJF	Приоритетное	статические
23.	Неприоритетные алгоритмы FIFO	Циклическое	динамические
24.	Приоритетные алгоритмы – по наименьшему алгоритму планирования	Приоритетное	динамические
25.	Неприоритетные алгоритмы FIFO	Циклическое	динамические

или ссылки на облачное хранилище.

КОНТРОЛЬНЫЕ ВОПРОСЫ:

1. Дайте определение «процесс», «планировщик процесса».
2. В каких состояниях может находится процесс?
3. Какую информацию о процессе хранит операционная система?
4. Какие очереди процессов формирует система? Каков принцип обслуживания очереди?
5. Дайте определение мультипрограммирование.
6. По каким критериям классифицируются операционные системы?
7. Какие приоритеты может получать процесс в системе?
8. Какие приоритетные алгоритмы планирования вам известны? Неприоритетные?

Практическое занятие №3
СОЗДАНИЕ И ДИСПЕТЧЕРИЗАЦИЯ ПОТОКОВ

ЦЕЛЬ РАБОТЫ:
знакомство со средствами создания и управления потоками; исследование влияния приоритета потока на время его выполнения при различной загрузке ЦП.

ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ:

Для работы с потоками в C# необходимо подключить специальные директивы:

```
using System.Threading;
using System.Threading.Tasks;
```

Любой поток в C# должен обязательно происходить в каком-либо методе или функции, поэтому для работы с новым потоком необходимо сначала создать для него, например, метод, который и будет точкой входа для этого потока.

Потоки в C# начинают выполняться не сразу после их инициализации. Каждый из созданных потоков необходимо сначала запустить. Делается это следующим образом: имя_потока.Start();. В случае из нашего примера строка запуска будет такой:

mythread.Start();

Если записать в методе Main сразу после инициализации самого потока, то, при запуске программы, как только в Main выполнится метод Start(), начнёт работать вновь созданный поток.

```
static void mythread1()
      {
         for (int i = 0; i < 10; i++)
         {
            Console.WriteLine("Поток 1 выводит " + i);
         }
      }

      static void mythread2()
      {
         for (int i = 0; i < 10; i++)
         {
            Console.WriteLine("Поток 2 выводит " + i);
         }
      }

      static void Main(string[] args)
      {
         Thread thread1 = new Thread(mythread1);
         Thread thread2 = new Thread(mythread2);

         thread1.Start();
         thread2.Start();

         for (int i = 0; i < 10; i++)
         {
            Console.WriteLine("Поток 3 выводит " + i);
         }
         Console.ReadLine();
      }
```

Иногда бывают такие сценарии, когда необходимо бывает приостановить поток, например для того, чтобы пропустить другие потоки и не мешать им выполнять свою работу, либо для снижения потребления процессорного времени.

Thread.Sleep(1000);

В этой строке мы указали, что поток, к которому будет относиться данная инструкция, будет приостановлен на 1000 миллисекунд. Ззначение в скобках указывает, на какой период времени в миллисекундах будет приостановлен поток. Как только заданное время пройдёт, поток снова возобновит свою работу и продолжит с места, на котором остановился. Кстати говоря, число, заданное в скобках — это количество миллисекунд для задержки в идеале, на практике поток может

возобновиться на несколько миллисекунд позже или раньше, это зависит не только от программы, но и от характеристик операционной системы.

В метод Sleep также можно передать и значение ноль:

Thread.Sleep(0);

В таком случае поток приостановится на положенный ему интервал времени (мы же помним, что все потоки выполняются по кусочку и по очереди, и при этом на каждый такой кусочек выделяется очень небольшое количество времени, но достаточное для того, чтобы выполнить некоторое количество действий), который он «отдаёт» любому другому готовому к выполнению потоку с таким же приоритетом, как и у него (про приоритеты поговорим ниже). Если же не удастся найти ни один такой «ожидающий» поток, то выполнение текущего потока не будет приостановлено, и он продолжит свою работу.

Приоритеты потоков в C#
Когда в программе фигурирует несколько потоков, выбор процессором следующего потока для выполнения не является рандомным. Дело в том, что у каждого потока имеется значение приоритета, чем выше приоритет потока, тем важнее для процессора предоставить время и ресурсы для выполнения именно ему. Если же приоритет потока не слишком высокий, значит он может и подождать в очереди, пока выполняются более приоритетные потоки.

Всего существует пять вариантов приоритетов потоков в C#:

Highest — самый высокий
AboveNormal — выше среднего
Normal — стандартный
BelowNormal — ниже среднего
Lowest — самый низкий
Как уже было сказано, чем выше приоритет, тем быстрее он выполнится, опережая потоки с меньшим значением. Если у потоков одинаковые приоритеты, они выполняются по очереди.

По умолчанию все создаваемые потоки в C# имеют стандартный приоритет Normal. Однако приоритеты потоков можно и менять, делается это так:

имяпотока.Priority = ThreadPriority.вариантприоритета;

ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ

1. Разработать приложение, рассмотренное в примере, проверить его работу.
2. Разработать приложение с задержкой sleep.
3. Установить для потоков разные приоритеты.
4. Запустить поочередно приложения, меняя приоритет потоков низший, нормальный, самый высший. Как изменится производительность потоков приложения от того, какой уровень приоритета выставлен в потоке? Отследить в диспетчере задач.
Составить отчет по работе с обязательными скриншотами выполнения и выводами по работе.

Практическое занятие №4
СРЕДСТВА СИНХРОНИЗАЦИИ ПОТОКОВ

Цель работы: практическое знакомство с методами синхронизации потоков процессов.

Теоретические сведения:
Когда используется несколько потоков, то иногда приходится координировать действия двух или более потоков. Процесс достижения такой координации называется синхронизацией. Самой распространенной причиной применения синхронизации служит необходимость разделять среди двух или более потоков общий ресурс, который может быть одновременно доступен только одному потоку. Например, когда в одном потоке выполняется запись информации в файл, второму потоку должно быть запрещено делать это в тот же самый момент времени. Синхронизация требуется и в том случае, если один поток ожидает событие, вызываемое другим потоком. В подобной ситуации требуются какие-то средства, позволяющие приостановить один из потоков до тех пор, пока не произойдет событие в другом потоке. После этого ожидающий поток может возобновить свое выполнение.

Основные средства синхронизации потоков в C#

Sleep	Блокировка на указанное время.
Join	Ожидание окончания другого потока.
lock (Monitor)	Гарантирует, что только один поток может получить доступ к ресурсу или секции кода.
Mutex	То же. Может использоваться для предотвращения запуска нескольких экземпляров приложения.
Semaphore	Гарантирует, что не более заданного числа потоков может получить доступ к ресурсу или секции кода.
EventWaitHandle	Позволяет потоку ожидать сигнала от другого потока.

В основу синхронизации положено понятие блокировки, посредством которой организуется управление доступом к кодовому блоку в объекте. Когда объект заблокирован одним потоком, остальные потоки не могут получить доступ к заблокированному кодовому блоку. Когда же блокировка снимается одним потоком, объект становится доступным для использования в другом потоке.

Средство блокировки встроено в язык C#. Благодаря этому все объекты могут быть синхронизированы. Синхронизация организуется с помощью ключевого слова lock.. Она была предусмотрена в C# с самого начала, и поэтому пользоваться ею намного проще, чем кажется на первый взгляд. В действительности синхронизация объектов во многих программах на C# происходит практически незаметно.

Ниже приведена общая форма блокировки:

```
lock(lockObj) {
// синхронизируемые операторы
}
```

где lockObj обозначает ссылку на синхронизируемый объект. Если же требуется синхронизировать только один оператор, то фигурные скобки не нужны. Оператор lock гарантирует, что фрагмент кода, защищенный блокировкой для данного объекта, будет использоваться только в потоке, получающем эту блокировку. А все остальные потоки блокируются до тех пор, пока блокировка не будет снята. Блокировка снимается по завершении защищаемого ею фрагмента кода.

Блокируемым считается такой объект, который представляет синхронизируемый ресурс. В некоторых случаях им оказывается экземпляр самого ресурса или же произвольный экземпляр объекта, используемого для синхронизации. Следует, однако, иметь в виду, что блокируемый объект не должен быть общедоступным, так как в противном случае он может быть заблокирован из другого, неконтролируемого в программе фрагмента кода и в дальнейшем вообще не разблокируется.

Рассмотрим пример:

```
int x = 0;

// запускаем пять потоков
for (int i = 1; i < 6; i++)
{
    Thread myThread = new Thread(Print);
    myThread.Name = $"Поток {i}";   // устанавливаем имя для каждого потока
    myThread.Start();
}

void Print()
{
    x = 1;
    for (int i = 1; i < 6; i++)
    {
        Console.WriteLine($"{Thread.CurrentThread.Name}: {x}");
        x++;
        Thread.Sleep(100);
    }
}
```

Здесь у нас запускаются пять потоков, которые вызывают метод Print и которые работают с общей переменной x. И мы предполагаем, что метод выведет все значения x от 1 до 5. И так для каждого потока. Однако в реальности в процессе работы будет происходить переключение между потоками, и значение переменной x становится непредсказуемым. Например, в моем случае я получил следующий консольный вывод (он может в каждом конкретном случае различаться):

```
Поток 1: 1
Поток 5: 1
Поток 4: 1
Поток 2: 1
Поток 3: 1
Поток 1: 6
Поток 5: 7
Поток 3: 7
Поток 2: 7
Поток 4: 9
Поток 1: 11
Поток 4: 11
Поток 2: 11
Поток 3: 14
Поток 5: 11
Поток 1: 16
Поток 2: 16
Поток 3: 16
```

Поток 5: 18
Поток 4: 16
Поток 1: 21
Поток 5: 21
Поток 3: 21
Поток 2: 21
Поток 4: 21

Решение проблемы состоит в том, чтобы синхронизировать потоки и ограничить доступ к разделяемым ресурсам на время их использования каким-нибудь потоком. Для этого используется ключевое слово lock. Оператор lock определяет блок кода, внутри которого весь код блокируется и становится недоступным для других потоков до завершения работы текущего потока. Остальный потоки помещаются в очередь ожидания и ждут, пока текущий поток не освободит данный блок кода. В итоге с помощью lock мы можем переделать предыдущий пример следующим образом:

```
int x = 0;
object locker = new object();  // объект-заглушка
// запускаем пять потоков
for (int i = 1; i < 6; i++)
{
   Thread myThread = new Thread(Print);
   myThread.Name = $"Поток {i}";
   myThread.Start();
}

void Print()
{
   lock (locker)
   {
      x = 1;
      for (int i = 1; i < 6; i++)
      {
         Console.WriteLine($"{Thread.CurrentThread.Name}: {x}");
         x++;
         Thread.Sleep(100);
      }
   }
}
```

Для блокировки с ключевым словом lock используется объект-заглушка, в данном случае это переменная locker. Обычно это переменная типа object. И когда выполнение доходит до оператора lock, объект locker блокируется, и на время его блокировки монопольный доступ к блоку кода имеет только один поток. После окончания работы блока кода, объект locker освобождается и становится доступным для других потоков.

В этом случае консольный вывод будет более упорядоченным:

Поток 1: 1
Поток 1: 2
Поток 1: 3
Поток 1: 4
Поток 1: 5
Поток 5: 1
Поток 5: 2

```
Поток 5: 3
Поток 5: 4
Поток 5: 5
Поток 3: 1
Поток 3: 2
Поток 3: 3
Поток 3: 4
Поток 3: 5
Поток 2: 1
Поток 2: 2
Поток 2: 3
Поток 2: 4
Поток 2: 5
Поток 4: 1
Поток 4: 2
Поток 4: 3
Поток 4: 4
Поток 4: 5
```

МЬЮТЕКСЫ

Еще один инструмент управления синхронизацией потоков представляет класс Mutex или мьютекс, который также располагается в пространстве имен System.Threading.
Сначала создаем объект мьютекса:
Mutex mutexObj = new Mutex()

Основную работу по синхронизации выполняют методы WaitOne() и ReleaseMutex(). Метод mutexObj.WaitOne() приостанавливает выполнение потока до тех пор, пока не будет получен мьютекс mutexObj. Изначально мьютекс свободен, поэтому его получает один из потоков. После выполнения всех действий, когда мьютекс больше не нужен, поток освобождает его с помощью метода mutexObj.ReleaseMutex(). А мьютекс получает один из ожидающих потоков.

Таким образом, когда выполнение дойдет до вызова mutexObj.WaitOne(), поток будет ожидать, пока не освободится мьютекс. И после его получения продолжит выполнять свою работу.

СЕМАФОРЫ

Семафоры являются еще одним инструментом, который предлагает нам платформа .NET для управления синхронизацией. Семафоры позволяют ограничить количество потоков, которые имеют доступ к определенным ресурсам. В .NET семафоры представлены классом Semaphore.

Для создания семафора применяется один из конструкторов класса Semaphore:

Semaphore (int initialCount, int maximumCount): параметр initialCount задает начальное количество потоков, а maximumCount - максимальное количество потоков, которые имеют доступ к общим ресурсам

Semaphore (int initialCount, int maximumCount, string? name): в дополнение задает имя семафора

Semaphore (int initialCount, int maximumCount, string? name, out bool createdNew): последний параметр - createdNew при значении true указывает, что новый семафор был успешно создан. Если этот параметр равен false, то семафор с указанным именем уже существует

Для работы с потоками класс Semaphore имеет два основных метода:

WaitOne(): ожидает получения свободного места в семафоре

Release(): освобождает место в семафоре.

Класс EventWaitHandle

Поля
WaitTimeout
Указывает, что время ожидания операции WaitAny(WaitHandle[], Int32, Boolean) истекло до получения сигнала каким-либо из дескрипторов ожидания. Это поле является константой.

Свойства
Handle Является устаревшей. Возвращает или задает собственный дескриптор операционной системы.
SafeWaitHandle Возвращает или задает собственный дескриптор операционной системы.

Методы
Close() Освобождает все ресурсы, удерживаемые текущим объектом WaitHandle.
CreateObjRef(Type) Создает объект, который содержит всю необходимую информацию для создания прокси-сервера, используемого для взаимодействия с удаленным объектом.
Dispose() Освобождает все ресурсы, используемые текущим экземпляром класса WaitHandle.
Equals(Object) Определяет, равен ли указанный объект текущему объекту.
GetType() Возвращает объект Type для текущего экземпляра.
OpenExisting(String) Открывает указанное именованное событие синхронизации, если оно уже существует.
Reset() Задает несигнальное состояние события, вызывая блокирование потоков.
Set() Задает сигнальное состояние события, позволяя одному или нескольким ожидающим потокам продолжить.
WaitOne() Блокирует текущий поток до получения сигнала объектом WaitHandle.

(Унаследовано от WaitHandle)

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

using System.Threading;

namespace ConsoleApplication7 {
   class Program {
      static EventWaitHandle ewh = new AutoResetEvent(false);
      static void Main(string[] args) {
         //Запуск потока
         new Thread(Run).Start();
         //Ожидание...
         Console.WriteLine("Waiting...");
         ewh.WaitOne();
         //Получили сигнал!
         Console.WriteLine("Get signal!");
```

```
            Console.ReadLine();
        }
        static void Run() {
            //Ждем...
            Thread.Sleep(3000);
            //Сигналим о завершении
            ewh.Set();
        }
    }
}
```

ВЫПОЛНЕНИЕ РАБОТЫ:

На оценку «удовлетворительно» создать приложение по представленному алгоритму и проверить его работу с критическими секциями lock. Написать вывод о синхронизации потоков с помощью критических секций.
На оценку «хорошо» заменить в приложении средство синхронизации lock на мьютексы. Сравнить работу приложений, сделать вывод об использовании средств синхронизации потоков в приложениях.
На оценку «отлично» заменить в приложении средство синхронизации lock на семафоры или сигналы. Сравнить работу приложений, сделать вывод об использовании средств синхронизации потоков в приложениях.

ЛИТЕРАТУРА (рекомендуемая)

1. Лекции по курсу операционные системы.
2. https://metanit.com/sharp/tutorial/11.4.php lock
3. https://docs.microsoft.com/ru-ru/dotnet/standard/threading/overview-of-synchronization-primitives методы синхронизации платформы Net

Практическое занятие №5
СРЕДСТВА ОБМЕНА ДАННЫМИ МЕЖДУ ПРИЛОЖЕНИЯМИ. БУФЕР ОБМЕНА

Цель работы: практическое знакомство со средствами передачи данных между процессами, (Interprocess Communications-IPC), выполняющимися на одном компьютере.

Краткие теоретические сведения

Простейшие приемы работы с буфером обмена.
Класс Clipboard

Предоставляет статические методы, которые способствуют двунаправленной передаче данных в системный буфер обмена.

Методы

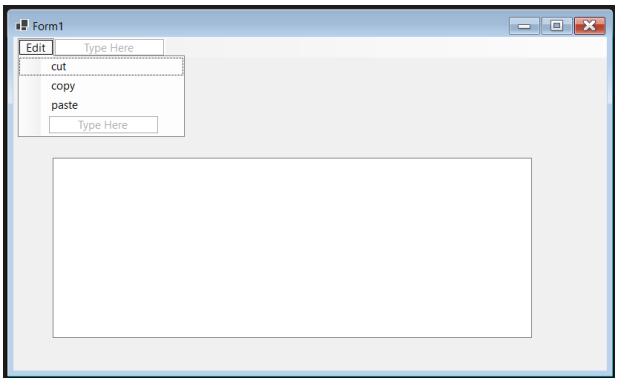
Clear()	Очищает любые данные из системного буфера обмена.
ContainsAudio()	Запрашивает буфер обмена на наличие данных в формате WaveAudio.
ContainsData(String)	Запрашивает буфер обмена на наличие данных в указанном формате.
ContainsFileDropList()	Запрашивает буфер обмена на наличие данных в формате FileDrop.
ContainsImage()	Запрашивает буфер обмена на наличие данных в формате Bitmap.
ContainsText()	Запрашивает буфер обмена на наличие данных в формате UnicodeText.
ContainsText(TextDataFormat)	Запрашивает буфер обмена на наличие данных в текстовом формате.
Flush()	Окончательно добавляет данные из Clipboard, чтобы они были доступны после закрытия исходного приложения.

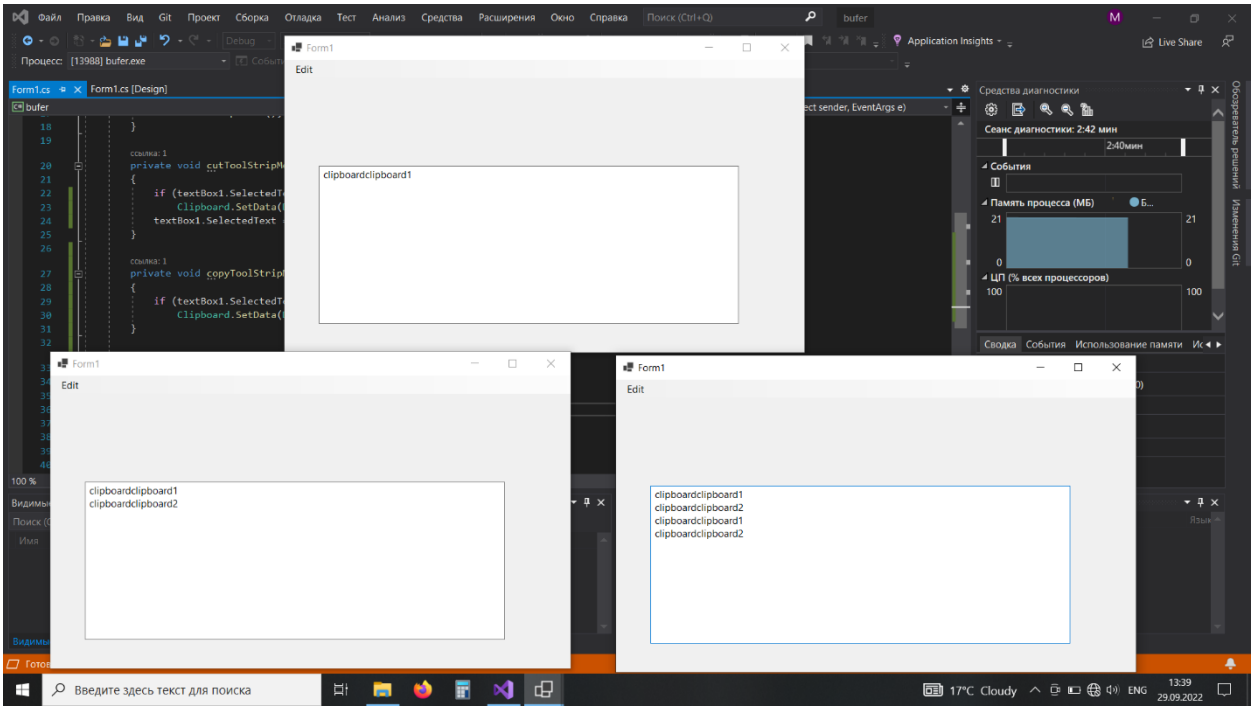
GetAudioStream() Возвращает поток данных из буфера обмена в формате WaveAudio.
GetData(String) Извлекает данные в указанном формате из буфера обмена.
GetDataObject() Возвращает объект данных, представляющий все содержимое буфера обмена.
GetFileDropList() Возвращает коллекцию строк, содержащую список перенесенных файлов, доступный в буфере обмена.
GetImage() Возвращает объект BitmapSource из буфера обмена, содержащий данные в формате Bitmap.
GetText() Возвращает строку, содержащую данные UnicodeText из буфера обмена.
GetText(TextDataFormat) Возвращает строку, содержащую текстовые данные из буфера обмена.
IsCurrent(IDataObject) Сравнивает указанный объект данных с содержимым буфера обмена.
SetAudio(Byte[]) Сохраняет аудиоданные (в формате WaveAudio) в буфере обмена. Аудиоданные указываются в виде массива байтов.
SetAudio(Stream) Сохраняет аудиоданные (в формате WaveAudio) в буфере обмена. Аудиоданные указываются в виде потока.
SetData(String, Object) Сохраняет указанные данные в буфере обмена в указанном формате.
SetDataObject(Object) Размещает указанный непостоянный объект данных в системном буфере обмена.
SetDataObject(Object, Boolean) Размещает указанный объект данных в системном буфере обмена и принимает логический параметр, указывающий, следует ли оставлять этот объект в буфере обмена при выходе из приложения.
SetFileDropList(StringCollection) Сохраняет в буфере обмена данные FileDrop. Список перенесенных файлов указывается в виде коллекции строк.
SetImage(BitmapSource) Сохраняет в буфере обмена данные Bitmap. Данные изображения обрабатываются как BitmapSource.
SetText(String) Сохраняет в буфере обмена данные UnicodeText.
SetText(String, TextDataFormat) Сохраняет текстовые данные в буфере обмена в указанном текстовом формате. Данные UnicodeText для сохранения указываются в виде строки.

// текст

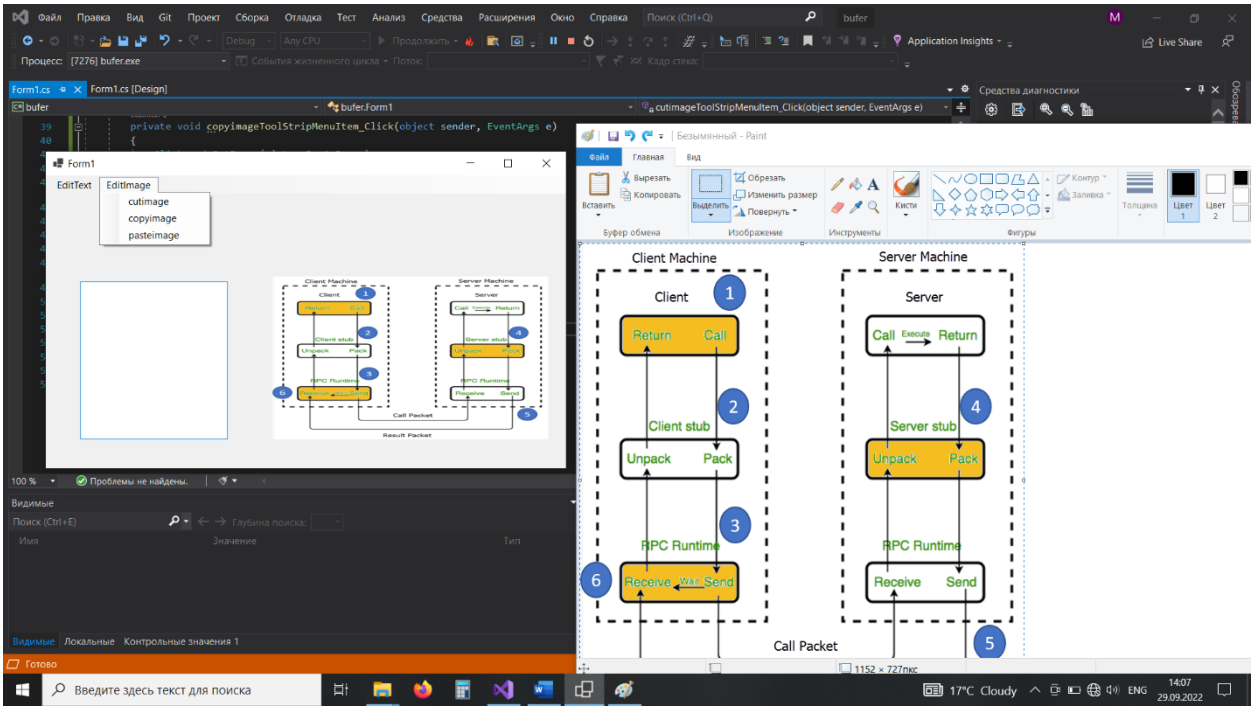
```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace bufer
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        private void cutToolStripMenuItem_Click(object sender, EventArgs e)
        {
            if (textBox1.SelectedText != string.Empty)
                Clipboard.SetData(DataFormats.Text, textBox1.SelectedText);
            textBox1.SelectedText = string.Empty;
        }

        private void copyToolStripMenuItem_Click(object sender, EventArgs e)
        {
            if (textBox1.SelectedText != string.Empty)
                Clipboard.SetData(DataFormats.Text, textBox1.SelectedText);
        }

        private void pasteToolStripMenuItem_Click(object sender, EventArgs e)
        {
            int position = textBox1.SelectionStart;
            textBox1.Text = textBox1.Text.Insert(position, Clipboard.GetText());
        }
```

```
    }
}
```





// картинки через PictureBox



ВЫПОЛНЕНИЕ РАБОТЫ:

1. **На оценку «хорошо»** Написать приложение windows form, выполняющее занесение в буфер обмена текста и получающее из буфера обмена находящиеся в нем данные в разных окнах. Сделать вывод о способах обмена данными между приложениями.
2. **На оценку «отлично»** Написать приложение windows form, выполняющее занесение в буфер обмена текста и графического изображения через OpenPictureDialog, и получающее из буфера обмена находящиеся в нем данные в разных окнах. Сделать вывод о способах обмена данными между приложениями.

ИИ:

https://docs.microsoft.com/ru-ru/dotnet/api/system.windows.clipboard?view=netcore-3.1
http://plssite.ru/csharp/clipboard_article_part_1.html
https://www.demo2s.com/csharp/csharp-clipboard-getimage.html

Практическое занятие №6
РАСПРЕДЕЛЕНИЕ ПАМЯТИ ПК

Цель работы: знакомство с функциями Win32 и структурами данных, используемыми для управления памятью.

Теоретические сведения:

Класс **Win32_OperatingSystem** WMI представляет сведения об операционной системе на основе Windows, установленной на компьютере.

FreePhysicalMemory

Тип данных: **uint64** Тип доступа: только для чтения
Квалификаторы: **единицы** ("килобайты")
Число в килобайтах физической памяти, неиспользуемой и доступной в настоящее время.

FreeSpaceInPagingFiles

Тип данных: **uint64** Тип доступа: только для чтения

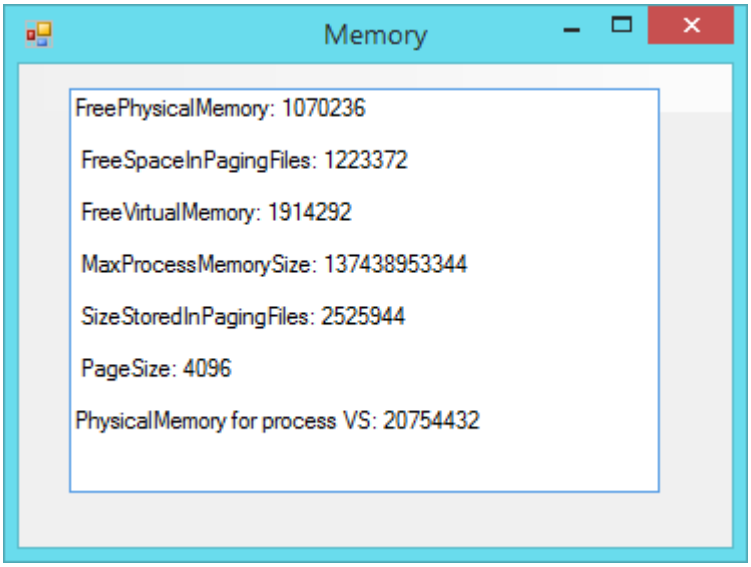
Квалификаторы: **MappingStrings** (MIF. DMTF| Системная память Параметры|001.4"), **единицы** (килобайты)
Число в килобайтах, которое может быть сопоставлено с файлами подкачки операционной системы без переключения других страниц.

FreeVirtualMemory
Тип данных: **uint64** Тип доступа: только для чтения
Квалификаторы: **единицы** ("килобайты")
Число в килобайтах виртуальной памяти, неиспользуемой и доступной в настоящее время.

MaxProcessMemorySize
Тип данных: **uint64** Тип доступа: только для чтения
Квалификаторы: **единицы** ("килобайты")
Максимальное число в килобайтах памяти, которую можно выделить процессу. Для операционных систем без виртуальной памяти это значение обычно равно общему объему физической памяти за исключением памяти, используемой BIOS и операционной системой. Для некоторых операционных систем это значение может быть бесконечностью, в этом случае следует ввести 0 (ноль). В других случаях это значение может быть константой, например 2G или 4G.

SizeStoredInPagingFiles
Тип данных: **uint64** Тип доступа: только для чтения
Квалификаторы: **MappingStrings** (MIF. DMTF| Системная память Параметры|001.3"), **единицы** ("килобайты")
Общее количество килобайтов, которые могут храниться в файлах подкачки операционной системы — 0 (ноль) указывает на отсутствие файлов подкачки. Имейте в виду, что это число не представляет фактический физический размер файла подкачки на диске.

```
using System.Management.Instrumentation;
using System.Management;
...

private void Form1_Load(object sender, EventArgs e)
 {
        ObjectQuery mem = new ObjectQuery("SELECT * FROM Win32_OperatingSystem");
        ManagementObjectSearcher searcher = new ManagementObjectSearcher(mem);
        ManagementObjectCollection results = searcher.Get();
        foreach (var result in results)
        {
           textBox1.Text = Convert.ToString("FreePhysicalMemory: " + result["FreePhysicalMemory"]);
           textBox1.Text += Convert.ToString("\r\n\r\n FreeSpaceInPagingFiles: "+
result["FreeSpaceInPagingFiles"]);
...
        }
     }
...
textBox1.Text += Convert.ToString("\r\n\r\n PageSize: " + Environment.SystemPageSize);
```

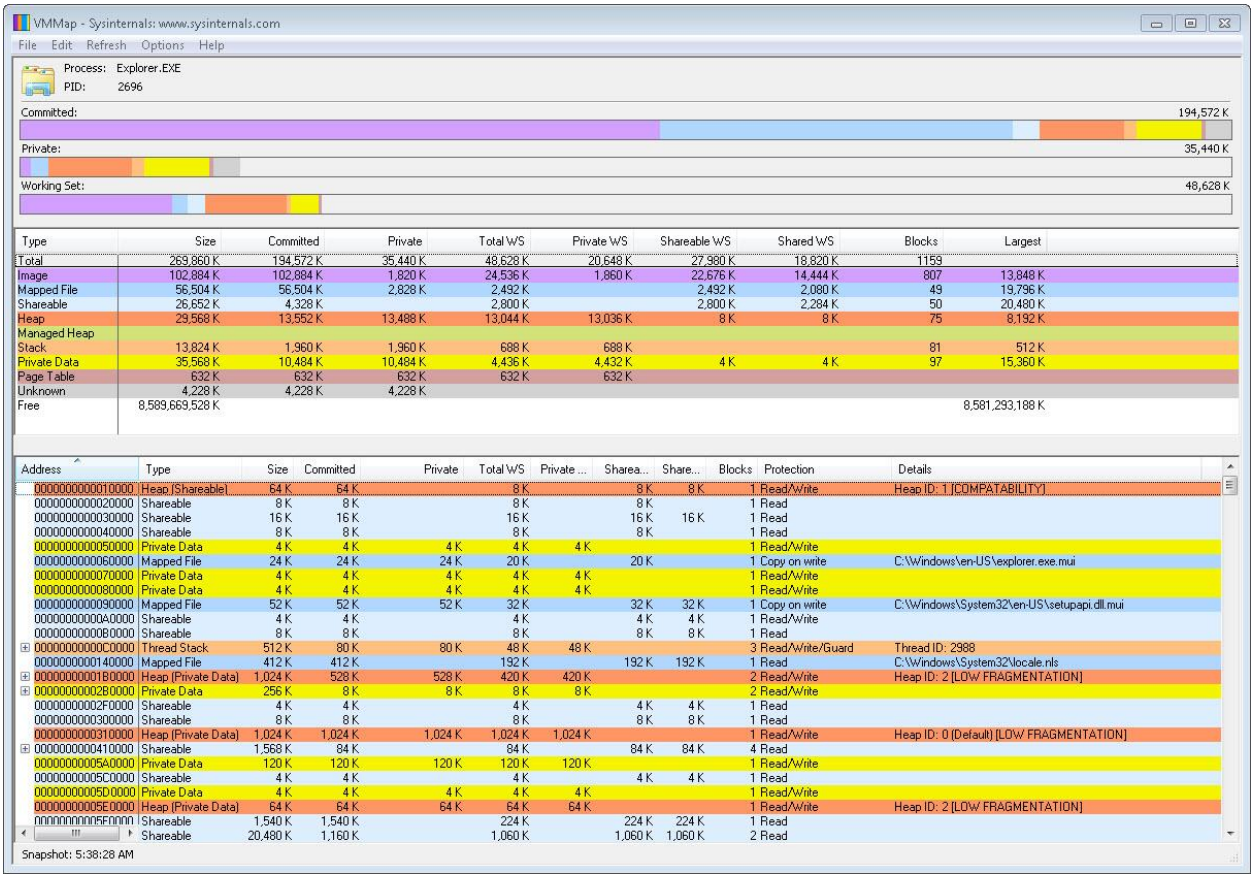


VMMap — это утилита для анализа виртуальной и физической памяти процессов. Он показывает разбивку типов виртуальной памяти, выделенных процессом, а также объем физической памяти (рабочий набор), назначенный операционной системой этим типам. Помимо графического представления использования памяти, VMMap также показывает сводную информацию и подробную карту памяти процесса. Мощные возможности фильтрации и обновления позволяют определить источники использования памяти процессом и стоимость памяти для функций приложения.

Помимо гибких представлений для анализа текущих процессов, VMMap поддерживает экспорт данных в нескольких формах, включая собственный формат, который сохраняет всю информацию, чтобы вы могли загрузить ее обратно. Он также включает параметры командной строки, которые позволяют создавать сценарии.

VMMap — идеальный инструмент для разработчиков, желающих понять и оптимизировать использование ресурсов памяти своего приложения.

https://docs.microsoft.com/en-us/sysinternals/downloads/vmmap



ВЫПОЛНЕНИЕ РАБОТЫ:

Создать приложение windows form, осуществляющее

1) вывод на экран системной информации, в том числе о всей памяти, занятой и свободной, размере страницы памяти.
2) скачать утилиту vmmap, установить и сравнить данные вашей программы и утилиты от microsoft, предоставить скриншоты, сделать выводы.

ЛИТЕРАТУРА (рекомендуемая)

4. Лекция управление памятью
5. *Таненбаум Э., Бос Х.* Современные операционные системы. 4-е изд. — СПб.: Питер, 2015. — 1120 с.: ил. — (Серия «Классика computer science»). ISBN 978-5-496-01395-6— Текст : электронный // Образовательные ресурсы Интернета [сайт]. — URL: https://www.ss-20.ru/
6. https://docs.microsoft.com/en-us/sysinternals/downloads/vmmap
7. https://learn.microsoft.com/ru-ru/windows/win32/cimwin32prov/win32-operatingsystem
8. https://learn.microsoft.com/ru-ru/visualstudio/profiling/memory-usage?view=vs-2022

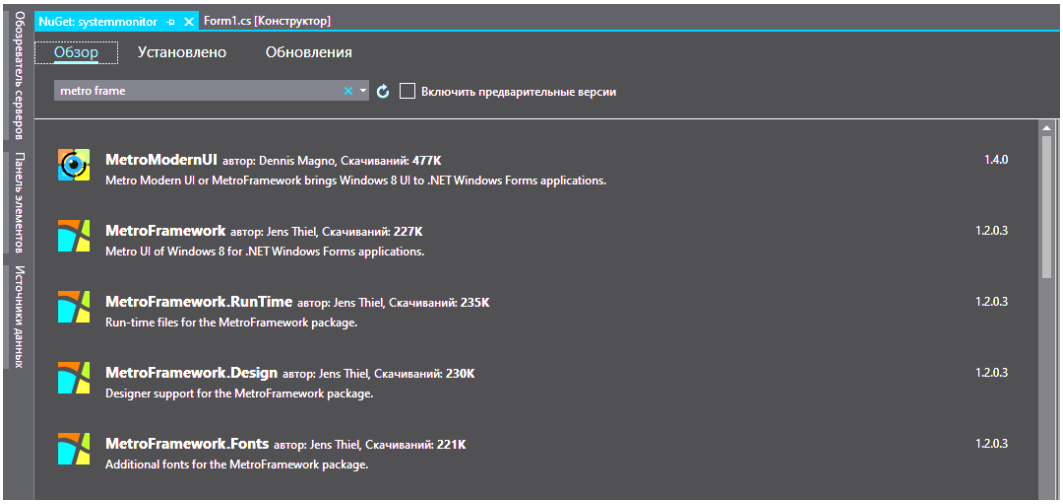
Практическое занятие №7

Монитор ресурсов памяти

Цель работы: знакомство с функциями Win32 и структурами данных, используемыми для мониторинга памяти.

Ход работы:

В обозревателе решений ПКМ – Управление пакетами NuGet, выбрать вкладку ОБЗОР.
В строке поиска ввести metro framework



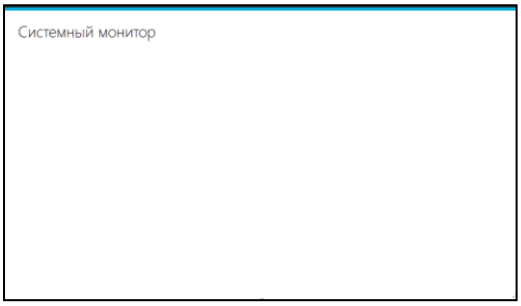
Установить пакет MetroFramework. Пересобрать решение.
На форме нажать ПКМ, выбрать перейти к коду.
Заменить код формы в заголовке

```
  public partial class Form1 : Form
```

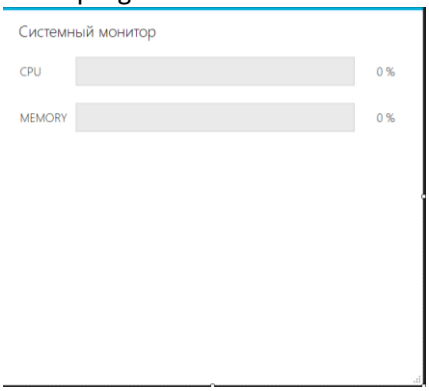
на

```
  public partial class Form1 : MetroFramework.Forms.MetroForm
```

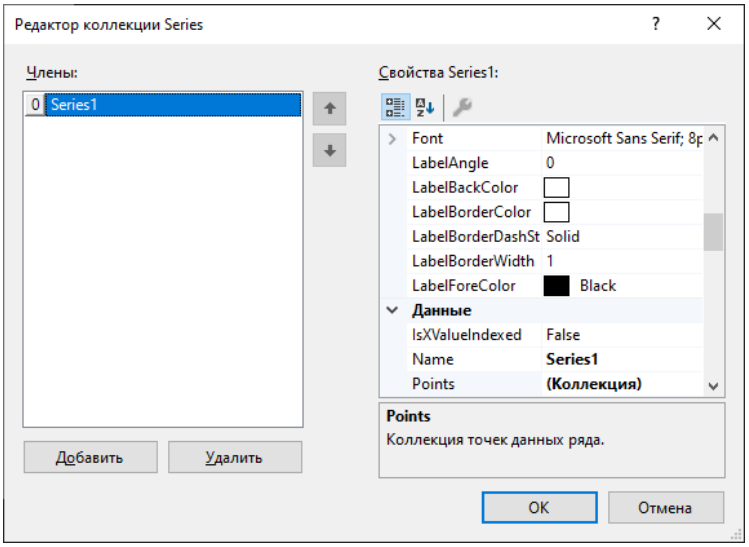
Пересобрать решение.



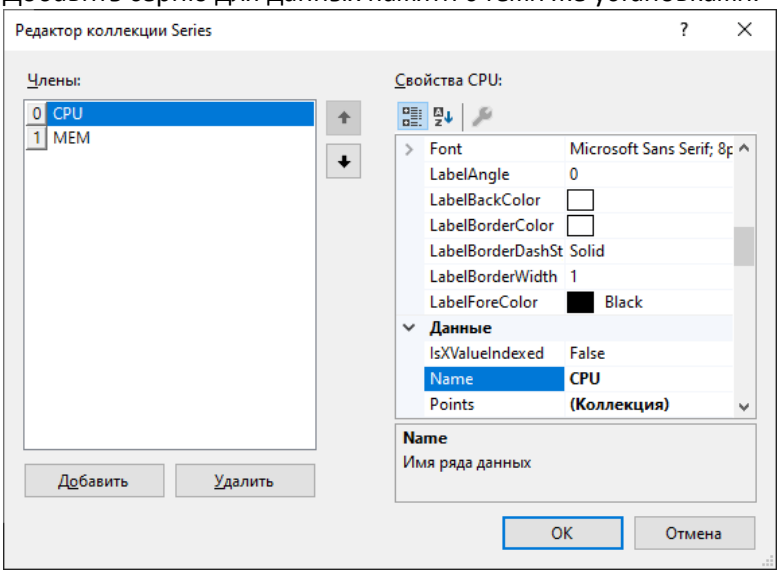
Установить на форму 4 метки metrolabel и переименовать их в CPU и MEMORY и две 0%. Установить два metroprogreebar.

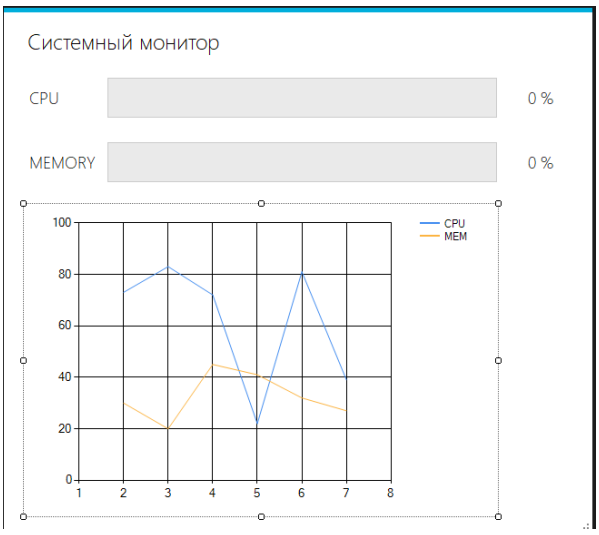


Добавьте компонент chart – Диаграмму. Выделите ее и в свойствах нажмите на три точки у свойства Series. Редакторе коллекций Series переименуйте Series1 в CPU и свойство Charttype установить в Line.



Добавить серию для данных памяти с теми же установками.





Внести код в обработчики событий load и timer внести код

```
ссылка: 1
private void timer1_Tick(object sender, EventArgs e)
{
    var pCPU = new PerformanceCounter("Processor","% Processor Time","_Total");
    var pMEM = new PerformanceCounter("Memory", "% Committed Bytes in Use");
    float fcpu = pCPU.NextValue();
    float fmem = pMEM.NextValue();
    metroProgressBar1.Value = (int)fcpu;
    metroProgressBar2.Value = (int)fmem;
    metroLabel3.Text = string.Format("{0:0.00}%", fcpu);
    metroLabel4.Text = string.Format("{0:0.00}%", fmem);
    chart1.Series["CPU"].Points.AddY(fcpu);
    chart1.Series["MEM"].Points.AddY(fmem);
}

ссылка: 1
private void Form1_Load(object sender, EventArgs e)
{
    timer1.Start();
}
```

Если вы все сделали правильно, ваш системный монитор покажет загрузку процессора и памяти в реальном режиме времени.

ВЫПОЛНЕНИЕ РАБОТЫ:

Создать приложение windows form, осуществляющее мониторинг загрузки процессора и памяти, предоставить скриншоты, сделать выводы.

ЛИТЕРАТУРА (рекомендуемая)

1. Лекция управление памятью
2. *Таненбаум Э., Бос Х.* Современные операционные системы. 4-е изд. — СПб.: Питер, 2015. — 1120 с.: ил. — (Серия «Классика computer science»). ISBN 978-5-496-01395-6— Текст : электронный // Образовательные ресурсы Интернета [сайт]. — URL: https://www.ss-20.ru/
3. https://learn.microsoft.com/ru-ru/dotnet/api/system.diagnostics.performancecounter?view=dotnet-plat-ext-6.0
4. https://learn.microsoft.com/ru-ru/dotnet/api/system.windows.forms.progressbar?view=windowsdesktop-6.0
5. https://learn.microsoft.com/en-us/dotnet/api/system.windows.forms.datavisualization.charting.chart?view=netframework-4.8

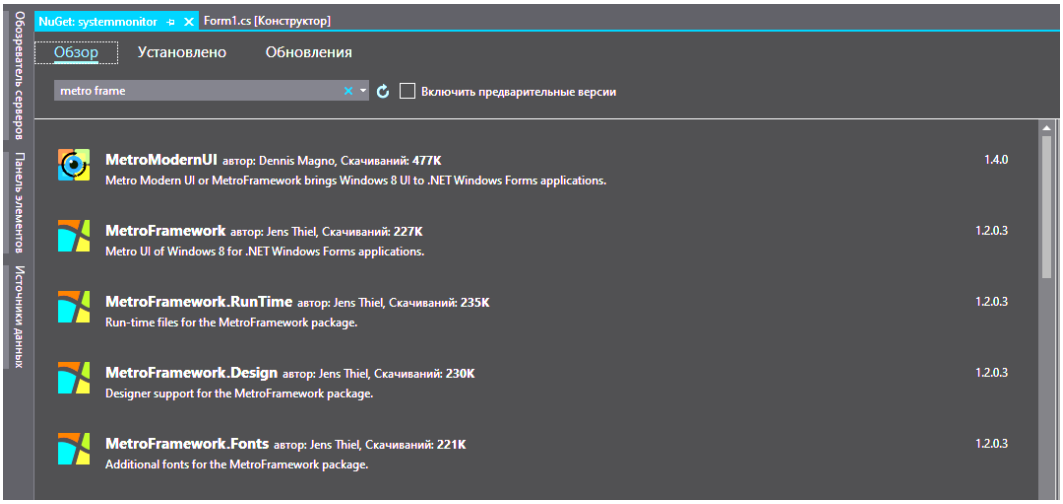
Практическое занятие №7

Монитор ресурсов памяти

Цель работы: знакомство с функциями Win32 и структурами данных, используемыми для мониторинга памяти.

Ход работы:

В обозревателе решений ПКМ – Управление пакетами NuGet, выбрать вкладку ОБЗОР.
В строке поиска ввести metro framework



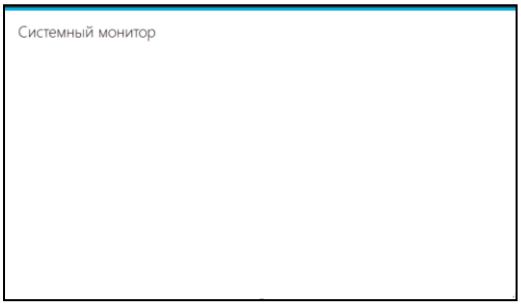
Установить пакет MetroFramework. Пересобрать решение.
На форме нажать ПКМ, выбрать перейти к коду.
Заменить код формы в заголовке

```
  public partial class Form1 : Form
```

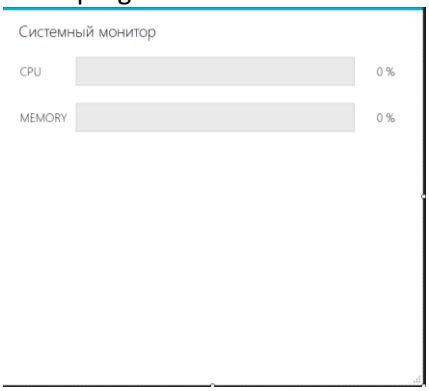
на

```
  public partial class Form1 : MetroFramework.Forms.MetroForm
```

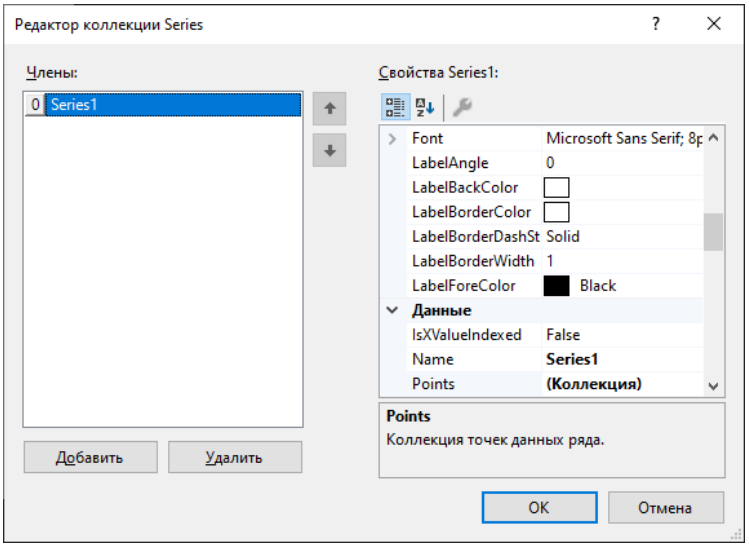
Пересобрать решение.



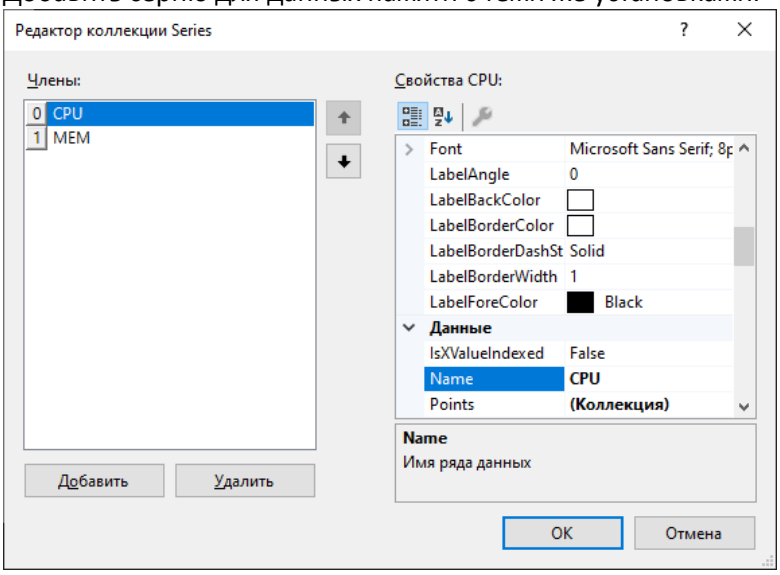
Установить на форму 4 метки metrolabel и переименовать их в CPU и MEMORY и две 0%. Установить два metroprogreebar.

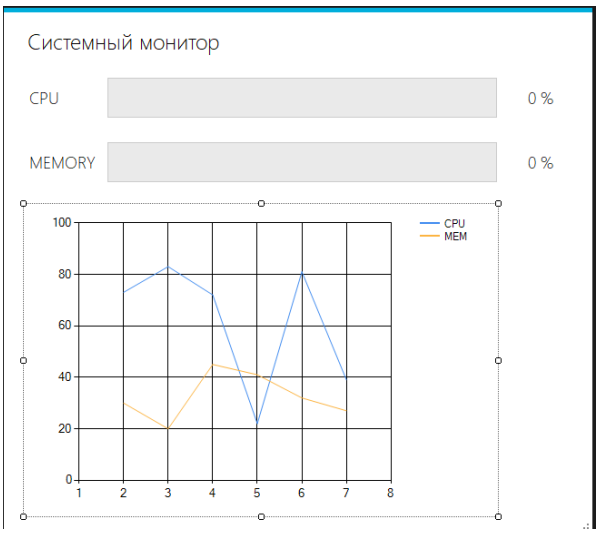


Добавьте компонент chart – Диаграмму. Выделите ее и в свойствах нажмите на три точки у свойства Series. Редакторе коллекций Series переименуйте Series1 в CPU и свойство Charttype установить в Line.



Добавить серию для данных памяти с теми же установками.





Внести код в обработчики событий load и timer внести код

```
ссылка: 1
private void timer1_Tick(object sender, EventArgs e)
{
    var pCPU = new PerformanceCounter("Processor","% Processor Time","_Total");
    var pMEM = new PerformanceCounter("Memory", "% Committed Bytes in Use");
    float fcpu = pCPU.NextValue();
    float fmem = pMEM.NextValue();
    metroProgressBar1.Value = (int)fcpu;
    metroProgressBar2.Value = (int)fmem;
    metroLabel3.Text = string.Format("{0:0.00}%", fcpu);
    metroLabel4.Text = string.Format("{0:0.00}%", fmem);
    chart1.Series["CPU"].Points.AddY(fcpu);
    chart1.Series["MEM"].Points.AddY(fmem);
}

ссылка: 1
private void Form1_Load(object sender, EventArgs e)
{
    timer1.Start();
}
```

Если вы все сделали правильно, ваш системный монитор покажет загрузку процессора и памяти в реальном режиме времени.

ВЫПОЛНЕНИЕ РАБОТЫ:

Создать приложение windows form, осуществляющее мониторинг загрузки процессора и памяти, предоставить скриншоты, сделать выводы.

ЛИТЕРАТУРА (рекомендуемая)

1. Лекция управление памятью
2. *Таненбаум Э., Бос Х.* Современные операционные системы. 4-е изд. — СПб.: Питер, 2015. — 1120 с.: ил. — (Серия «Классика computer science»). ISBN 978-5-496-01395-6— Текст : электронный // Образовательные ресурсы Интернета [сайт]. — URL: https://www.ss-20.ru/
3. https://learn.microsoft.com/ru-ru/dotnet/api/system.diagnostics.performancecounter?view=dotnet-plat-ext-6.0
4. https://learn.microsoft.com/ru-ru/dotnet/api/system.windows.forms.progressbar?view=windowsdesktop-6.0
5. https://learn.microsoft.com/en-us/dotnet/api/system.windows.forms.datavisualization.charting.chart?view=netframework-4.8

Практическое занятие №9
НАСТРОЙКА ДРАЙВЕРОВ УСТРОЙСТВ

Цель работы: ознакомление со способами установки и обновления драйверов, формирование навыков настройки драйверов, закрепление знаний о подсистеме управления вводом-выводом.

Теоретические сведения:

Драйвер – программное обеспечение, с помощью которого операционная система получает доступ к аппаратному обеспечению некоторого устройства. Обычно он создается производителем устройства и поставляется вместе с этим устройством. Поскольку для каждой операционной системы нужны собственные драйверы, производитель устройства обычно поставляет драйверы для нескольких наиболее популярных операционных систем.

Каждый драйвер устройства обычно управляет одним типом устройства или как максимум одним классом родственных устройств. Например, драйвер SCSI-диска обычно может управлять несколькими SCSI-дисками разного объема и разной скорости и, может быть, приводом Blu-ray-диска со SCSI-интерфейсом. А вот мыши и джойстики настолько отличаются друг от друга, что для них, как правило, требуются разные драйверы. Тем не менее технически вполне возможно создание одного драйвера

устройства, управляющего несколькими разнородными устройствами. Это в большинстве случаев не самая лучшая идея.

Но иногда совершенно разные устройства основаны на одной и той же базовой технологии. Наверное, наиболее известным примером может послужить USB — технология последовательной шины, которая не зря называется универсальной. К USB-устройствам относятся диски, карты памяти, фотоаппараты, мыши, клавиатуры, мини-вентиляторы, беспроводные сетевые карты, роботы, считыватели кредитных карт, аккумуляторные бритвы, измельчители бумаг, сканеры штрих-кодов и портативные термометры. Все они используют USB, хотя занимаются совершенно разными вещами.

Характерная особенность заключается в том, что из USB-драйверов обычно формируется стек, подобный TCP/IP-стеку в сетях. Внизу, как правило, в оборудовании, находится уровень USB-ссылок (последовательного ввода-вывода), обрабатывающий все, что касается аппаратуры, например сигнализацию и декодирование потоков сигналов в USB-пакеты. Он используется для более высоких уровней, работающих с пакетами и функциями USB, общими для большинства устройств. И наконец, наверху над всем этим находятся высокоуровневые API-интерфейсы, например интерфейсы для накопителей, камер и т. д. Таким образом, у нас по-прежнему имеются отдельные драйверы устройств, несмотря на то что ими совместно используются части стека протокола.

Чтобы получить доступ к аппаратной части устройства, то есть к регистрам контроллера, драйвер устройства, как правило, должен быть частью ядра операционной системы, по крайней мере в существующих на сегодняшний день архитектурах. Но вообще-то можно создавать и драйверы, работающие в пространстве пользователя, используя при этом системные вызовы для чтения и записи регистров устройств. Такое решение позволит изолировать ядро от драйверов и драйверы друг от друга, устранив при этом основной источник системных сбоев — «сырые» драйверы, тем или иным образом мешающие работе ядра. Несомненно, это хороший выход из положения при создании высоконадежных систем. Примером системы, в которой драйверы устройств запускаются в качестве процессов пользователя, может послужить MINIX 3 (www.minix3.org).

Но поскольку большинство других операционных систем для настольных компьютеров предполагают запуск драйверов в ядре, рассматриваться будет именно такая модель.

Так как разработчики любых операционных систем знают, что эти фрагменты программного кода (драйверы), созданные другими разработчиками, будут устанавливаться в их систему, им нужна такая архитектура, которая позволит подобную установку. А это значит, что должна быть вполне определенная модель того, чем занимается драйвер и как он взаимодействует со всей операционной системой. Как показано на рис. 1, драйверы устройств обычно размещаются ниже остальных компонентов операционной системы.

Рисунок 1 – Логическое позиционирование драйверов устройств

Обычно операционная система относит драйверы к одной из немногочисленных категорий. Самые распространенные категории — это драйверы блочных устройств, к ним относятся драйверы дисков, содержащих множество блоков данных, к которым можно обращаться независимо от всех остальных блоков, и драйверы символьных устройств, к которым относятся драйверы клавиатур и принтеров — устройств, которые генерируют или воспринимают поток символов.

Для многих операционных систем определены стандартный интерфейс, который должен поддерживаться всеми драйверами блочных устройств, и еще один стандартный интерфейс, который должен поддерживаться всеми драйверами символьных устройств. Эти интерфейсы состоят из нескольких процедур, которые могут вызываться всей остальной операционной системой для обращения к драйверу. К этим процедурам относятся, например, процедуры чтения блока (в блочных устройствах) или записи строки символов (в символьных устройствах).

В некоторых системах операционная система представляет собой единую программу в двоичных кодах, в которой содержатся все необходимые ей скомпилированные драйверы. Такая схема долгие годы была нормой для систем семейства UNIX, поскольку они работали в компьютерных центрах, где устройства ввода-вывода менялись очень редко. При добавлении нового устройства системный администратор просто перекомпилировал ядро с новым драйвером для создания нового двоичного кода. С наступлением эры персональных компьютеров с несметным количеством устройств ввода-вывода эта модель уже не работает. Лишь немногие пользователи способны перекомпилировать или перекомпоновать ядро, даже если у них будут исходные коды или объектные модули, что случается довольно редко. Вместо этого операционные системы, начиная с MS-DOS, перешли к модели, в которой драйверы стали динамически загружаться в систему в процессе работы. Управление загрузкой драйверов ведется в разных системах по-разному.

На драйвер устройства возлагается несколько функций. Наиболее очевидные из них — восприятие абстрактных запросов на чтение и запись от независимого от конкретных устройств программного обеспечения, находящегося выше них по уровню, и отслеживание порядка их выполнения. Но на них возлагается также ряд других функций.

Например, драйвер должен при необходимости инициализировать устройство. Он может понадобиться также для управления энергопотреблением устройства и регистрации событий.

Многие драйверы устройств имеют сходную общую структуру. Типичный драйвер начинает свою работу с проверки приемлемости входных параметров. Если они неприемлемы, возвращается сообщение об ошибке. Если с параметрами все в порядке, может понадобиться перевод абстрактных

понятий в конкретные. Для драйвера диска это может означать преобразование обычного номера блока в номера головки, дорожки, сектора и цилиндра, относящихся к геометрии диска. Затем драйвер может проверить, используется ли устройство в данный момент. Если оно используется, запрос будет поставлен в очередь для последующей обработки.

Если устройство простаивает, проверяется состояние аппаратуры, чтобы определить, может ли запрос быть обработан. Перед началом передачи данных может понадобиться включить устройство или запустить его двигатель. Как только устройство включится и будет готово к работе, им можно будет управлять.

Управление устройством означает выдачу в его адрес последовательности команд. Именно драйвер определяет последовательность команд в зависимости от того, что должно быть сделано. После того как драйвер поймет, какие команды он собирается выдать, он начнет записывать их в регистры контроллера устройства. После записи каждой команды в контроллер может потребоваться проверка того, принял ли контроллер команду и готов ли к приему следующей команды. Эта последовательность повторяется до тех пор, пока не будут выданы все команды. Некоторым контроллерам можно указывать на связанный список команд (в памяти) и предписывать самостоятельное чтение и обработку этих команд без дальнейшей помощи со стороны операционной системы.

После того как команды были выданы, может сложиться одна из двух ситуаций. В большинстве случаев драйвер должен ждать, пока контроллер не сделает в его интересах какую-нибудь работу, поэтому он самоблокируется до тех пор, пока не поступит прерывание на его разблокировку. Но в других случаях операция завершается без задержки и драйверу не нужно блокироваться. В качестве примера последней ситуации можно привести прокрутку экрана в символьном режиме, требующую лишь записи нескольких байтов в регистры контроллера. Для этого не нужно никаких механических перемещений, поэтому вся операция может быть завершена за несколько наносекунд.

В первом случае заблокированный драйвер будет активизирован прерыванием. Во втором случае он никогда не будет переходить в неактивное состояние. В любом случае по завершении операции драйвер должен провести проверку на отсутствие ошибок. Если все в порядке, драйвер может получить данные (например, только что считанный блок) для передачи программному обеспечению, не зависящему от применяемого устройства. И наконец, он возвращает вызывавшей его программе определенную информацию о состоянии устройства, наличии или отсутствии ошибок. Если в очереди были какие-нибудь другие запросы, то теперь один из них может быть выбран и запущен на выполнение. Если запросов в очереди не было, драйвер блокируется в ожидании следующего запроса.

Эта упрощенная модель дает весьма приблизительное понятие о реальной работе. На самом деле программный код из-за множества различных факторов куда сложнее. Прежде всего, устройство ввода-вывода может завершить свою работу и прервать работу драйвера. Прерывание может стать причиной запуска драйвера. Между прочим, оно может вызвать запуск текущего драйвера. К примеру, при обработке сетевым драйвером входящего пакета может прибыть еще один пакет. Следовательно, драйверы должны быть реентерабельными, то есть работающий драйвер еще до завершения первого вызова должен ожидать повторного вызова.

В системе, допускающей горячее подключение, устройство может быть добавлено или удалено во время работы компьютера. В результате, пока драйвер занят чтением с какого-нибудь устройства, система может ему сообщить, что пользователь внезапно удалил это устройство из системы. Драйверу придется не только прервать текущую передачу данных, не повредив при этом каких-либо структур данных ядра, но и умудриться элегантно удалить из системы все отложенные запросы для только что удаленного устройства и сообщить плохие новости пославшим их программам. Более того, неожиданное добавление новых устройств может заставить ядро перераспределить ресурсы (например, линии запроса прерываний), забирая у драйвера старые и предоставляя вместо них новые.

Драйверам не разрешается делать системные вызовы, но часто возникает необходимость взаимодействовать с остальной частью ядра. Обычно им разрешается вызывать определенные процедуры ядра. Например, для использования в качестве буферов выделенных аппаратуре страниц памяти драйверы вызывают процедуры, которые занимаются их выделением и освобождением. Другие полезные вызовы нужны для управления таймерами, контроллером DMA, контроллером прерываний и т. д.

ВЫПОЛНЕНИЕ РАБОТЫ:

1) Ознакомьтесь с теоретическими сведениями к работе.
2) Вызовите диспетчер устройств. Опишите какими способами можно выполнить эту команду (не менее трех). В каких режимах могут отображаться устройства/ресурсы ПК в диспетчере устройств, вставьте скриншоты с пояснениями в отчет.
3) Внесите в отчет сведения о драйверах дисков, клавиатуры и мыши вашего компьютера (скриншоты).
4) Внесите в отчёт информацию по каждому способу установки и обновления драйверов, перечисленных в списке:
   a) установка и обновление драйверов с диска
   b) через диспетчер устройств
   c) Поиск драйвера по ID устройства
   d) через «Центр обновления windows»
   e) с сайта производителя ПК
   f) с сайта производителя комплектующих.
5) Изучите ПО для установки и обновления драйверов и внесите в отчет описание, достоинства и недостатки указанных программ для установки и обновления драйверов:
   a) DriverHub
   b) Slimware Driverupdate
   c) Driver Genius
   d) Driver Booster
   e) Snappy Driver Installer
   f) DriverPack Solution
6) Сделать выводы о проделанной работе.

ЛИТЕРАТУРА (рекомендуемая)

   a) Конспект/презентации лекций
   b) Таненбаум Э., Бос Х. Современные операционные системы. 4-е изд. — СПб.: Питер, 2015. — 1120 с. 380-411.

Практическое занятие №10

РАБОТА В КОМАНДНОЙ ОБОЛОЧКЕ ОПЕРАЦИОННОЙ СИСТЕМЫ LINUX.

Цель работы: практическое знакомство с операционной системой Linux, создание учетной записи в менеджере пользователей, получение справки о командах терминала.

.

Краткие теоретические сведения

Файловая система Linux

Перед тем как начать работать с **Linux**, прочитайте некоторые сведения о ее файловой структуре, которая в корне отличается от файловой структуры **Windows**.

Чтобы было удобнее пользоваться файлами в различных каталогах, в операционной системе **Linux** введено понятие — корневой каталог, который обозначается: «/». Это та точка, от которой отсчитывается расстояние до любого файла. Сколько бы не было установлено винчестеров в компьютере, корневой каталог всегда один. Это, следует отметить, самое существенное отличие **Linux** от **Windows** для пользователя, так как в **Windows** в каждом дисковом разделе существует свой корневой каталог, к которому обращаются как **C:**, **D:**, **E:** и т. д.

Но если корневой каталог **/** является стандартным для всех **Unix**-систем, то названия и назначение подкаталогов меняется от дистрибутива к дистрибутиву. Особенно большая разница между различными ветвями дистрибутивов **Linux**. Далее описано назначение наиболее важных подкаталогов для дистрибутивов **Linux**, разработанных на основе **Mint**:
/ — корневой каталог, от которого идет отчет пути ко всем остальным каталогам и файлам.

/bin — этот каталог является первым, в котором ищутся файлы команд, введенных с клавиатуры. В нем содержатся основные системные программы (исполняемые файлы, файлы в двоичном коде), такие как общие команды управления файлами, оболочки командной строки и традиционные **Linux**-программы.

/boot — здесь расположены файлы, в которых содержится образ ядра. При запуске компьютера отсюда загружается операционная система.

/dev — каталог предназначен для файлов устройств. То есть в каталоге находятся специальные файлы, посредством которых операционная система взаимодействует с системными и физическими устройствами.

/etc — в каталоге расположены основные конфигурационные файлы операционной системы, а также подкаталоги с конфигурационными файлами прикладных программ.

/home — в этом каталоге располагаются все домашние каталоги пользователей, в которые устанавливаются личные конфигурационные файлы и программы.

/lib — в каталоге собраны общесистемные библиотеки,

/mnt — здесь размещаются каталоги, в которых монтируются внешние файловые системы, в основном это гибкий диск, CD-ROM, различные USB-устройства.

/root — домашний каталог администратора (root, суперпользователя).

/sbin — каталог для системных файлов программ, применяемых для администрирования системы.

/tmp — каталог для временных файлов. Файлы в этом каталоге могут автоматически удаляться по прошествии некоторого времени.

/usr — в этом каталоге создаются подкаталоги для пользовательских программ, их библиотек и документации.

/var — каталог для рабочих и журнальных файлов, которые часто меняются, например, здесь можно найти протоколы системных событий, в них создаются временные файлы во время работы принтера и т. д.

В остальном организация файловой структуры традиционна и привычна для пользователей. Например, персональные файлы пользователя располагаются в каталоге **home**:

/home/dima/имя_файла_пользователя_dima

Имена файлов в Linux

Также к существенным отличиям **Linux** от **Windows** относится принцип именования файлов и названия внутренних и внешних устройств.

Имя файла может содержать до 255 любых символов, кроме символа "/" и нулевого кода (**00Hex**). Расширения имени файла идут через точку в виде трех символов, но на самом деле, они не обязательны.

Кроме того, так как с файлами работает оболочка операционной системы, для которой ряд символов имеет специальное значение, то *не рекомендуется* использовать в именах файлов следующие символы:

! @ # $ & ~ % * ( ) {] { } < " \ : ; > < ‘ "пробел"

Принципы именования устройств

В **Linux** все устройства, с точки зрения пользовательских программ, — это файлы. К дисководу, принтеру, сканеру любая прикладная программа обращается как к обычному файлу.

Такой принцип привести к общему знаменателю все особенности конструкции самых разнообразных внешних устройств. Значительно упрощается процедура интегрирования в операционную систему новых периферийных устройств, которые регулярно выходят из лабораторий и внедряются в производство.

Так как в **Linux** все манипуляции с внешними устройствами сводятся к операциям с обычными файлами, то для пользователя архитектура компьютера представляется в виде файловой системы. В одном каталоге находятся устройства ввода-вывода, а в других пользовательские файлы и программы.

Имена внешних устройств

Так как любое внешнее устройство отождествляется в **Linux** с файлом, то в имени этого файла имеется цепочка символов, характеризующая его особенности. Причем принято включать в имя устройства название каталога **/dev**. Например, дисковод **A:** называется **/dev/fd0**, дисковод **B:** называется **/dev/fd1**. Но так как в компьютере теперь обычно устанавливается всего один дисковод гибких дисков, то обычно используется универсальное имя: **/dev/floppy**.

Когда модем подключен к **COM**-порту компьютера и конфигурация операционной системы настроена автоматически, то программы при обращении к модему используют имя **/dev/modem**. К модему можно обратиться и по имени порта, например, порт **COM1** называется **/dev/ttyS0** и т.д.

Имена винчестеров жестко связаны с точками подключения к **IDE**-интерфейсу и типом устройства: первый (*master*) винчестер на первом канале **IDE** называется **/dev/hda**, второй (*slave*) −**/dev/hdb**, первый (*master*) винчестер на втором канале **IDE** называется **/dev/hdc**, второй (*slave*) −**/dev/hdd**.

При использовании интерфейса **SCSI** первый **SCSI**-диск называется **/dev/sda**, второй −**/dev/sdb**.

Учитывая подобный подход к именованию винчестеров, следует всегда помнить, что переустановка винчестера на другой шлейф или изменение его статуса, например, переустановка винчестера с *Master* на *Slave* приведет к тому, что потребуется изменить информацию в конфигурационных файлах.

В **Linux** разделы винчестера, которые по умолчанию являются в **Windows**, например, дисками **C:** и **D:**, имеют другие имена. Их составляют из имени винчестера и номера раздела на винчестере. Например, если винчестер подключен на первый **IDE**-канал как *Master*, то имена **C:** и **D:** будут звучать так: диск **C** − **/dev/hda1**, диск **D** − **/dev/hda5**. Имя **hdal** говорит о том, что раздел является основным. Таких основных разделов может быть только *четыре*, поэтому возможны только следующие варианты: **/dev/hdal**, **/dev/hda2**, **/dev/hda3** и **/dev/hda4.**

Логический диск **D:** в расширенном разделе получил имя **hda5**. Соответственно диск **E:** будет иметь имя **/dev/hda6**, а **F:** − **/dev/hda7**.

Если винчестер подключен на вторичный канал, то имена разделов будут: диск **C** − **/dev/hdc1**; диск **D** − **/dev/hdc5**.

Если винчестер включен в режиме *Slave*, то имена разделов изменятся: диск **C** − **/dev/hdb1** или**/dev/hdd1**, диск **D** − **/dev/hdb5** или **/dev/hdd5**.

В тех случаях, когда в компьютер установлен винчестер с интерфейсом **serial ATA**, имена разделов на нем будут именоваться, например, **/dev/sda1** и **dev/sda2**

Монтирование файловой системы

Так как в **Linux** точка «/» «одна на всех», поэтому возникает вопрос: где располагаются другие файловые системы, разделы, винчестеры, носители и т. д. Тем более что все устройства представляют собой файлы.

Поэтому в операционной системе **Linux** есть понятие ***монтирование файловой системы***. С общей точки зрения любое устройство, дисковый накопитель, прежде чем к нему можно обратиться (читать или записать данные), должно быть смонтировано в какой-либо точке файловой системы. Причем точка монтирования может быть произвольной. Для обеспечения совместимости различных дистрибутивов **Linux** друг с другом и для упрощения взаимодействия пользователей в файловой структуре всегда существует каталог:

/mnt

В этом каталоге по традиции монтируются все внешние накопители в виде отдельных каталогов: дисковод гибких дисков − **/mnt/floppy**, привод компакт-дисков − **/mnt/cdrom**.

Здесь же стараются монтировать и разделы **Windows**, хотя это необязательно, например, диск **C:**− **/mnt/windows**.

X Window

Для вывода графики на экран монитора в **Linux** используется система **X Window**, известная как **X11** или просто **X**. Это самостоятельный программный продукт, который разработан группой компьютерных компаний совместно с *Массачусетским* технологическим институтом, предназначенный для **Unix**-систем.

Основной изюминкой **X Window** является использование архитектуры клиент-сервер. Причем на компьютере пользователя запускается **X-сервер**, управляющий непосредственно оборудованием ввода (клавиатура, мышь) и вывода (монитор). Любые другие программы (клиенты), когда запущена программа **X Window**, получают и выводят информацию только с ее помощью.

Чтобы *вручную* запустить **X Window**, следует в командной строке набрать

startx

Для того чтобы прекратить работу **X Window**, например, у вас по каким-то причинам зависла программа, используйте **Ctrl+Alt+Backspace**. Это как бы горячая перезагрузка компьютера, которая вызывается в **Windows** тремя популярными клавишами, но в данном случае операционная система не перезагружается, а только прекращается работа **X-сервера**.

X Window поддерживает еще две горячие комбинации клавиш: **Ctrl+Alt+Plus** и **Ctrl+Alt+Minus**. Клавиши **Plus** и **Minus** находятся на цифровой клавиатуре. С помощью этих комбинаций клавиш можно оперативно менять разрешение экрана с **1024x768** на **800x600**.

Виртуальные консоли

Пусть вы работаете с каким-то документом, а в этот момент надо дать соседу поработать с другой программой. В таком случае достаточно переключиться на другую виртуальную консоль и войти в систему под другим именем. После этого можно спокойно отойти от компьютера и при этом быть абсолютно спокойным за сохранность данных в программе.

Традиционно в **Linux** допускается создание до **63** виртуальных консолей, хотя обычно используется **2**−**3**, изредка **5**−**7**. Причем каждая виртуальная консоль может принадлежать одному пользователю, но можно войти в систему и под разными именами. Так как физически к персональному компьютеру подключена одна клавиатура и один монитор, то, чтобы переключаться между виртуальными консолями, используются комбинации клавиш от **Alt+Fl** до **Alt+F12.**

При работе в графической среде **X Window** комбинации **Alt+Fx** блокируются. Причем почти всегда для работы в графической среде используется виртуальная консоль **7** (**Alt+F7**). Чтобы переключаться на другие виртуальные консоли, используются комбинации клавиш от **Ctrl+Alt+F1** до **Ctrl+Alt+F12**. Для возврата в графическую среду используются клавиши **Alt+F7**.

Пользователи в Linux. Вход в систему

Вход в систему будет осуществлен под пользователем **root** пароль (преподавателя подзовите). Если вы завершите этот сеанс и попробуете войти заново, то пароль необходимо будет ввести.

Вход в систему начинается с запроса вашего имени и пароля, так как только после авторизации в системе можно выполнять какие-либо действия:

Login: *имя пользователя*

Password: *пароль пользователя*

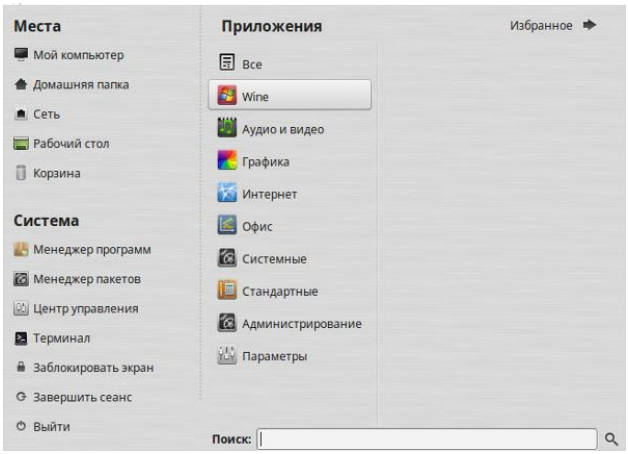
В том случае, когда пользователю разрешено войти в систему, на экране появится приглашение, например такое:

[my@dog my] $

В квадратных скобках дается подсказка пользователю, а символ "**$**" говорит о том, что в систему вошел обычный пользователь. Справа от этого символа командная строка, где можно вводить команды. Окончание ввода команды указывается пользователем нажатием на клавишу **Enter**.

Разберем информацию в квадратных скобках: до символа "**@**" указывается *имя пользователя*, а после — *имя компьютера*. После пробела указывается *имя каталога*, в котором находится в данный момент пользователь (путь к каталогу не выводится). Заметим, что домашний каталог пользователя ту находится в каталоге **/home/my**.

Для входа в систему с правами суперпользователя (администратора) используется имя **root**. Оно одинаково для всех дистрибутивов **Linux**. Процедура регистрации в системе для администратора точно такая же, как и для остальных пользователей:



Login: root

Password: *пароль администратора*

Информация в квадратных скобках формируется так же, но в качестве домашнего каталога для администратора выступает каталог **/root**.

Виртуальная консоль. Терминал. Обозреватель файлов

Итак, вы запустили операционную систему **Linux**. Перед тем как изучить возможности графической среды, просмотрите состав меню.

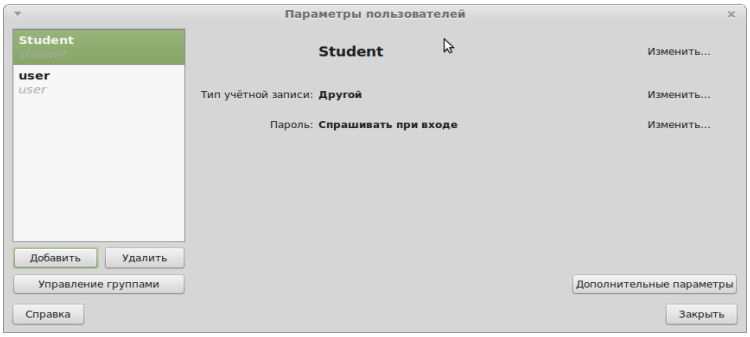
Аналог консоли Windows - Система /Терминал. Аналог Проводника — Приложения/Системные/Caja-просмотр файловой системы в менеджере файлов.

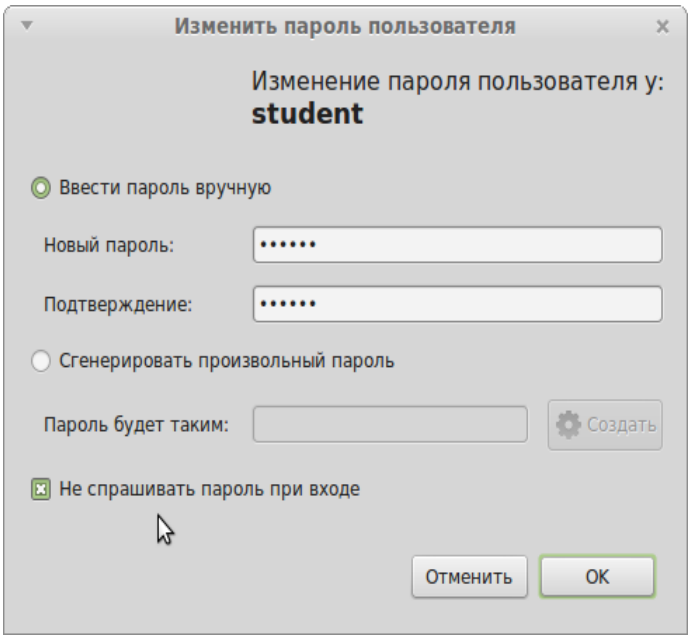
Просмотрите содержимое меню Приложения: системные, стандартные, администрирование, параметры.

ВЫПОЛНЕНИЕ РАБОТЫ:

В текстовом редакторе LibreOffice writer сформируйте отчет по работе и в конце занятия предъявите его преподавателю.

1. Создайте с помощью **Менеджера пользователей** Приложения/Администрирование/ Пользователи и группы нового пользователя Student с паролем 315315. Зайдите под пользователем Student.
2. Запустите Terminal. Эта программа предназначена для выполнения функций командной строки ОС Linux. Здесь в интерактивном режиме вы можете выполнять любые команды и программы, зарегистрированные в системе.





3. Выполните команду history
   history -c. **********
   Объясните разницу.
4. Запомните формат команд в ОС Linux:
имя команды [аргументы] [параметры] [метасимволы]
5. Выполните команду man
   man ls
   man pwd
6. Ознакомьтесь с командами Linux, запишите назначение команд в отчет в виде таблицы:

cal
cat
cd
date
echo
free
ln
ls
mkdir
ps
pwd
rm
top
wc
who

7. Выполните 4 любые команды из таблицы с различными с различными параметрами.
8. Выведите календарь на месяц и год вашего рождения, на 2022 год.
9. Создайте файл summa.cpp.

```
#include <iostream>
#include <conio.h>

using namespace std;
int a, b, c;
int main() {
  cin >> a >> b;
  c = a + b;
  cout << c;
  getch();
  return 0;
}
```

10. Выведите файл summa.cpp последовательно и с нумерацией строк с учетом пустых строк.
11. Выведите текущую дату и время.
12. Перейдите в родительскую директорию, вернитесь в домашнюю, перейдите в предыдущую, вернитесь в домашнюю.
13. Объясните разницу в выполнении команд
echo shell
echo $shell
echo $SHELL
echo we learn linux
echo -e “we \nlearn \nlinux”
14. Выведите общее количество свободной памяти в мегабайтах.
15. Определите в каком каталоге вы находитесь.
16. Выведите полный пронумерованный список директорий домашнего каталога.
17. Выведите список процессов, выполняющихся в терминале.
18. Выполните команду bash. Как изменился список процессов? В чем различия команд who и ps?
19. Войдите в свой домашний каталог (команда cd ~). Здесь хранятся ваши пользовательские файлы и настройки программ, которые вы используете.
20. Создайте следующую структуру каталогов и файлов:

- в домашнем каталоге создайте каталог inform;
- перейдите в каталог inform и создайте в нем каталог pz10;
- внутри каталога pz10 создайте каталог catalog1, файл file1 (например, используя команду echo), каталог catalog2. Перейдите в каталог catalog2;
- внутри каталога catalog2 создайте файлы file3 и file4, каталог catalog3;
- внутри каталога catalog3 создайте файл file5, жесткую ссылку на файл file1;
- создайте в каталоге pz12 символическую ссылку s_link на файл file5.

21. Откройте мой компьютер.
22. Посмотрите структуру созданных вами каталогов и содержимое файлов.
23. Убедитесь, что вы находитесь в домашнем каталоге, выведите содержимое каталога catalog2.
24. Удалите файл file4.
25. Выход из системы exit – окончание сеанса пользователя.

Вопросы

1.Понятие «ядро операционной системы».
2.Основные папки файловой системы **Linux**.
3.Принцип именования файлов и устройств в **Linux**.

4.Монтирование системы.
5.Особенность виртуальной консоли.
6.Перечислите группы меню в **Linux**.
7.Аналоги **программ** в Windows и в **Linux**.
8. Команды Терминала **Linux**

ОП.02 Архитектура аппаратных средств

Тематика практических занятий

1. Анализ конфигурации вычислительной машины.
2. Утилиты обслуживания жестких магнитных дисков и оптических дисков.
3. Периферийные устройства компьютера и интерфейсы их подключения. Устройство клавиатуры и мыши, настройка параметров работы клавиатуры и мыши.
4. Конструкция, подключение и инсталляция матричного принтера.Конструкция, подключение и инсталляция струйного принтера. Конструкция,подключение и инсталляция лазерного принтера. Конструкция, подключение и инсталляция графического планшета.

Практическая работа №1
Анализ конфигурации вычислительной машины

Задание 1. Заполните таблицу (в таблицу следует заносить только реальные данные по конфигурации Вашего компьютера, в случае отсутствия какого-либо устройства ставится прочерк).

п/п	**Наименование параметра**	**Значение параметра**
1.	Тип и модель монитора	
2.	Форм-фактор корпуса системного блока	
3.	Клавиатура, интерфейс подключения	
4.	Вид манипулятора "мыши", интерфейс ее подключения	
5.	Интерфейсы подключения периферийных устройствна задней панели системного блока (наименование и количество)	
6.	Интерфейсы подключения периферийных устройствна лицевой панели системного блока (наименование и количество)	
7.	Процессор, модель и тактовая частота	
8.	Объем оперативной памяти	
9.	Тип модема и сетевого интерфейса	
10.	Наименование и скорость привода для чтения оптических дисков	
11.	Модель и объем памяти накопителя на жестких магнитных дисках	
12.	Видеоадаптер, модель и объем видеопамяти	
13.	Модель звукового адаптера	
14.	Версия операционной системы	

15.	Другие периферийные устройства (принтер, сканер и т.д.)	

Практическая работа №2. Утилиты обслуживания жестких магнитных дисков и оптических дисков

Задание 1. Работа с разными режимами представления информации в панели

1. На правой панели перейдите к диску D:\, а на левой к диску C:\, используя кнопки дисков или окно выбора дисков.
2. Установите для левой панели Подробный режим отображения информации, используя соответствующую команду меню Вид или кнопку на панели инструментов. Что изменилось в окне ТС?
3. Активизируйте правую панель и выполните команду Вид-Дерево(или кнопка).
4. Отсортируйте выводимый на экран список файлов и каталогов диска D:\,выполняя следующие функции пункта меню Вид:

4.1. По имени – по именам в алфавитном порядке.

4.2. По типу – по расширению в алфавитном порядке.

4.3. По времени – по времени создания или последнего обновления.Обязательно убедитесь, что файлы отсортированы по дате.

4.4. По размеру – по размеру в байтах

4.5. Без сортировки – в том порядке, как они записаны на диске.

5. Выведите на левую панель файлы с расширением .doc, для чеговыполните команду Вид – Фильтр....
6. Выполните команду Вид – Все файлы

Задание 2. Работа с каталогами в ТС

1. Создайте в своем каталоге D:\студенты\Номер_группы\фамилия\лр следующее дерево каталогов: Фамилия Имя Отчество, например d:\ Студенты\номер_группы\фамилия\лр2\Петров\Пётр\петрович Для этого используйте клавишу F7
2. Переименуйте каталог Имя в мое_Имя (т.е. каталог Пётр в мое_Имя).Для этого используйте клавишу F6
3. Удалите каталог ОтчествоДля этого можно использовать клавиши F8, Delete

Задание 3. Копирование, перемещение и удаление файлов в ТС

1. Создайте с помощью MS Word и сохраните в каталогеD:\ Студенты\номер_группы\фамилия\лр три текстовых файла с именами Файл1,Файл2 и Файл3 следующего содержания:

1.1. Файл1 – Фамилия и Имя студента.

1.2. Файл2 – Фамилия и номер группы.

1.3. Файл3 – Фамилия, название факультета и курс.

2. Скопируйте Файл1 в каталог мое_Имя. Для этого:

2.1. На одной из панелей сделайте видимым файл Документ1.

2.2. На соседней панели сделайте текущим каталог моеИмя.

2.3. Нажмите F5 или перетащите мышью Файл1 на соседнюю панель.

2.4. В появившемся диалоговом окне нажмите кнопку ОК.

3. Скопируйте Файл1 из каталога мое_Имя в каталог Фамилия поддругим именем (Док1).

4. Переместите свой Файл2 в каталог Фамилия. Для этого:

4.1. На одной из панелей сделайте видимым Файл2.

4.2. На соседней панели сделайте текущим каталог Фамилия.

4.3. Выберите мышью Файл2 и нажмите клавишу F6 или кнопку F6Перемещ.

4.4. В появившемся диалоговом окне нажмите кнопку ОК.

5. Удалите файл Документ1 из каталога мое_Имя.Для этого можно использовать клавиши F8, Delete.

Задание 4. Поиск файлов в ТС

1.Найдите на диске D:\ файлы с расширением .xls. Для этого:

1.1. Выполните команду Инструменты – Поиск файлов. (Можно нажатьсоответствующую кнопку на панели инструментов).

1.2. В области Искать файлы введите шаблон *.xls.

1.3. Нажмите кнопку Диски и выберите [-D-].

1.4. Установите флажок Поиск в архивах.

1.5. Нажмите кнопку Начать поиск.

1.6. Перейдите к любому файлу с расширением .xls, выделив его инажав кнопку Перейти к файлу (или используя двойной щелчок мыши) .

2. Отыщите файлы с расширением .doc, в которых содержитсяопределенный текст (уточните текст у преподавателя), например, слово «Информационные технологии». Для этого укажите шаблон *. doc в области Искать файлы. Нажмите кнопку Диски и выберите [-d-]. Включите переключатель с текстом и введите слово. Проанализируйте результат поиска. В списке Результаты поиска выберите файл (желательно из своей папки) и откройте его. После просмотра закройте файл без сохранения.

3. Найдите файлы на локальных дисках, созданные за последнюю неделю. Для этого в окне Поиск файлов перейдите на закладку Дополнительно, установите флажок Не старше чем и выберите нужный интервал времени.

Задание 5. Архивация, разархивация файлов в ТС

1. Скопируйте файл

D:\Студенты\номер_группы\фамилия\Лаб1.doc всвою папку. Упакуйте файл Лаб1.doc из своей папки, выполнив командуФайл-Упаковать... (Можно нажать соответствующую кнопку на панели инструментов). Архив поместите в свою папку. В качестве архиватора выберите ZIP.

2. Загрузите упакованный файл Лаб1.doc, для чего войдите в архив, сделав двойной щелчок мышью на нем, а затем двойной щелчок на Лаб1.doc. В диалоговом окне Свойства архива нажмите кнопку Распаковать и выполнить. После этого загрузится файл Лаб1.doc. Просмотрите файл и закройте окно.

3. Распакуйте файл Лаб1.zip в каталог Фамилия:

3.1. Выделите файл Лаб1.zip.

3.2. Убедитесь, что на соседней панели активным является каталогФамилия.

3.3. Выполните команду Файл – Распаковать..., и нажмите ОК.

3.4. Убедитесь, что файл распакован.

4. Добавьте в уже созданный архив Лаб1.zip еще один файл (из своейпапки Файл3)

4.1. На одной из панелей перейдите в архив Лаб1.zip.

4.2. На соседней панели сделайте видимым Файл3.

4.3. Перетащите файл в архив.

4.4. В диалоговом окне Упаковка файлов нажмите ОК.

Практическая работа №3
Периферийные устройства компьютера и интерфейсы их подключения. Устройство клавиатуры и мыши, настройка параметров работы клавиатуры и мыши

Задания:

1. Убедитесь в том, что компьютерная система обесточена (при необходимости, отключите систему от сети).
2. Разверните системный блок задней стенкой к себе.
3. По наличию или отсутствию разъемов USB установите форм-фактор материнской платы (при наличии разъемов USB – форм-фактор ATX, при их отсутствии -AT).
4. Установите местоположение и снимите характеристики следующих разъемов:
   - питания системного блока;
   - питания монитора;
   - сигнального кабеля монитора;
   - клавиатуры;
   - последовательных портов (два разъема);
   - параллельного порта;
   - других разъемов.

5. Убедитесь в том, что все разъемы, выведенные на заднюю стенку системного блока, не взаимозаменяемы, то есть каждое базовое устройство подключается одним единственным способом.

6. Изучите способ подключения мыши.

Мышь может подключаться к разъему последовательного порта или к специальному порту PS/2, имеющему разъем круглой формы. Последний способ является более современным и удобным. В этом случае мышь имеет собственный выделенный порт, что исключает возможность ее конфликта с другими устройствами, подключаемыми к последовательным портам. Последние модели могут подключаться к клавиатуре через разъем интерфейса USB.

7. Заполните таблицу:

Разъем	**Тип разъема**	**Количество контактов**	**Примечания**

8. Определить наличие основных устройств персональногокомпьютера.

9. Установите местоположение блока питания, выясните мощностьблока питания (указана на ярлыке).

10. Установите местоположение материнской платы.

11. Установите характер подключения материнской платы к блокупитания.

Для материнских плат в форм-факторе AT подключение питания выполняется двумя разъемами. Обратите внимание на расположение проводников черного цвета – оно важно для правильной стыковки разъемов.

12. Установите местоположение жесткого диска.

Установите местоположение его разъема питания. Проследите направление шлейфа проводников, связывающего жесткий диск с материнской платой. Обратите внимание на местоположение проводника, окрашенного в красный цвет (на жестком диске он должен быть расположен рядом с разъемом питания).

13. Установите местоположения дисководов гибких дисков и дисковода CD-ROM.

Проследите направление их шлейфов проводников и обратите внимание на положение проводника, окрашенного в красный цвет, относительно разъема питания.

14. Установите местоположение платы видеоадаптера.

Определите тип интерфейса платы видеоадаптера.

15. При наличии прочих дополнительных устройств выявите их назначение, опишите характерные особенности данных устройств (типы разъемов, тип интерфейса и др.).

16. Заполните таблицу:

Устройство	Характерные особенности	Куда и при помощи чего подключается

Практическая работа №4

Конструкция, подключение и инсталляция матричного принтера. Конструкция, подключение и инсталляция струйного принтера.

Задание 1. Схематично зарисовать структуру печатающей головки пьезоэлектрического струйного принтера с продольной деформацией активного элемента. Составить таблицу с указанием ее основных узлов и их назначения.

Задание 2. Схематично зарисовать структуру печатающей головки матричного принтера со сдвиговой деформацией активного элемента. Составить таблицу с указанием ее основных узлов и их назначения.

Задание 3. Сравнить две конструкции и выделить достоинства и недостатки.

Задание 4. Составить правила эксплуатации пьезоэлектрических струйных принтеров на основе особенностей их конструкции.

Конструкция, подключение и инсталляция лазерного принтера.

Задание 1. Составить сравнительную таблицу по видам источников излучения лазерных принтеров. Сравнение провести по следующим критериям: особенности конструкции, качество печати, скорость печати, надежность, стоимость.

Задание 2. Составить правила эксплуатации лазерных и LED- принтеров с учетом особенностей их конструкции.

Конструкция, подключение и инсталляция графического планшета

Контрольные вопросы:

1. Что такое графический планшет?
2. В чем отличие графического планшета от дигитайзера?
3. В чем принципиальное преимущество перьевого ввода в отличиеот мышки?
4. В чем заключается принцип действия дигитайзера?
5. На какие виды делятся графические планшеты по принципудействия?
6. В чем достоинства и недостатки индукционных планшетов?
7. Назовите основные фирмы-производители графическихпланшетов?
8. Какими преимуществами обладают графические планшеты?
9. Может ли графический планшет полностью стать заменителеммыши?
10. Перечислите основные характеристики графических планшетов.
11. Какие основные действия рекомендуется предпринять дляустранения неисправностей при работе с графическим планшетом.

Литература

1. Колдаев, В. Д. Архитектура ЭВМ: учеб. пособие для СПО – М.:ИД

ФОРУМ: НИЦ Инфра-М, 2018.

2. Архитектура ЭВМ : учебное пособие / Министерство образования и науки Российской Федерации, Федеральное государственное автономное образовательное учреждение высшего профессиональногообразования «Северо-Кавказский федеральный университет» ; авт.-сост. Е.В. Крахоткина, В.И. Терехин. - Ставрополь : СКФУ, 2015. - 80 с. - Библиогр.: с. 74-75. ; То же [Электронный ресурс]. - URL:http://biblioclub.ru/index.php?page=book&id=457862

3. Гуров, В.В. Архитектура и организация ЭВМ / В.В. Гуров, В.О. Чуканов. - 2-е изд., испр. - М. : Национальный Открытый Университет «ИНТУИТ», 2016. - 184 с. : ил., схем. - (Основы информационныхтехнологий). - Библиогр. в кн. - ISBN 5-9556-0040-X ; То же [Электронный ресурс]. - URL:http://biblioclub.ru/index.php?page=book&id=429021

ЛАБОРАТОРНАЯ РАБОТА №1.
ВЫПОЛНЕНИЕ ПРОСТЫХ ПРОГРАММ В BLUEJ

1.1. Цель работы

Освоить основы применения BlueJ – подготовку текста программы, компиляцию программы, исправление ошибок и просмотр результатов.

1.2. Постановка задачи

Разработать простейшую линейную программу, согласно варианту задания, научиться запускать программу и контролировать выводимый текст (результат работы программы).

1.3. Краткие теоретические сведения

Составление программы – процесс, направленный на то, чтобы заставить компьютер выполнить определенную работу. Указание компьютеру должно быть представлено как последовательность инструкций (план, программа). Проверка того, что данный набор инструкций выполняется правильно, состоит в сравнении желаемого результата с тем, который дает компьютер.

BlueJ – система для разработки программ на языке Java, созданная специально для обучения (http://www.bluej.org). Для решения отдельной задачи BlueJ создает **проект**. Проект – это каталог, в котором находится файлс текстом программы и ряд других файлов. Имя проекта задает программист.

Набор и коррекция программы производится при помощи встроенного текстового редактора. Файл с программой имеет расширение java. Компилятор (javac) переводит программу в команды виртуальной Java- машины (JVM) и сохраняет в файле с расширением class. Такой файл будет выполнять Java-машина, и программа сможет проделать работу, радикоторой она создана (рисунок 1.1).

Для составления программ студент должен владеть базовыми понятиями программирования, перечисленными ниже.

Идентификаторы используются в качестве имен классов, методов и переменных. Идентификатор может быть любой последовательностью букв верхнего и нижнего регистра (в том числе, кириллических), чисел или символов подчеркивания и знака коммерческого S ($). Он не должен начинаться с цифры, чтобы не вступать в конфликт с числовой константой. Язык Java чувствителен к регистру, поэтому идентификатор VALUE, например, отличается от идентификатора Value.

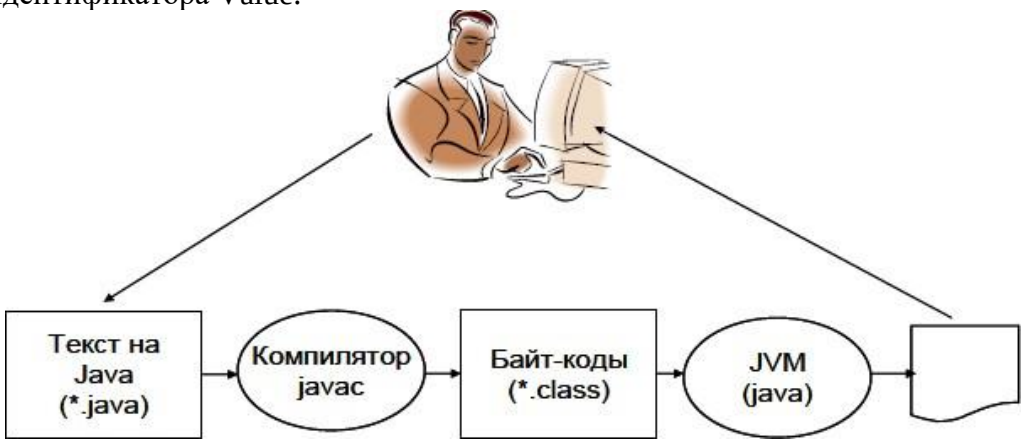


Рисунок 1.1. Подготовка и выполнение java-программы.

Переменная – это именованная область памяти, в которой программа может установить некоторое значение. Значение переменной может изменяться во время выполнения программы.

Переменная определяется комбинацией идентификатора (имени), типа и необязательного инициализатора. Переменная должна быть объявлена перед ее использованием.

Синтаксис объявления переменной:

type identifier [=value][, identifier [=value]…];

type – один из типов Java, имя класса или интерфейса,identifier –
имя переменной,
value – литерал (значение подходящего типа).Примеры:

int a, b, c;

int d = 3, e, f = 5;

Выражение – комбинация операндов и операций, задающая порядок вычисления некоторого значения (на основе приоритетов операций).

Операнд в простейшем случае является константой (литералом) или идентификатором (переменной). В общем случае каждый операнд выражения также представляет собой выражение, имеющее некоторое значение (в выражениях можно использовать скобки для изменения порядка действий).

Операция определяет действие, выполняемое над операндами. Возвращает некоторое значение.

Оператор − это некоторая конструкция, присущая данному конкретному языку, изменяющая состояние памяти компьютера, но ничего не возвращающая.

Замечание: *Не стоит путать два таких понятия как оператор и операция. Главное их отличие состоит в том, что операция возвращает значение, а оператор нет.*

Оператор присваивания предписывает вычисление выражения, находящегося правее знака (=) и присвоение полученного значения переменной, находящейся левее знака (=).

1.4. Порядок выполнения работы

Часть 1. Создание и выполнение Java –программы.

Основное окно системы BlueJ показано на рисунке 1.2. Окно содержит главное меню системы, управляющие кнопки, окно проекта, стенд объектов, окно команд (окно кода) и индикатор работы виртуальной машины Java.

Запуск BlueJ происходит после щелчка по иконке BlueJ. Дождитесь сообщения «Создается виртуальная машина…Готово». Выберите в меню BlueJ **Проект--Новый**. Введите имя проекта: «Фамилия_Группа_Lab1a».

Щелкните по кнопке **Новый класс**. Введите имя класса «Test1a».
Убедитесь, что тип класса – Класс.

Двойным щелчком по иконке класса Test1a в окне проекта откройтетекстовый редактор. При создании класса система BlueJ использует заготовку. Удалите из созданного класса весь код, нажав клавиши Ctrl-A(выделить все) и Delete.

Окно редактора с нужным текстом показано на рисунке1.3.

Введите текст, приведенный на рисунке 1.3. Проверьте правильность набора (оператор import (импортирование из библиотеки Java), точки с запятой после каждого оператора). Проверьте парность скобок { и }. Помните, что в Java большие и маленькие буквы различаются.

Правильность набора текста программы проверяется визуально и при помощи компилятора. Компилятор, анализируя программу по строкам сверху вниз, проверяет соответствие языку Java. При несоответствии выдается сообщение об обнаруженной ошибке. Подсвечивается строка программы, на которой компилятор прекратил работу. Обычно ошибка находится в этой строке или выше. Запомните: компилятор не проверяет правильность алгоритма, он проверяет только соответствие текста правилам языка Java! Поэтому возможно, что программа, которая компилируется без ошибок, будет работать неверно.

Рисунок 1.2 – Основное окно системы

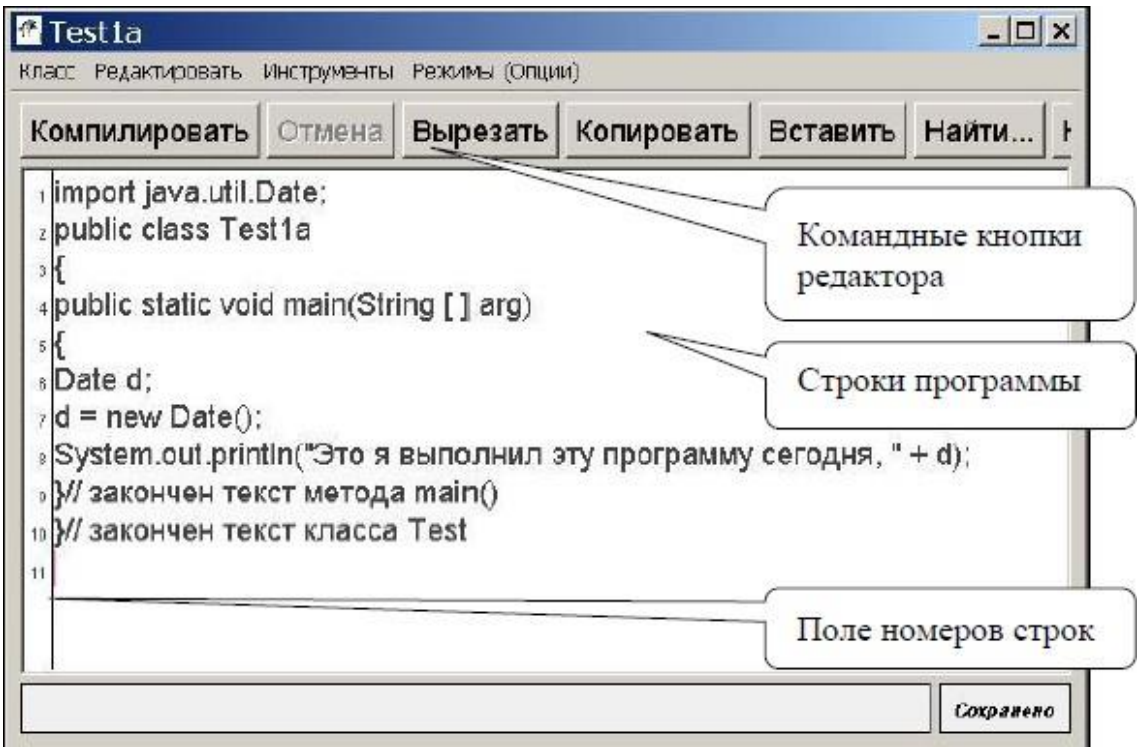


Рисунок 2.3 – Окно текстового редактора

Щелкните по кнопке **Компилировать**. Исправьте синтаксические ошибки, если они есть (ориентируйтесь на сообщения в нижней части окна). Если ошибок нет, можно выйти из редактора (кнопка Закрыть).

В окне проекта штриховка иконки класса показывает, что он не скомпилирован. Если какие-то классы проекта не скомпилированы, кнопкой Компилировать можно вызвать их компиляцию.

Щелкните правой кнопкой мышки по иконке класса, чтобы вызвать контекстное меню (рисунок 1.4). Выберите в меню метод**void main(String[ ] arg)**.

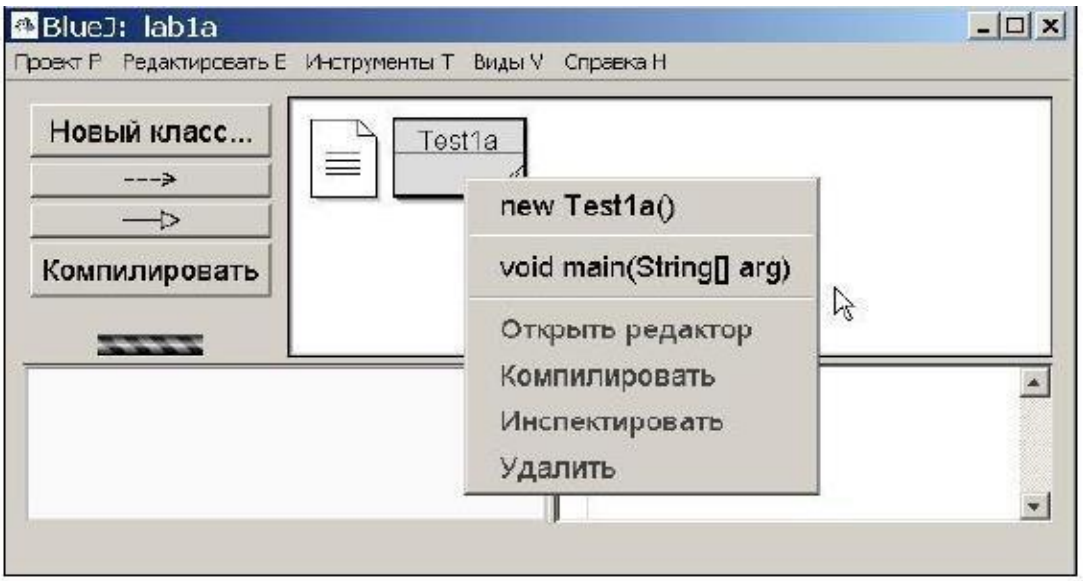


Рисунок 1.4. Окно проекта. Открыто контекстное меню класса Появится окно диалога, в котором можно задать аргумент arg метода

main. Но сейчас аргумент не нужен, просто щелкните по OK.

Запустится метод main, при этом индикатор работы VM начнет вращаться (вращение можно заметить только на медленных машинах).

Результат работы программы – текст, указанный в методе println и дата –будут выведены в окно терминала (рисунок 1.5). (Если окна терминала нет, в главном окне выберите Виды – Показать терминал).

Опции окна терминала позволяют выполнить ряд действий с содержимым терминала: очистить окно терминала, скопировать текст в буфер обмена или в файл, зафиксировать порядок вызова методов и т.д.

Часть 2. Внесение изменений в программу.

Измените строковый литерал (текст в двойных кавычках после println) так, чтобы выводились Ваши имя и фамилия. Обратите внимание на возможности, кото-рые предоставляет текстовый редактор. Кнопки редактора(**Отмена**, **Вырезать**, **Копировать**, **Вставить**, **Найти**) соответствуют типовым действиям по коррекции текстов. Эти же (и ряд других) действия скрыты под пунктами главного меню в верхней части окна. Многим командам редактора соответствуют "быстрые" комбинации клавиш. Например, Вырезать, Копировать и Вставить – это широко известные Ctrl+X, Ctrl+C, Ctrl+V, соответствующие одновременному нажатию клавиши Ctrl и буквенных клавиш X, C или V.

Откомпилируйте программу и проанализируйте результаты ее выполнения.

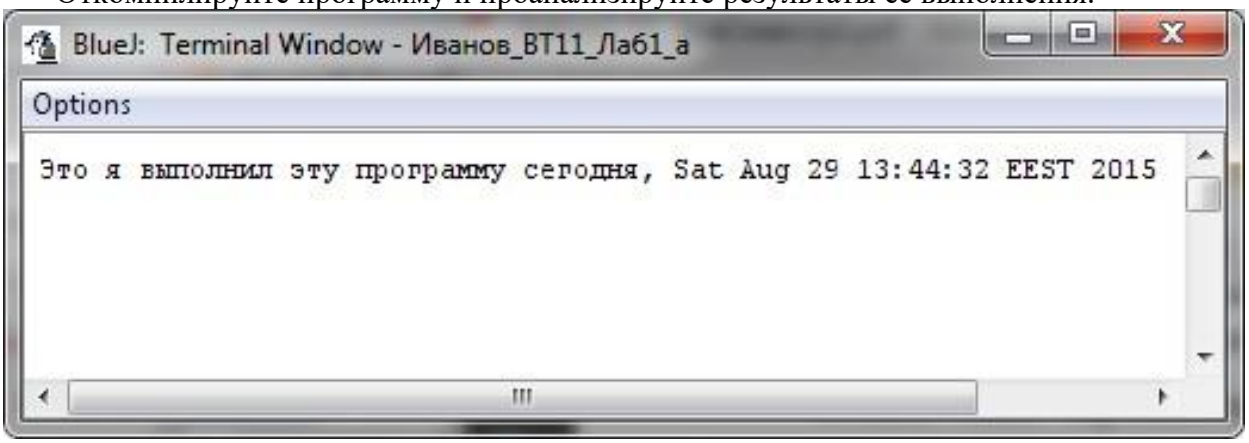


Рисунок 1.5. Окно терминала. Результат выполнения программы

Часть 3. Выполнение индивидуального задания.

Студент составляет программу (имя проекта: «Фамилия_Группа_Lab1b») согласно варианту задания (таблица 2.1) и оценивает результаты ее выполнения при помощи инструментальной системы BlueJ.

1.5. Варианты заданий

В качестве индивидуального задания на лабораторную работу предлагается разработать программу, выполняющую заданную операциюнад операндами целого типа (int). В программе должны быть определены соответствующие переменные для хранения операндов и результата. Программа должна осуществлять следующий вывод:

Программу выполнил:
Фамилия, имя, отчество студента,Шифр
группы,

Дата,
Вариант номер.
Название операнда1: значение операнда1, Название операнда2: значение операнда2, Название операции: значение результата. Проверил:
Фамилия, имя, отчество преподавателя.

Проверьте работу программы на нескольких тестовых примерах.

Варианты заданий представлены в таблице 2.1.

Таблица 2.1 – Варианты заданий

Номер варианта	Операнд 1	Операнд 2	Операция
1	Число студентов в группе	Число групп в потоке	Число студентов в потоке
2	Число недель для выполнения проекта	Число дней в неделе	Число дней для выполнения проекта
3	Число дней в неделе	Число выходных дней	Число рабочих дней
4	Число мест в кинотеатре	Число залов	Число мест в зале
5	Число книг в учебном абонементе	Число книг в читальном зале	Число книг в библиотеке
6	Число тетрадей	Число учебников	Общее число товаров в накладной
7	Число домов	Число квартир в доме	Число семей, получивших квартиры.
8	Число абитуриентов, подавших заявления на специальность	Число бюджетных мест	Конкурс – человекна место
9	Число студентов	Число преподавателей	Среднее число студентов на одного преподавателя
10	Стоимость билета	Число мест в автобусе	Выручка за рейс
11	Себестоимость товара	Наценка	Цена товара
12	Длина бассейна	Ширина бассейна	Площадь бассейна
13	Объем помещения	Высота потолка	Площадь помещения
14	Численность управляющего персонала	Численность рабочих	Всего сотрудниковна фабрике

15	Фонд заработной платы предприятия	Число сотрудников предприятия	Средняя зарплата
16	Вес товара	Вес тары	Общий вес
17	Длительность первой серии фильма в минутах	Длительность второй серии фильма в минутах	Длительность фильма в часах

Номер варианта	Операнд 1	Операнд 2	Операция
18	Число недель в семестре	Часов лекций по дисциплине в неделю	Всего часов лекций в семестре
19	Число студентов очной формы обучения	Число студентов заочной формы обучения	Всего студентов в университете.
20	Оклад	Отчисления	Зарплата на руки
21	Длина рулона ткани в метрах	Расход ткани на костюм в метрах	Число костюмов
22	Длительность серии фильма в минутах	Число серий	Длительность фильма в часах
23	Количество студентов, сдавших сессию на 4 и 5	Размер стипендии в рублях	Размер стипендиального фонда в рублях
24	Площадь стен помещения	Площадь рулона обоев	Число рулонов
25	Общая площадь участка	Площадь основания дома	Площадь приусадебного участка

1.6. Рекомендации по составлению отчета по лабораторной работе

Отчет должен содержать следующие разделы:

1) цель работы
2) постановка задачи;
3) тестовые примеры и результаты их обработки вручную;
4) текст Java-программы, заданной вариантом задания;
5) выводы (констатирует решение всех задач, описанных в разделе «Постановка задачи»).

В дополнение к общим требованиям к отчету (представленным в пункте 1), в разделе отчета «Выводы» приведите текст, составленный из ответов (рекомендуется ***своими словами***) на контрольные вопросы, приведенные в пункте 1.7. Ответ на каждый вопрос должен начинаться с нового абзаца

1.7. Контрольные вопросы

1) Опишите процесс создания проекта в инструментальной системеBlueJ.
2) Какие файлы входят в проект и каково их назначение?
3) Опишите назначение текстового редактора. Файлыс какимрасширением он создает?
4) Опишите назначение компилятора. Файлы с каким расширением онсоздает?
5) Что делает виртуальная Java-машина?
6) Для чего предназначен метод main?
7) Что делает встроенный метод System.out.println( );
8) Как получить значение текущей даты?
9) Как определить переменную целого типа в программе?
10) Для чего использован оператор присваивания в программе.
11) Что такое переменная? Что такое выражение? Какие операциицелочисленной арифметики выполнялись при вычислении выражения?

ЛАБОРАТОРНАЯ РАБОТА № 2. ОБРАБОТКА ДАННЫХ ПРОСТЫХ ТИПОВ. РАБОТА С ПАНЕЛЬЮ КОДА BLUEJ. ФОРМАТИРОВАННЫЙ ВЫВОД

3.1. Цель работы

Ознакомиться с простыми типами данных Java, научиться объявлять переменные и константы этих типов и выполнять операции над ними, научиться применять оператор присваивания для данных простых типов, научиться применять метод System.out.printf() – метод форматированного вывода − для вывода на экран значений различных типов, научиться применять окно кода в BlueJ.

3.2. Постановка задачи

1) Ввести заданные операторы в окно кода BlueJ и проанализировать полученные результаты.
2) Разработать программу, в которой используется метод System.out.printf() для вывода в окно терминала данных, предусмотренных вариантом задания.

3.3. Краткие теоретические сведения

3.3.1. Простые типы данных

Простые, или примитивные, типы данных - это встроенные в язык Java типы, аналогичные типам данных большинства языков программирования. Эти типы могут применяться самостоятельно или используются как составные части более сложных типов данных, которые создают программисты.

Java позволяет использовать 8 простых типов данных:

byte, short, int, long –целочисленные,
float, double –вещественные обычной и двойной точности,
boolean – булевские (логические, представляющие истину или ложь),
char – символьные, представляющие символы Unicode.

Диапазоны значений для простых типов данных приведены на рисунке 3.1.

Тип	Содержит	Значение по умолчанию	Ширина (биты)	Min и Max значения
boolean	true false	false	1	Не применимо
char	U-символ	\u0000	16	\u0000 до \uFFFF
byte	Целое	0	8	-128..127
short	Целое	0	16	-32 768..32 767
int	Целое	0	32	-2 147 483 648 .. 2 147 483 647
long	Целое	0	64	От -9 223 372 036 854 775 808 до 9 223 372 036 854 775 807
float	IEEE754 плав.тчк.	0.0	32	От +/-3.40282347E+38 до +/-1.40239846E-45
double	IEEE754 плав.тчк.	0.0	64	От +/-1.79769313486231570E+308 до +/-4.94065645841246544E-324

Рисунок 3.1 – диапазоны значений для примитивных типов данных

3.3.2. Базовые понятия программирования

Вспомним некоторые понятия программирования, упомянутые в предыдущей лабораторной работе.

Переменные и именованные константы.

Переменные – элементы, значение которых при выполнении программы можно изменить, но тип изменить нельзя.

Переменные имеют имена и значения. Имя переменной – это название области памяти, в которой хранится значение переменной. Чтобы ввестипеременную в употребление, ее необходимо объявить одним из двух способов:

1) тип имя;

2) тип имя = начальноеЗначение; (точка с запятой – это часть объявления!)

Объявление переменной – это приказ выделить для нее память.

Имена переменных пишут в стиле —camelStyle, начиная с маленькой буквы, а каждое следующее слово в имени – с большой буквы.

Именованные константы – это переменные, значение которых определяется «при рождении», после чего изменять их значения запрещено.

Форма задания константы с именем: **final тип ИМЯ = значение;**

Слово final указывает, что значение нельзя изменить. Принято имена констант записывать БОЛЬШИМИ БУКВАМИ.

Оператор присваивания

Оператор присваивания заменяет старое значение некоторой переменной новым. После замены старое значение теряется, т.е. использовать его нельзя.

Форма оператора: переменная = выражение;

Переменная является приемником данных, а выражение – источником. Выражение определяет новое значение переменной. Равенство левой и правой частей наступает только после выполнения оператора присваивания иимеет смысл равенства значений источника и приемника. Порядок выполнения оператора: сначала, без учета левой части, вычисляется выражение с текущими значениями переменных. Затем полученное значение выражения становится значением переменной-приемника. Если в выражении используется та же переменная, скажем, x, что и слева, при вычислениивыражения применяется имеющееся значение x, которое заменяется новым значением – значением выражения – после вычисления выражения. Например, пусть x=5.0. При выполнении оператора **x=x+1.2;** берется значение переменной **x**, равное 5, к нему добавляется число 1.2, и результат становится значением переменной **x**.

Язык Java требует, чтобы размер приемника был не меньше, чем размер результата. Когда это требование нарушается, возникает ошибка «Возможна потеря точности – possible loss of precision». Пример, при котором возникает такая ошибка: float x; x = 1.0; (размер слева – 32 бита, размер справа – 64 бита, т.к. 1.0 считается константой (литерал) типа double).

3.3.3. Использование окна кода BlueJ

Система BlueJ имеет специальное окно – панель (окно) кода (или окно команд) (рисунки 3.2, 3.3). Окно кода позволяет проверить фрагменты программы до их включения в проект. Действия в окне кода выполняются по строкам после нажатия Enter. Объявленные переменные запоминаются, т.е. повторно объявить переменную нельзя. Если окно кода не видно после запуска BlueJ, откройте проект, созданный ранее, и нажмите Виды – Показать панель кода. При необходимости вводимый в окно кода текст можно расположить в нескольких строках, при этом промежуточные строки заканчиваются нажатием Shift-Enter. Проверенные операторы можновставить в текст программы, пользуясь копированием (Ctrl-C) и вставкой(Ctrl-V).

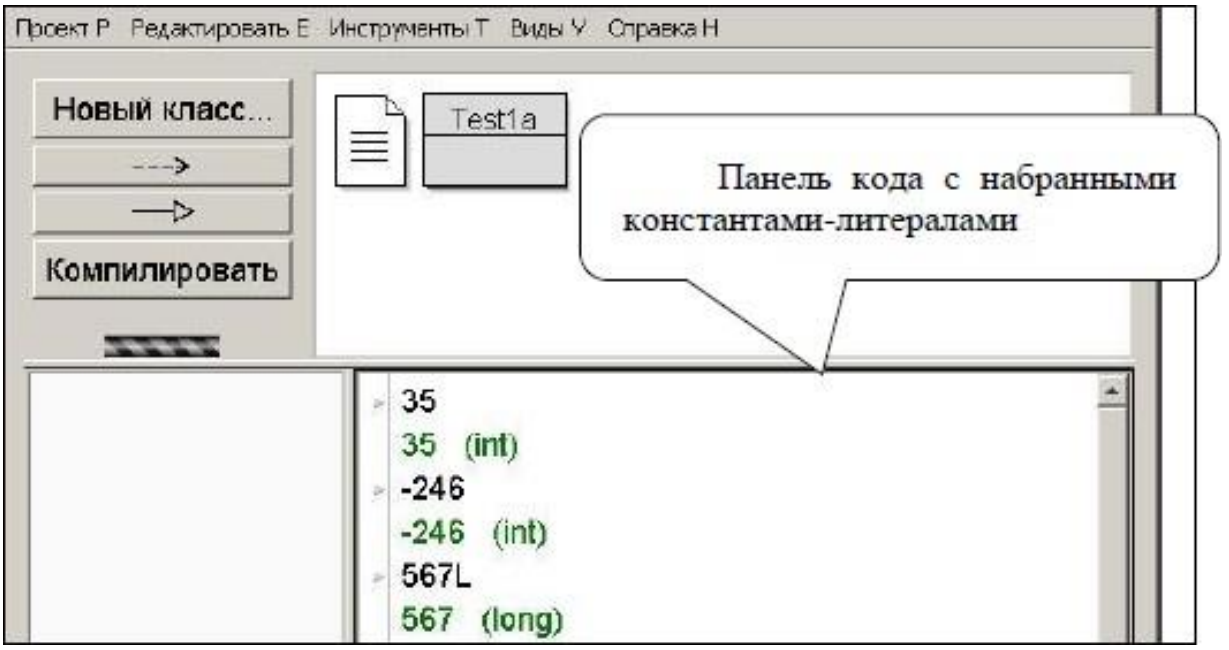


Рисунок 3.2. Окно кода. Примеры набора констант

Рисунок 3.3. Окно кода. Примеры набора выражений

3.3.4. Метод форматированного вывода System.out.printf()

В предыдущей работе использовался метод System.out.println(), который выводил в окно терминала символьную строку (объект типа String).

В этой работе рекомендуется применить метод форматированного вывода System.out.printf(), который более удобен, например, для представления результатов в виде аккуратной таблицы. Вызов метода имеет форму:

System.out.printf (format,
list);

Строка форматов **format** содержит **форматные коды**, по одному накаждый элемент списка выводимых переменных **list**. Форматный код имеет представленную ниже структуру (в скобки [и] заключены необязательные элементы, т.е. в программе такой элемент или присутствует без скобок, или отсутствует)

%[номер_аргумента$][флаги][место][.дробь]тип

Флаги **- + 0 , (** определяют дополнительные характеристики выводимых данных: минус – выравнивание влево, + – обязательный вывод знака числа, пробел – пробел вместо плюса

для положительных чисел, ноль – заполнение старших пустых позиций числа нулями, запятая – локализованный разделитель в числах, скобка – отрицательные числа заключаются в скобки.

Параметр тип определяет вариант преобразования внутреннего представления данных во внешнюю форму:

- s − String (строка символов);
- c − char (символ);
- d − десятичное целое;
- o - восьмеричное целое;
- x − шестнадцатеричное целое;
- f − действительное в фиксированном формате (в форме 999.9999);
- e − действительное в математическом формате (в форме 0.999E99);
- tD − дата в форме «месяц-день-год»;
- tF − дата в формате «год-месяц-день»;
- tT − время в формате «часы:минуты:секунды».

Параметры место и дробь определяют соответственно количество позиций для выводимых данных (ширину поля) и количество цифр в дробной части (применяются с параметрами типа d и f). Например, %08.3f применяется для вывода действительных данных. Под данные отводится 8 позиций (с учетом знака и разделяющей точки). Из них под дробную часть отводится 3 позиции. При необходимости в целую часть добавляются ведущие нули, чтобы полностью использовать 8 позиций.
Элементы между форматными кодами выводятся как литералы, в частности,
\t – табуляция, \n – переход на новую строку (**%n** также означает переход на другую строку).

Примеры:

```
printf("Hello %s!", "World"); // "Hello World!"
printf("%7d", 1); //  "      1" – минимум 7 позиций
printf("%07d", 1); //  "0000001" – начальные позиции
                                  //заполнены нулями
printf("%.10f", Math.PI);  //  "3,1415926536" – с точностью
                           // до 10 знаков после запятой
printf("%tF", new Date()); //  "2011-01-27"
printf("%tT", new Date()); //  "22:42:37"
```

Фрагмент программы:

```
int a = 256;
double b = 312.678951236;
String str = "форматированный вывод";
System.out.printf("Демонстрируем %s: a=%08d(10)=%x(16), b=%010.4f\n",
                  str,a,a,b);
```

Данный фрагмент повлечет за собой следующий вывод в окно терминала:

```
Демонстрируем форматированный вывод: a=00000256(10)=100(16), b=00312,6790
```

Элементы между форматными кодами (выделены красным в вызове метода printf()) выводятся как строковые литералы (константы).

3.4. Целочисленные типы данных

Характеристики целочисленных типов данных приведены в таблице 4.1.Таблица 4.1 – Целочисленные типы данных и их характеристики

Тип	Значение	Значение по умолчанию	Размер (биты)	Min и Max значения	Класс-оболочка
byte	Целое	0	8	от -128 до 127	Byte
short	Целое	0	16	от -32768 до 32767	Short
int	Целое	0	32	от -2147483648 до 2147483647	Integer
long	Целое	0	64	-9223372036854775808	Long

Для каждого целочисленного типа существует библиотечный класс- оболочка [1, 3], содержащий полезные методы для работы с целочисленными данными. Названия этих классов приведены в четвертом столбце таблицы4.1.

Внутреннее и внешнее представление целочисленных данных.

Внутренне представление – это то, которое «видит» процессор, внешнее – то, которое видит и применяет при записи литералов (в данном случае, целых чисел) программист. Внутреннее представление целочисленных данных – двоичное, количество битов приведено в столбце «Размер» таблицы 4.1. Под знак числа выделен левый бит. Значение 0 этого бита имеют положительные числа, значение 1 – отрицательные. Формат двоичного целого числа представлен на рисунке 4.2. Для типа byte n=8, для типа short n=16, для типа int n=32, для типа long n=64.

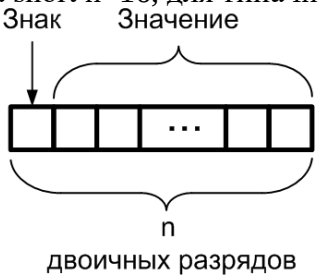


Рисунок 4.2 – Формат целого двоичного числа

Отрицательные числа представляются Java в дополнительных кодах следующим образом:

Знак=1;

Значение* = 2^{n-1} — |Значение|,

где Значение* - *значение числа в дополнительном коде,*

Значение – *обычное представление двоичного числа.*

Принято, что для любого положительного числа дополнительный код совпадает с прямым кодом (обычным представлением числа в двоичной позиционной системе счисления).

Все операции с целыми числами выполняются в двоичной системе счисления. Использование дополнительного кода для представления двоичных чисел позволяет процессору корректно выполнять операциисложения и вычитания на двоичном сумматоре.

Литералы (целые числа, присутствующие в программе) автоматически переводятся из внешнего представления (последовательность символов) во внутреннее

(последовательность битов двоичного целого числа). Метод println() автоматически преобразует числа из внутреннего во внешнеепредставление. Все другие преобразования выполняются под контролем программиста.

Запись целочисленных литералов в Java-программах.

Восьмеричные значения обозначаются в Java ведущим нулем.Десятичные значения не могут иметь ведущего нуля.

Значение 09, встреченное в программе, вызовет ошибку компилятора,т.к. цифра 9 ─ вне восьмеричного диапазона от 0 до 7.

Шестнадцатеричную константу обозначают с ведущими нулями: 0x или 0X.

Целые литералы создают значение типа int (32-разрядное целое число). Когда литеральное значение назначается byte- или short-переменной,

ошибка не генерируется, если это значение находится в пределах диапазона целевого типа.

Целый литерал может всегда назначаться переменной типа long. Однако, чтобы задать длинный литерал, нужно явно сообщить компилятору, что значение имеет тип long. Это осуществляется добавлением символа L (в верхнем или нижнем регистре).

Например, 0x7FFFFFFFFFFFFFFL или 92233720368547758O7L – самый большой целочисленный литерал.

Операции для целочисленных типов данных.

Арифметические операции – (+ - * / %) – (сложение, вычитание умножение, деление нацело, вычисление остатка от деления). Результат этих операций – целочисленный.

Операции сравнения (или отношения) – (< > <= >= == !=) – (меньше, больше, меньше или равно (не больше), больше или равно (не меньше), равно, не равно). Результат сравнения – булевское значение **true** (истина) или **false** (ложь), например, 2<3 имеет значение true, а 2>3 – значение false.

Поразрядные логические операции будут рассмотрены в следующей лабораторной работе.

Арифметические операции с целочисленными данными выполняются точно, без округления. При выполнении целочисленного деления, дробная часть результата отбрасывается. В

Виду ограниченного диапазона (таблица 4.1), сложение и вычитание могут дать «переполнение» – результат, выходящий за границы диапазона. Переполнение выглядит как отрицательный результат при положительных слагаемых или как положительный результат при отрицательных слагаемых. Например, если к максимальному положительному int, равному 0x7FFF FFFF, прибавить 1, получится наименьшее целое отрицательное число 0x8000 0000. Если это число сложить с максимальным положительным (0x8000 0000 + 0x7FFF FFFF), получится число 0xFFFF FFFF (двоичное представление числа -1 в дополнительном коде), которое на единицу меньше нуля (легко проверить, прибавив к нему 1).

При соединении ряда операций в одном выражении следует учитывать приоритеты операций. Так, арифметические операции имеют большийприоритет, чем операции сдвигов и поразрядные операции, а операции над одним операндом имеют больший приоритет, чем операции над двумя операндами.

При вычислении значения выражения Java автоматически расширяет (повышает) тип каждого byte- или short-операнда и результата до int (т.к. результат сложения двух коротких операндов с одинаковыми знаками может не поместиться в короткий формат).

Это может вызывать плохо распознаваемые ошибки во время компиляции.

Например, казалось бы, благополучный код

```
byte      b=50;
b=b*2;
```

вызовет ошибку компиляции «Возможна потеря точности – possible loss of precision», т.к. значение выражения b*2 имеет тип int, а целевая переменная bимеет тип byte. Такая же ошибка возникнет и при компиляции строки кода **int i = 2L;** (переменной типа int присваивается значение типа long).Вообще, когда размер переменной слева меньше, чем размер результата справа, возникает ошибка «Возможна потеря точности – possible loss of precision».

Для преодоления этой трудности есть два пути:

1) числа длиной 1 байт хранить, например, в переменных типа int;
2) использовать явное преобразование типа значения в правой части оператора присваивания:

```
byte      b=50;
b=(byte)(b*2);
.
```

Следует отметить, что тип int – наиболее универсален и эффективен и должен быть использован для расчетов, индексации элементов массива или выполнения целочисленных операций.

Может показаться, что использование short или byte экономит память, но нет никакой гарантии, что Java не будет внутренне так или иначе расширять эти типы до int, Помните, что тип определяет поведение, а не размер. Единственное исключение – массивы, где тип byte гарантирует использование только одного байта на элемент массива, тип short будет забирать 2 байта, а int 4 байта [1].

3.5. Порядок выполнения работы

3.5.1. Работа в окне кода BlueJ

1) Введите целые константы в окне кода, нажимая после каждой константы Enter: **`35 -246 5671L.`**

2) Введите выражения **`4*3+5 4+3*5 25-3/5-6+20/3 5%6`** («пять помодулю шесть» − остаток от деления 5 на 6) **`7%6 7%5.`** После ввода каждого выражения нажимайте Enter. Обратите внимание на тип и результат.

3) Наберите выражения **`1==1 1<5 2<=5 2>6 3/2>1 2!=9.`** После ввода каждого выражения нажимайте Enter. В этих выражениях используются операции сравнения. Результат операции (и выражения) имеет логический тип (boolean). Если выражение верно, то его значение равно true, иначе – false. Проверяйте результат каждый раз после нажатия Enter.

4) ведите в панели кода действительные литералы, каждый в отдельной строке **`3.1415 2.71823F 0.314E+1.`** Проанализируйте результаты (что происходит после нажатия клавиши Enter).

5**).** Наберите в панели кода выражения **`25-3.5-6+21.3 5/6.0 5.0/6 3.7f/5.3f`**. Не забывайте нажимать Enter! Обратите внимание на

значение результата и на его тип.Наберите в панели кода выражения **1==1.0 1.0<1.5 2.3<=2.3f 1.5==1.5f**

7) Наберите в панели кода: **`double x; float a=3.5; float d=2.5;`** (После точки с запятой нажимайте Enter). Проанализируйте сообщения системы.

8) Наберите в панели кода: **`x=3.5;`** (тип x уже задан) **`float a=3.5f; double d=2.5f;`** (После точки с запятой нажимайте Enter). Проверьте значения переменных, введя имя и Enter.

9) Наберите **`d=d+3.75; x=a+7.5f;`** Проверьте значения переменных.

10) Введите **`final float K=5.4f;`** Проверьте значение K. Наберите оператор **`K=5.45;`** . Обратите внимание на сообщение. Наберите **`final double PI=3.1415926;`** , вычислите длину окружности с радиусом 10: **`2*PI*10`**.

3.5.2. Разработка программы с использованием методаформатированного вывода System.out.printf()

Разработайте программу согласно варианту задания (таблица 3.1), проведите ее отладку и испытание на нескольких тестовых примерах.

В качестве индивидуального задания на лабораторную работу предлагается разработать программу, которая выполняет следующие действия:

1) определяет и инициализирует (задает тип и значение) переменные str, a, b, c, d;

2) осуществляет следующий вывод:

Привет, значение_str: a=значение_a, b=значение_b, c=значение_c, d=значение_d!

Значения переменных должны выводиться в формате, предусмотренном вариантом задания. После вывода строки, должен быть осуществлен переход на следующую строку.

После выявления синтаксических ошибок программа должна быть откомпилирована и запущена не менее трех раз (с различными значениями переменных). В качестве тестовых значений переменных студент долженподобрать такие значения, которые проверяют все возможности программы. Например, проверку обязательного вывода знака числа нужно осуществитьна положительных и отрицательных значениях. Желательно также проверить, как будет вести себя ваша программа, если заданное значение переменной не помещается в формат, заданный в методе printf().Варианты индивидуальных заданий представлены в таблице 3.1.

Использованы следующие условные обозначения:С10 – десятичная система счисления;
С16 – шестнадцатеричная система счисления;С8 – восьмеричная система счисления;
Ш – ширина (минимальное количество цифр);Т – точность (число цифр после запятой);
НН – наличие ведущих нулей; ОН –

отсутствие ведущих нулей;ВЗ –
обязательный вывод знака.

3.5.3 Разработайте программу согласно варианту задания (таблица 3.2)

В качестве индивидуального задания на лабораторную работу ,

предлагается разработать программу, демонстрирующую выполнение арифметических операций над целыми числами.
В программе задать целочисленные переменные a, b, c. Вывести на экран в двоичной и десятичной системах счисления значения a, b, c, а также результаты выполнения операций. -a, a+b, a-b, a*b, a/b, a%b, a++, b--. Например,

a+b=100000000(2)=100(16)= 400(8)=256 (10); .

Подобный вывод можно осуществить одним вызовом метода printf(). Для получения представления двоичного числа в виде строки (String) используйтебиблиотечную функцию **Integer.toBinaryString(x)**, где x – аргумент типа int.

Сравнить с результатами ручных расчетов.

Варианты заданий представлены в таблице 3.2.

3.6. Варианты заданий

Таблица 3.1 – Варианты заданий

Номер варианта	Строка str	Целоеа (int)	Целое b (short)	Целое c (byte)	Действительное d (double)	Действительное f (float)
1	Фамилия_Группа:	С10, Ш8, НН, ВЗ	С8	С16	Ш10, Т5, НН,	Ш8, Т3, ОН
2	Фамилия_Группа:	С16	С10, Ш5, ОН	С8	Ш5, Т3, ОН, ВЗ	Ш7, Т2, НН
3	Фамилия_Группа:	С8	С16	С10, Ш6, ОН	Ш8, Т4, НН	Ш6, Т3, ОН, ВЗ
4	Фамилия_Группа:	С16	С8	С10,	Ш7, Т3,	Ш9, Т4,
				Ш8, НН	ОН, ВЗ	НН
5	Фамилия_Группа:	С10, Ш5, ОН, ВЗ	С8	С16	Ш8, Т2, НН	Ш10, Т4, ОН
6	Фамилия_Группа:	С10, Ш5, ОН	С8	С16	Ш8, Т2, ОН, ВЗ	Ш6, Т3, НН
7	Фамилия_Группа:	С8	С10, Ш9, НН	С16	Ш10, Т4, ОН	Ш8, Т2, НН, ВЗ
8	Фамилия_Группа:	С16	С10, Ш7, ОН, ВЗ	С8	Ш12, Т6, НН	Ш6, Т3, ОН
9	Фамилия_Группа:	С16	С8	С10, Ш8, НН,	Ш11, Т5, ОН, ВЗ	Ш9, Т4, НН
10	Фамилия_Группа:	С9, Ш7, ОН, ВЗ	С8	С16	Ш14, Т5, НН,	Ш8, Т3, ОН

11	Фамилия_Группа:	С10, Ш8, НН, ВЗ	С8	С16	Ш10, Т5, НН,	Ш8, Т3, ОН
12	Фамилия_Группа:	С16	С10, Ш5, ОН	С8	Ш5, Т3, ОН, ВЗ	Ш7, Т2, НН
13	Фамилия_Группа:	С8	С16	С10, Ш6, ОН	Ш8, Т4, НН	Ш6, Т3, ОН, ВЗ
14	Фамилия_Группа:	С16	С8	С10, Ш8, НН	Ш7, Т3, ОН, ВЗ	Ш9, Т4, НН
15	Фамилия_Группа:	С10, Ш5, ОН, ВЗ	С8	С16	Ш8, Т2, НН	Ш10, Т4, ОН
16	Фамилия_Группа:	С10, Ш5, ОН	С8	С16	Ш8, Т2, ОН, ВЗ	Ш6, Т3, НН
17	Фамилия_Группа:	С8	С10, Ш9, НН	С16	Ш10, Т4, ОН	Ш8, Т2, НН, ВЗ
18	Фамилия_Группа:	С16	С10, Ш7, ОН, ВЗ	С8	Ш12, Т6, НН	Ш6, Т3, ОН
19	Фамилия_Группа:	С16	С8	С10, Ш8, НН,	Ш11, Т5, ОН, ВЗ	Ш9, Т4, НН
20	Фамилия_Группа:	С9, Ш7, ОН, ВЗ	С8	С16	Ш14, Т5, НН,	Ш8, Т3, ОН
21	Фамилия_Группа:	С10, Ш8, НН, ВЗ	С8	С16	Ш10, Т5, НН,	Ш8, Т3, ОН
22	Фамилия_Группа:	С16	С10, Ш5, ОН	С8	Ш5, Т3, ОН, ВЗ	Ш7, Т2, НН
23	Фамилия_Группа:	С8	С16	С10, Ш6, ОН	Ш8, Т4, НН	Ш6, Т3, ОН, ВЗ
24	Фамилия_Группа:	С16	С8	С10, Ш8, НН	Ш7, Т3, ОН, ВЗ	Ш9, Т4, НН
25	Фамилия_Группа:	С10, Ш5,	С8	С16	Ш8, Т2,	Ш10, Т4,
		ОН, ВЗ			НН	ОН

Таблица 3.2 – Варианты заданий

Номер варианта	A	b	c
1	356	725	(b-a)%a+56
2	125	49	(a+b)/b-a+100%b
3	1024	541	a%b*2-b/4
4	986	25	a/(b+20)-253%b
5	1255	345	(a+b)/(b+20)%20
6	236	16	a/b%5+12*b

7	512	48	a/b+(a%b)*2
8	2056	185	(a+2*b)/(a-5*b)%9
9	145	312	a/16+b%16-10*a
10	1345	1200	(a-b)/5+(b-500)%5*3
11	712	256	(b-a+a/4+b*4)%10
12	123	56	a*b-(a+b/2)%7
13	875	298	(a%b-a/b)*20
14	1421	1200	(a-b/2+b%10)*3
15	127	512	a/b*(a-b)%15+25
16	2024	341	-(a/b+a%b)*5
17	456	1348	(b/a+b%a*3)/10
18	25	354	(a*a-b)%25/2
20	5	7	(a*a+b*b*b)/10%5
21	15	12	((a*a-b*b)+1024)%12/3
22	1500	2800	((b-a)/10)*((b-a)/10)%20
23	456	1234	(2*a+b/45)*3%14
24	1155	956	((b-a)*2560%b)/12
25	1024	2048	-2*(a%100+b%200)/5

3.9. Контрольные вопросы

1) Какие простые типы данных существуют в Java?
2) От чего зависит диапазон значений числовых типов?
3) Какой из типов обеспечивает б*о*льшую точность double или float?
4) Что такое литерал?
5) Как задается переменная?
6) Как задается именованная константа?
7) Как задать литерал типа float?
8) Как задать литерал типа long?
9) Опишите, как работает оператор присваивания?
10) Когда возникает ошибка «Потеря точности»?
11) Какие возможности предоставляет окно кода BlueJ?

12) Чему равно значение выражения 2<5? Какого оно типа?
13) Чему равно значение выражения 10<=5? Какого оно типа?
14) Какие возможности предоставляет оператор форматированного выводаSystem.out.printf (format, list); .
15) Перечислите возможности текстового процессора Microsoft Word, которые вы использовали для оформления списков при выполнении отчета по лабораторной работе.
1) Что такое литерал?
2) Как в программе задать целочисленные литералы в различных системах счисления?
3) Как получить представление двоичного целого числа в виде строки символов (String)?
4) Какие арифметические операции и операции сравнения допустимы для целочисленных типов?
5) Как выполняется операция целочисленного деления?
6) Как выполняется операция, обозначаемая символом %?
7) Что такое переменная? Как объявляется переменная целочисленного типа?
8) Как задать именованную константу целочисленного типа?
9) Какие ошибки могут возникать при компиляции оператора присваивания, использующего переменные и выражения различных целых типов?Приведите примеры.
10) Какой тип будет у результата выражения, в котором выполняются операции над переменными типа short и byte?