**АРХАНГЕЛЬСКИЙ КОЛЛЕДЖ ТЕЛЕКОММУНИКАЦИЙ**

**ИМ. Б.Л. РОЗИНГА (ФИЛИАЛ) СПбГУТ**

**(АКТ (ф) СПбГУТ)**

**Отчеты по лабораторным работам**

**ОАиП**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Студент | ИСПП-31 | |  |  | *Арсланов Р.Е.* |
|  | (Группа) | | (Подпись) | (Дата) | (И.О. Фамилия) |
| Преподаватель | | |  |  | *Садовский Р. В. Абрамова П.А.* |
|  | |  | (Подпись) | (Дата) | (И.О. Фамилия) |

Архангельск 2024

**Лабораторная работа №9**

**Организация процедур**

1. **Цель работы:**
   1. Изучить процесс написания процедур средствами языка C++.
2. **Ответы на контрольные вопросы**
   1. Процедура - это разновидность функции, которая ничего не возвращает.
   2. Функция - подпрограмма, выполняющая какие-либо операции и возвращающая значения.
   3. Любой тип
   4. Ключевое слово – return
   5. Void название процедуры( тип данных имя переменной(ых)){тело процедуры}
3. **Вывод:**
   1. Изучили процесс написания процедур средствами языка C++.

**Лабораторная работа №10**

**Работа с текстовыми файлами**

**1 Цель работы:**

* 1. Научиться работать с файлами средствами языка C++.

1. **Ответы на контрольные вопросы:**
   1. Используйте класс std::ofstream для открытия файла в режиме добавления
   2. Порядок закрытия файла в C++ следующий: после завершения работы с файлом его следует закрыть с помощью функции close()
   3. Используйте класс std::ifstream для открытия файла в режиме чтения
2. **Вывод:**
   1. Научились работать с файлами средствами языка C++.

**Лабораторная работа №11**

**Работа с текстовыми файлами с использованием потоков**

**1 Цель работы:**

* 1. Изучить процесс обработки текстовых файлов средствами языка C++ с использованием потоков.

1. **Ответы на контрольные вопросы:**
   1. ofstream — для вывода (записи) данных в файл. ifstream — для ввода (чтения) данных из файла. fstream — для чтения и для записи данных (двунаправленный обмен).
   2. Право на чтение (r — read). Право на запись (w — write).
   3. Для открытия файла в C++ используется функция open()
   4. Открытие файлов, Запись данных в файл, Чтение данных из файла, Закрытие файла
   5. Используется функция feof()
2. **Вывод:**
   1. Изучили процесс обработки текстовых файлов средствами языка C++ с использованием потоков.

**Лабораторная работа №12**

**Разработка структур**

**1 Цель работы:**

* 1. Изучить процесс разработки и применения структур на языке C++.

1. **Ответы на контрольные вопросы:**
   1. Структура в C++ — это пользовательский тип данных, который позволяет объединять переменные разных типов под одним именем.
   2. Структура должна содержать в себе: имя структуры и компоненты структуры
   3. Формат определения структуры: struct имя\_структуры { компоненты\_структуры }.
   4. Для обращения к элементам структуры используется операция «точка»:имя\_переменной\_структуры.имя\_элемента
   5. поля структуры могут быть различными данными, но не должны ими быть.
2. **Вывод:**
   1. Изучили процесс разработки и применения структур на языке C++.

**Лабораторная работа №13**

**Работа с двоичными файлами**

**1 Цель работы:**

* 1. Изучить процесс обработки двоичных файлов средствами языка C++ с использованием потоков.

1. **Ответы на контрольные вопросы:**
   1. Увеличивается скорость извлечения из файла данных, не носящих чисто текстового характера, так как не требуется действий по преобразованию их из текстового формата в двоичный.
   2. Для записи информации в двоичные файлы на C++ можно использовать функцию fwrite.
   3. Для считывания информации из двоичных файлов на C++ можно использовать функцию fread.
   4. Чтобы считать переменные стандартных типов данных из двоичного файла на C++, можно использовать метод read() класса fstream.
   5. При помощи QDataStream:: readRawData:
2. **Вывод:**
   1. Изучили процесс обработки двоичных файлов средствами языка C++ с использованием потоков.

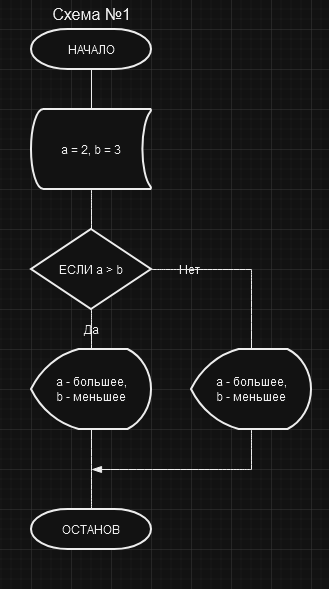
**Лабораторная работа №14**

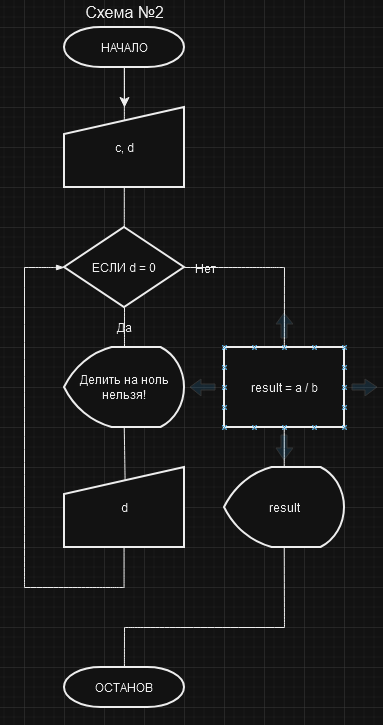
**Изучение правил разработки схем алгоритмов**

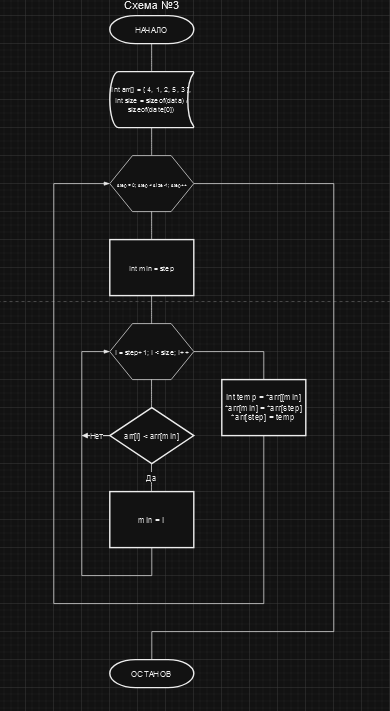
**1 Цель работы:**

* 1. Изучить процесс построения схем алгоритмов согласно ГОСТ 19.701-90;
  2. Научиться применять графические редакторы на этапе проектирования программного обеспечения;
  3. Закрепить навык разработки программ методами структурного программирования.

1. Ход работы:







1. **Ответы на контрольные вопросы:**
   1. ГОСТ 19.701-90 (ИСО 5807-85) Единая система программной документации (ЕСПД)
   2. Во вкладке «общие» находятся графические примитивы для создания схем алгоритмов.
   3. Нужно выделить нужные фигуры затем нажать на вкладку положение потом на кнопку выравнивание и выбрать нужное выравнивание.
   4. Чтобы изменить соединительную линию на стрелку нужно нажать на соединительную линию и выбрать вкладку «Линия», где можно поменять вид на стрелку.
   5. Нужно нажать на нужную фигуру затем в меню справа перейти на вкладку положение и там уже выбрать нужный размер.
2. **Вывод:**
   1. Изучили процесс построения схем алгоритмов согласно ГОСТ 19.701-90;
   2. Научились применять графические редакторы на этапе проектирования программного обеспечения;
   3. Разработали навык разработки программ методами структурного программирования.

**Лабораторная работа №15**

**Тестирование, отладка приложения**

**1 Цель работы:**

* 1. Изучить возможности отладки приложений в MS Visual Studio

1. **Ответы на контрольные вопросы:**
   1. Этап разработки программы на котором обнаружают, локализируют и устраняют ошибки.
   2. Этапы отладки: 1) в режиме отладки собрать информацию об ошибке; 2) локализировать ошибку; 3) определить ошибку; 4) исправить ошибку; 5) Протестировать код.
   3. Это преднамеренное прерывание программы в установленном разработчиком месте.
   4. Кликнуть по левой части строки ЛКМ.
   5. f5 для отладки по точкам останова, f10 пошагово, f11 со входом в функцию.
   6. Локальные(просмотр переменных внутри текущей области), видимые(значение всех переменных, используемые в текущей или предыдущей строке).
   7. ctrl + b прерывание на функцию; ctrl + shift +f5 быстрый перезапуск; shift + f11 шаг с выходом.
2. **Вывод:**
   1. Изучили возможности отладки приложений в MS Visual Studio

**Лабораторная работа №16**

**Программирование модуля**

**1 Цель работы:**

* 1. Изучить процесс создания и применения модулей в программах на языке С++.

1. **Ответы на контрольные вопросы:**
   1. Некоторые коды могут быть разделены на модули для удобства разработки в больших командах программистов. а также использование 1 и того же модуля в разных проектах, чтобы не переписывать его несколько раз.
   2. использование модуля в разных проектах, чтобы не переписывать одинаковый код несколько раз (Эффективность).
   3. module.h – шапка модуя (в нем объявляются функции); module.cpp – хранит код функций; main.cpp — код программы.
   4. #include “module.h”.
2. **Вывод:**
   1. Изучили процесс создания и применения модулей в программах на языке С++.

**Лабораторная работа №17**

**Создание библиотеки подпрограмм**

**1 Цель работы:**

* 1. Изучить процесс создания и применения библиотек в программах на языке С++.

1. **Ответы на контрольные вопросы:**
   1. Назначение библиотек в C++ заключается в предоставлении готовых функций, классов и переменных для выполнения различных задач.
   2. Обычно расширение файла будет .a для статической библиотеки и либо .dll (в Windows), либо .so (в Linux) для динамической библиотеки.
   3. Это специальное расширение Microsoft для C++, которое позволяет приписывать типу или функции информацию о классе хранения.
   4. В верхней части заголовочного файла библиотеки на C++ обычно пишут директивы препроцессора (#ifndef и #define)
   5. Перед объявлением типа функции в заголовочном файле библиотеки на C++ необходимо использовать препроцессорную директиву #include, которая указывает компилятору обработать заголовочный файл перед компиляцией
2. **Вывод:**
   1. Изучили процесс создания и применения библиотек в программах на языке С++.

**Лабораторная работа №18**

**Работа с алгоритмами сортировки массивов**

**1 Цель работы:**

* 1. Изучить принципы реализации алгоритмов сортировки массивов средствами языка программирования С++.

1. Ответы на контрольные вопросы:
   1. Сортировка массива в C++ означает упорядочивание его элементов по возрастанию или убыванию значений.
   2. Пузырьковая сортировка (Bubble sort); Сортировка выбором (Selection sort); Сортировка вставками (Insertion sort); Быстрая сортировка (Quick sort); Сортировка слиянием (Merge sort); Сортировка Шелла (Shell sort); Сортировка кучей (Heap sort).
   3. Несколько способов обменять значения элементов массива: Временная переменная; Метод splice().
2. Вывод:
   1. Изучили принципы реализации алгоритмов сортировки массивов средствами языка программирования С++.

**Лабораторная работа №19**

**Использование указателей для организации связанных списков**

**1 Цель работы:**

* 1. Изучить процесс применения указателей для организации списков в программах на языке С++.

1. **Ответы на контрольные вопросы:**
   1. Динамические структуры данных – это структуры данных, память под которые выделяется и освобождается по мере необходимости.
   2. Виды динамических структур в C++: Список; Стек.
   3. В самоссылочных структурах на C++ указатель на структуру указывает на всю структуру целиком, а не на отдельные компоненты, находящиеся внутри неё.
   4. Динамические структуры называются самоссылочными, потому что кроме информационных полей они обязательно содержат поля-указатели на свой собственный тип структуры.
   5. Стек организован по принципу LIFO (Last In First Out, «последним пришёл — первым вышел»). Это значит, что последний элемент, который добавили в стек, первым выйдет из него.
   6. LIFO (Last In First Out, «последним пришёл — первым вышел»)
2. **Вывод:**
   1. Изучили процесс применения указателей для организации списков в программах на языке С++.

**Лабораторная работа №20**

**Изучение процесса разработки классов на С++**

**1 Цель работы:**

* 1. Изучить процесс разработки классов на С++;
  2. Изучить реализацию механизмов полиморфизма и инкапсуляции на языке C++;
  3. Закрепить навык разработки программ методами структурного, процедурного и модульного программирования.

1. **Ответы на контрольные вопросы:**
   1. это специальная функция, направленная на инициализацию, то есть создание, подготовку к работе, определение параметров
   2. это разное поведение одного метода в разных классах
   3. Это использование нескольких методов с одним и тем же именем, но с разными типами параметров
   4. это ограничение доступа к методам и атрибутам, делающий их защищёнными или приватными
   5. public, protected, private
   6. разница между структурами и классами в том, что в классах все private по умолчанию
   7. классы используются для описания объектов
2. **Вывод:**
   1. Изучили процесс разработки классов на С++;
   2. Изучили реализацию механизмов полиморфизма и инкапсуляции на языке C++;
   3. Закрепили навык разработки программ методами структурного, процедурного и модульного программирования.

**Лабораторная работа №21**

**Изучение процесса разработки дочерних классов на С++**

**1 Цель работы:**

* 1. Изучить процесс разработки дочерних классов на С++;
  2. Изучить реализацию механизмов наследования, инкапсуляции и полиморфизма на языке C++;
  3. Закрепить навык разработки программ методами структурного, процедурного и модульного программирования.

1. **Ответы на контрольные вопросы:**
   1. это механизм, посредством которого один класс может преобретать свойства другого
   2. Это когда класс приобретает свойства от нескольких классов.
   3. Классы, которые дают основу свойств для других классов
   4. Классы, которые наследуют свойства от родительского класса
   5. Это метод класса, который может быть переопределён в дочерних классах
   6. Классы, в которых есть хотя бы один пустой виртуальный метод, который будет определяться в дочерних классах
   7. Абстрактные классы нужны, чтобы объединить реализацию нескольких схожих классов
2. **Вывод:**
   1. Изучили процесс разработки дочерних классов на С++;
   2. Изучили реализацию механизмов наследования, инкапсуляции и полиморфизма на языке C++;
   3. Закрепили навык разработки программ методами структурного, процедурного и модульного программирования.

**Лабораторная работа №22**

**Изучение процесса разработки шаблонов функций и классов на С++**

**1 Цель работы:**

* 1. Изучить процесс разработки шаблонов функций и классов на С++;
  2. Изучить реализацию параметрического полиморфизма на языке C++;
  3. Закрепить навык разработки программ методами структурного, процедурного и модульного программирования

1. **Ответы на контрольные вопросы:**
   1. Шаблоны используются для кодирования обобщённых алгоритмов без привязки к некоторым параметрам
   2. template <class тип>

возвращаемый\_тип имя\_функции(список параметров) {

тело функции

}

* 1. template <class T> class имя класса {

тело класса

}

* 1. Чтобы передать в шаблон несколько обобщённых типов, их нужно перечислить через запятую с помощью ключевого слова class или typename
  2. Значения параметров шаблона при создании объекта задаются после имени шаблона в скобках

1. **Вывод:**
   1. Изучили процесс разработки шаблонов функций и классов на С++;
   2. Изучили реализацию параметрического полиморфизма на языке C++;
   3. Закрепили навык разработки программ методами структурного, процедурного и модульного программирования

**Лабораторная работа №23**

**Изучение среды разработки**

**1 Цель работы:**

* 1. Изучить процесс разработки оконных приложений на языке C# в среде Visual Studio;
  2. Изучить способы настройки свойств формы и процесс обработки событий.

1. Ответы на контрольные вопросы:
   1. Application задает полное название производного класса приложения
   2. Метод Main
   3. Метод Main запускает главную форму приложения.
   4. Forms
   5. В Form1.cs содержится код, в Form1.Designer.cs содержится код визуальной части приложения
   6. InitializeComponent() и вызывается он в конструкторе формы.
   7. Show - запускает отображение обычного окнаShowDialog - открывает форму как диалоговое окно
   8. Close()
   9. в обработчике события FormClosing. Нужно задать для свойства Cancel объекта, переданного FormClosingEventArgs обработчику событий, значение true.
2. **Вывод:**
   1. Изучили процесс разработки оконных приложений на языке C# в среде Visual Studio;
   2. Изучили способы настройки свойств формы и процесс обработки событий.

**Лабораторная работа №24**

**Создание проекта с использованием компонентов для работы с текстом**

**1 Цель работы:**

* 1. Изучить различные типы полей ввода, применяющихся в оконных приложениях на C#;
  2. Изучить свойства полей ввода и процесс обработки событий полей ввода.

1. **Ответы на контрольные вопросы:**
   1. Элемент управления TextBox в C# используется для получения входных данных от пользователя или для отображения текста.
   2. Элемент управления Label в C# используется для отображения простого текста на форме, доступного только для чтения.
   3. Событие TextChanged срабатывает, когда пользователь вводит что-то в TextBox и в результате меняется значение. Также оно активируется, когда программа сама изменяет текст TextBox в коде.
   4. В C# для обработки нажатий клавиш используются следующие события: KeyDown, KeyPress, KeyUp
   5. Обработаем событие KeyDown для текстового поля и выведем данные о нажатой клавише в текстовый блок.
2. **Вывод:**
   1. Изучили различные типы полей ввода, применяющихся в оконных приложениях на C#;
   2. Изучили свойства полей ввода и процесс обработки событий полей ввода.

**Лабораторная работа №25**

**Создание проекта с использованием компонентов ввода и отображения чисел, дат и времени**

**1 Цель работы:**

* 1. Изучить различные типы полей ввода, применяющихся в оконных приложениях на C#;
  2. Изучить свойства полей ввода и процесс обработки событий полей ввода.

1. **Ответы на контрольные вопросы:**
   1. Элемент управления TrackBar в Windows Forms (иногда называемый элементом управления «ползунок») используется для перемещения по большому объему информации или для визуальной настройки числовых параметров.
   2. Элемент управления NumericUpDown в C# используется для выбора числа из определённого диапазона.
   3. Элемент DateTimePicker управления позволяет пользователю выбрать дату и время, а также отобразить эти дату и время в указанном формате.
   4. Чтобы произошло ValueChanged событие, Value свойство можно изменить в коде, нажав кнопку вверх или вниз или введя новое значение, которое считывается элементом управления. Новое значение считывается, когда пользователь нажимает клавишу ВВОД или переходит в сторону от элемента управления.
2. **Вывод:** 
   1. Изучили различные типы полей ввода, применяющихся в оконных приложениях на C#;
   2. Изучили свойства полей ввода и процесс обработки событий полей ввода.

**Лабораторная работа №26**

**Создание проекта с использованием компонентов выбора**

**1 Цель работы:**

* 1. Изучить свойства и процесс обработки событий элементов выбора в приложении на C#.

1. **Ответы на контрольные вопросы:**
   1. Элемент управления ListBox в C# поддерживает множественный выбор. Для этого нужно установить свойство SelectionMode в значение «Multiple» или «Extended»
   2. RadioButton в C# — это элемент управления, позволяющий пользователю выбрать один из нескольких вариантов.
   3. Событие CheckedChanged в C# срабатывает, когда изменяется свойство Checked.
   4. Text у компонента класса RadioButton — строка, описывающая назначение компонента; Checked — логическое свойство, позволяющее проверить или задать выбор компонента; Items у компонента класса ListBox — свойство, описывающее список объектов и методы работы с ними;
2. **Вывод:** 
   1. Изучили свойства и процесс обработки событий элементов выбора в приложении на C#.

**Лабораторная работа №27**

**Создание проекта с использованием компонентов отображения списков**

**1 Цель работы:**

* 1. Изучить свойства и процесс обработки событий элементов выбора и элементов отображения списков в приложении на C#.

1. **Ответы на контрольные вопросы:**
   1. Класс ListBox представляет распространенный компонент среды Windows — списки переменной длины, которые позволяют пользователю выбрать один из элементов.
   2. ComboBox в C# — это элемент управления, который образует выпадающий список и совмещает функциональность компонентов ListBox и TextBox.
   3. CheckedListBox — это элемент управления, который отображает данные в формате списка и позволяет пользователям проверять или проверять один или несколько элементов.
   4. Событие SelectedIndexChanged срабатывает, когда изменяется индекс выделенного элемента.
   5. Событие CheckedChanged в C# срабатывает, когда изменяется свойство Checked.
2. **Вывод:** 
   1. Изучили свойства и процесс обработки событий элементов выбора и элементов отображения списков в приложении на C#.

**Лабораторная работа №28**

**Изучение способов представления и обработки табличных данных в приложениях на C#**

**1 Цель работы:**

* 1. Изучить свойства и процесс обработки событий элемента управления DataGridView в приложении на C#;
  2. Изучить процесс привязки данных к элементу отображения табличных данных.

1. **Ответы на контрольные вопросы:**
   1. Объект DataGridView предназначен для отображения всей информации из таблиц, запросов или фильтров на форме в виде таблицы
   2. Компонент BindingSource предназначен для упрощения процесса привязки элементов управления к источнику данных.
   3. Элемент управления DataGridView может отображать данные в трех разных режимах: с привязкой, без привязки и в виртуальном режиме.
   4. Можно связать их так:

var list = new BindingList<Person>(persons);

myGrid.DataSource = list;  
var list = new List<Person>()

{

new Person { Name = "Joe", },

new Person { Name = "Misha", },

};

var bindingList = new BindingList<Person>(list);

var source = new BindingSource(bindingList, null);

grid.DataSource = source;

* 1. Некоторые типы столбцов, поддерживаемые в DataGridView на C#:  
     DataGridViewTextBoxColumn; DataGridViewCheckBoxColumn; DataGridViewImageColumn; DataGridViewButtonColumn и другие.
  2. С помощью DataRow
  3. Некоторые свойства, которые позволяют настроить стиль отображения строк и ячеек DataGridView на C#:  
     DefaultCellStyle; ColumnHeadersDefaultCellStyle и другие

1. **Вывод:**
   1. Изучили свойства и процесс обработки событий элемента управления DataGridView в приложении на C#;
   2. Изучили процесс привязки данных к элементу отображения табличных данных.

**Лабораторная работа №29**

**Создание проекта с использованием компонентов стандартных диалогов**

**1 Цель работы:**

* 1. Научиться применять компоненты стандартных диалоговых окон в приложении на C#.

1. **Ответы на контрольные вопросы:** 
   1. Для выбора цвета в C# используется диалоговое окно ColorDialog.
   2. Для выбора шрифта в C# используется диалоговое окно FontDialog.
   3. Для выбора файла в C# используется диалоговое окно OpenFileDialog.
   4. SaveFileDialog в C# — это диалоговое окно, которое позволяет пользователю выбрать местоположение и имя файла для сохранения.
   5. Filter: задает фильтр файлов, благодаря чему в диалоговом окне можно отфильтровать файлы по расширению. Фильтр задается в следующем формате Название\_файлов|\*.расширение. Например, Текстовые файлы(\*.txt)|\*.txt. Можно задать сразу несколько фильтров, для этого они разделяются вертикальной линией |. Например, Bitmap files (\*.bmp)|\*.bmp|Image files (\*.jpg)|\*.jpg
   6. FolderBrowserDialog в C# — это модальное диалоговое окно для просмотра и выбора папок в графическом интерфейсе приложения на C#
   7. Path.GetDirectoryName — возвращает информацию о каталоге для указанного пути, представленного символом.
2. **Вывод:**
   1. Научились применять компоненты стандартных диалоговых окон в приложении на C#.

**Лабораторная работа №30**

**Создание проекта с использованием кнопочных компонентов**

**1 Цель работы:**

* 1. Научиться применять и настраивать кнопки в приложении на C#.

1. **Ответы на контрольные вопросы:**
   1. Button в C# — это элемент графического интерфейса, который позволяет пользователю взаимодействовать с приложением или программой.
   2. При нажатии на кнопку в C# происходит событие Click.
   3. Свойство AcceptButton в C# позволяет назначить кнопку по умолчанию, которая будет срабатывать при нажатии на клавишу Enter.
   4. Свойство CancelButton в C# используется для назначения кнопки отмены.
   5. Цикл с постусловием в C# всегда выполняется хотя бы один раз.
   6. Целью использования свойства Anchor является указание того, какие расстояния от краев элемента управления до краев контейнера должны быть фиксированы.
2. **Вывод:**
   1. Научились применять и настраивать кнопки в приложении на C#.

**Лабораторная работа №31**

**Создание проекта с использованием меню и панели инструментов**

**1 Цель работы:**

* 1. Научиться применять и настраивать различные виды меню, панель инструментов, строку состояния в приложении на C#.

1. **Ответы на контрольные вопросы:**
   1. MenuStrip — это компонент для создания меню в Windows Forms. Он выступает своего рода контейнером для отдельных пунктов меню, которые представлены объектом ToolStripMenuItem.
   2. ContextMenuStrip представляет контекстное меню. Данный компонент во многом аналогичен элементы MenuStrip за тем исключением, что контекстное меню не может использоваться само по себе, оно обязательно применяется к какому-нибудь другому элементу, например, текстовому полю.
   3. ToolStrip — это базовый класс для MenuStrip, StatusStrip и ContextMenuStrip. Используйте ToolStrip и связанные с ними классы в новых приложениях Windows Forms для создания панелей инструментов.
   4. StatusStrip представляет строку состояния, во многом аналогичную панели инструментов ToolStrip. Строка состояния предназначена для отображения текущей информации о состоянии работы приложения.
2. **Вывод:** 
   1. Научились применять и настраивать различные виды меню, панель инструментов, строку состояния в приложении на C#.

**Лабораторная работа №32**

**Обработка исключений**

**1 Цель работы:**

* 1. Научиться обрабатывать исключительные ситуации в приложениях.

1. **Ответы на контрольные вопросы:**
   1. В языке C# исключение — это любое состояние ошибки или непредвиденное поведение, возникающее при выполнении программы.
   2. Блок try в C# предназначен для разбиения на разделы кода, который может затрагиваться исключением.
   3. Блок catch в C# предназначен для обработки возможных исключений, которые могут возникать во время выполнения кода.
   4. Блок finally добавляет в программу прогнозируемость, позволяя выполнить определенный код при любых обстоятельствах.
   5. Чтобы сгенерировать исключение в C#, используется оператор throw.
2. **Вывод:** 
   1. Научились обрабатывать исключительные ситуации в приложениях.

**Лабораторная работа №33**

**Обработка файлов в оконном приложении**

**1 Цель работы:**

* 1. Научиться применять классы для работы с файловой системой в приложениях на C#.

1. **Ответы на контрольные вопросы:**
   1. System.IO — пространство имён, которое содержит классы для работы с файловой системой и файловыми потоками в C#
   2. Для работы с текстовыми файлами в C# предназначены классы StreamReader (для чтения) и StreamWriter (для записи)
   3. Для работы с бинарными файлами в C# предназначены классы BinaryReader и BinaryWriter
   4. Несколько классов в C#, которые предоставляют информацию о дисках и каталогах:  
      DriveInfo; DirectoryInfo.
   5. В C# информацию о файлах предоставляют следующие классы из пространства имён System.IO:  
      Directory и File; DriveInfo, DirectoryInfo и FileInfo.
2. **Вывод:** 
   1. Научились применять классы для работы с файловой системой в приложениях на C#.

**Лабораторная работа №34**

**Разработка интерфейса приложения**

**1 Цель работы:**

* 1. Научиться программно создавать и настраивать элементы управления в приложениях на C#;
  2. Научиться использовать таймер в приложениях на C#.

1. **Ответы на контрольные вопросы:**
   1. Некоторые основные свойства таймера в приложении C#:  
      Enabled; Interval; AutoReset.
   2. Чтобы запустить таймер в приложении C#, можно использовать метод Start(). Чтобы остановить таймер, нужно вызвать метод Stop().
   3. Используется для хранения изображений, которые затем будут отображаться элементами управления.
   4. Некоторые режимы изменения размеров изображения (SizeMode) у PictureBox:  
      Normal; StretchImage; AutoSize.
   5. Чтобы программно создать и добавить элементы на форму в приложении C#, можно использовать коллекцию Controls формы.
2. **Вывод:**
   1. Научились программно создавать и настраивать элементы управления в приложениях на C#;
   2. Научились использовать таймер в приложениях на C#.

**Лабораторная работа №35**

**Разработка многооконного приложения**

**1 Цель работы:**

* 1. Научиться создавать различные типы многооконных приложений на C#.

1. **Ответы на контрольные вопросы:**
   1. Обычные, диалоговые, mdi
   2. MDI-приложение на языке C# — это способ отображения формы Windows, где есть по крайней мере одно родительское и множество дочерних окон.
   3. При ShowDialog(); блокируются другие окна
   4. Используется MessageBox;
2. **Вывод:**
   1. Научились создавать различные типы многооконных приложений на C#.

**Лабораторная работа №36**

**Разработка функциональной схемы работы приложения**

**1 Цель работы:**

* 1. Научиться разрабатывать функциональную схему работы приложения.

1. **Ответы на контрольные вопросы:**
   1. Функциональная схема приложения определяется через анализ функций, которые должно выполнять приложение
   2. Условия, действия, компоненты
   3. Требуется создать обработчик события.
2. Вывод:
   1. Научились разрабатывать функциональную схему работы приложения.

**Лабораторная работа №37**

**Разработка игрового приложения**

**1 Цель работы:**

* 1. Научиться разрабатывать интерактивные игровые приложения на C#.

1. **Ответы на контрольные вопросы:**
   1. KeyDown, KeyPress, KeyUp
   2. Click, MouseHover и т.д.
   3. IntersectsWith() используется для проверки пересечения между двумя элементами, например, для обнаружения столкновения между двумя объектами.  
      Intersect() применяется для нахождения общих элементов между двумя наборами данных. Он возвращает новое множество, содержащее элементы, которые присутствуют и в текущем множестве, и в множестве, переданном в качестве аргумента.
   4. Add и Remove
2. **Вывод:**
   1. Научились разрабатывать интерактивные игровые приложения на C#.

**Лабораторная работа №38**

**Разработка приложений**

**1 Цель работы:**

* 1. Научиться разрабатывать приложения на C#.

1. **Ответы на контрольные вопросы:**
   1. постановка задачи, разработка алгоритма, программирование, отладка, тестирование
   2. try, catch, finally
   3. buttons и другие
   4. для выявления его ошибок, багов, лагов и глюков.
2. **Вывод:**
   1. Научились разрабатывать приложения на C#.