Московский государственный технический университет

имени Н.Э. Баумана

Факультет Информатика, искусственный интеллект и системы управления

Кафедра Компьютерные системы и сети (ИУ-6)

Г.С. Иванова, Т.Н. Ничушкина

**Создание консольных приложений**

**в среде Lazarus**

.

Методические указания по выполнению лабораторной работы № 1

по дисциплине Основы программирования

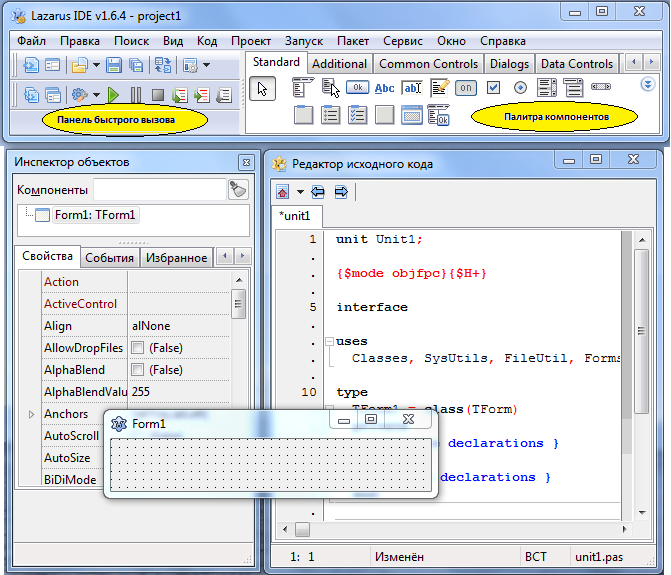
Москва 2021

# Часть 1 Создание программ в консольном режиме

Интегрированная среда программирования ***Lazarus*** — [открытая](https://ru.wikipedia.org/wiki/%D0%9E%D1%82%D0%BA%D1%80%D1%8B%D1%82%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%B5_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D0%B5) [среда разработки программного обеспечения](https://ru.wikipedia.org/wiki/%D0%A1%D1%80%D0%B5%D0%B4%D0%B0_%D1%80%D0%B0%D0%B7%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D0%BA%D0%B8_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%B3%D0%BE_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D1%8F) на языке [***Object Pascal***](https://ru.wikipedia.org/wiki/Object_Pascal) для компилятора ***[Free Pascal](https://ru.wikipedia.org/wiki/Free_Pascal" \o "Free Pascal)***. Эта среда реализована для нескольких платформ: [***Linux***](https://ru.wikipedia.org/wiki/Linux), ***[FreeBSD](https://ru.wikipedia.org/wiki/FreeBSD" \o "FreeBSD)***, ***[Mac OS X](https://ru.wikipedia.org/wiki/Mac_OS_X" \o "Mac OS X)***, ***[Microsoft Windows](https://ru.wikipedia.org/wiki/Microsoft_Windows" \o "Microsoft Windows)***, ***[Android](https://ru.wikipedia.org/wiki/Android" \o "Android)*** и относится к классу визуальных, в которых разработчику предоставляется возможность прямо на экране формировать интерфейс разрабатываемого программного продукта из стандартных элементов управления.

При этом среда позволяет создавать не только приложения c графическим интерфейсом, но и консольные приложения. *Консольным приложением* называется программный продукт, который для вывода результатов использует специальное окно Консоль или Терминал. При выводе в это окно не применяются стандартные графические средства организации интерфейсов, а потому создание консольных приложений существенно проще, чем других приложений.

При запуске среда *Lazarus* открывает несколько окон (рисунок 1).



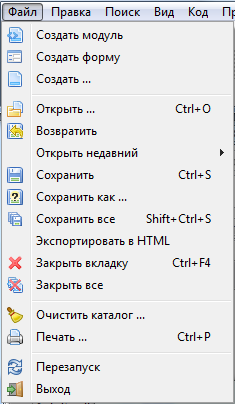
**Рисунок 1** – Вид главного и дочерних окон при входе в среду *Lazarus*

Главное окно содержит заголовок, основное меню, панель инструментов быстрого вызова и палитру визуальных и невизуальных компонентов. Отдельно открываются дочерние окна Инспектор объектов, Редактор исходного кода и формы заготовки приложения с графическим интерфейсом.

Все дочерние окна размещаются на экране и управляются (открываются, закрываются, перемещаются, меняют размеры и т.п.) независимо от главного окна. Помимо перечисленных дочерних окон пользователь имеет возможность с помощью меню открывать дополнительные окна, например, окна отладки.

Работа в среде осуществляется с использованием основного и контекстных меню, а также кнопок панели инструментов быстрого вызова. Эти кнопки применяют для упрощения доступа к часто выполняемым операциям (они дублируют соответствующие пункты меню).

Первый пункт меню **Файл** контролирует создание, открытие и чтение файлов – программ.

Он со­дер­жит следующие основные подпункты (рисунок 2):

**Создать модуль** – создание модуля проекта;

**Создать форму** – создание новой формы проекта;

**Создать...** – создание новых проектов, файлов и т. п.;

**Открыть** – открытие уже существующих проектов и других файлов;

**Открыть недавний** – открытие недавно созданных проектов;

**Сохранить** – сохранение текущего файла на том же мес­те;

**Сохранить** как**…** – сохранение текущего файла с новым именем или на новом месте;

**Закрыть вкладку** – закрытие вкладки текущего файла в редакторе;

**Закрыть все** – закрытие всех открытых файлов, включая проект.

**Рисунок 2 –** Пункт меню **Файл**

Выход – выход из среды.

Пункт меню **Правка** позволяет выполнить основные операции с текстом программы: удаление, копирование через буфер, вставку и т. п. Пункт меню **Поиск** содержит операции поиска в текущем файле и всех файлах проекта. Через меню пункта меню **Вид** осуществляется управление внешним видом окна среды. Пункт меню **Проект** объединяет операции работы с проектом (программой). Команды пункта **Запуск** осуществляют запуск и отладку программы.

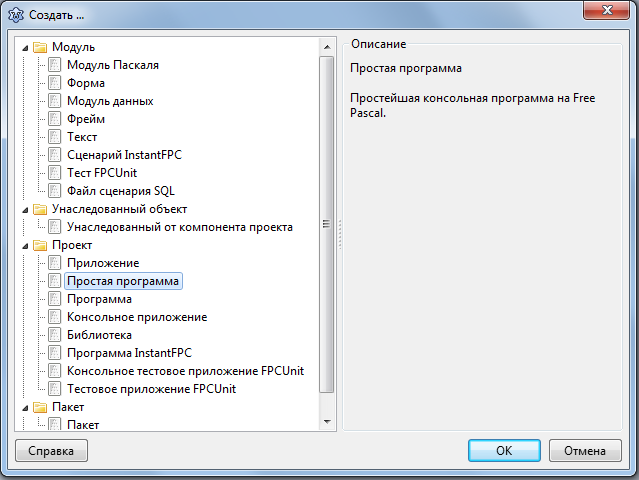
*Примечание* – Консольное приложение в простейших случаях не работает с компонентами среды *Lazarus*, поэтому окно инспектора объектов лучше закрыть, чтобы не отвлекало внимание.

**Задание 1**

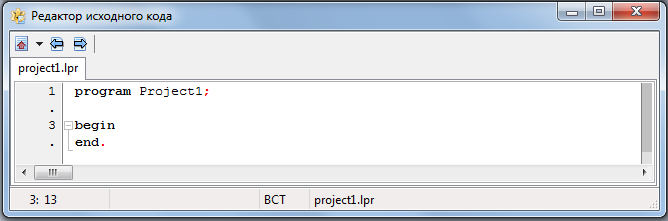
**Создать консольное приложение для вычисления корней квадратного уравнения.**

Порядок работы:

1. Для создания заготовки консольного приложения выберите пункт **Создать ... .** На экране появляется окно выбора создаваемых файлов в том числе проектов. В этом окне выберите подпункт **Проект** и далее подподпункт **Простая программа** (рисунок 3).

**Рисунок 3** *–* Выбор шаблона простейшего консольного приложения

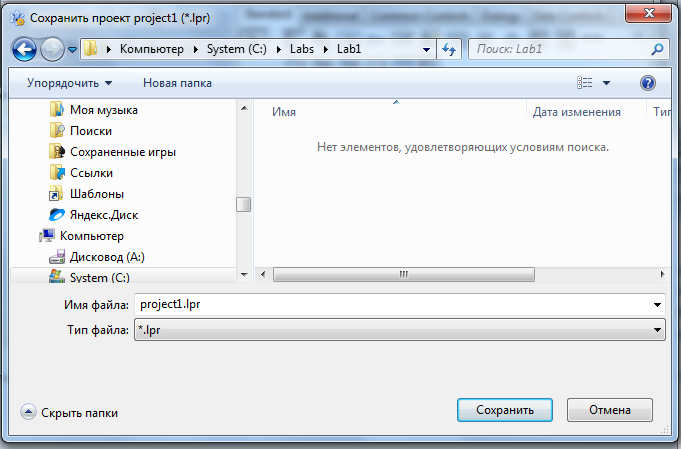
После этого на экране появится заготовка консольного приложения со стандартным именем ***Project1*** (рисунок 4). Автоматически созданный при этом файл имеет расширение ***.lpr***.

****

**Рисунок 4** – Вид окна редактора с заготовкой консольного приложения

2. Созданный файл проекта следует сохранить. Для удобства работы с создаваемыми программами каждый отдельный проект рекомендуется сохранять *в* *отдельной папке*. При этом целесообразно организовать отдельное место для хранения всех лабораторных работ и домашних заданий по дисциплине, например, создать папку ***labs*** и уже в ней папку ***lab1*** для хранения файлов лабораторной работы № 1.

Затем используйте пункт меню **Сохранить как...**. На экране появится диалог сохранения проекта (рисунок 5).



**Рисунок5** *–* Внешний вид диалога сохранения проекта

3. Введите в открытое окно редактора программ выделенный текст:

Program Project1;

**uses SysUtils;**

**Var A,B,C,D,E,X1,X2:Single;**

Begin **WriteLn('Input A, B, C:');**

**ReadLn(A,B,C);**

**D:= Sqr(B)- 4\*A\*C;**

**if D>=0 then**

**begin E:=2\*A;**

**X1:= (-B+Sqrt(D))/E;X2:= (-B-Sqrt(D))/E;**

**WriteLn('X1=', X1:6:1, ' X2=', X2:6:1)**

**end**

**else WriteLn('No result');**

**ReadLn;**

end.

В результате вы получите текст программы с подсветкой, принятой в Lazarus (рисунок 6).

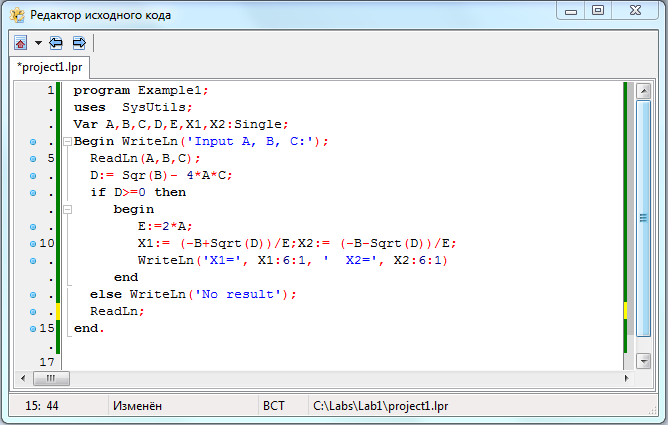


Рисунок 6 – Текст программы в окне Редактора исходного кода

4. Для работы с текстом программы удобно использовать окно обозревателя кода (рисунок 7).

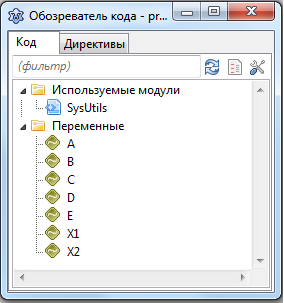


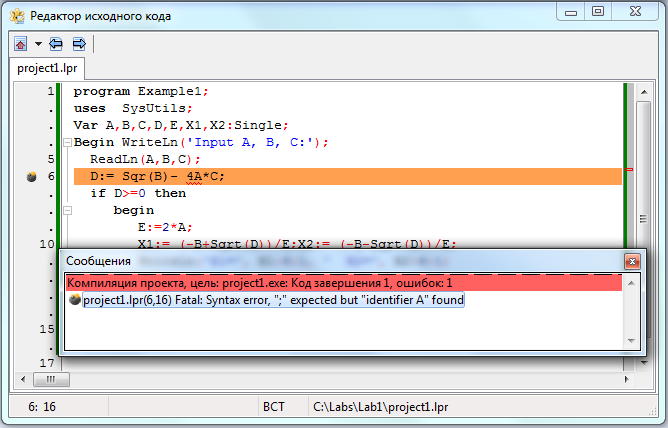
Рисунок 7 – Внешний вид окна обозревателя кода

Используя ссылки в этом окне, вы можете получить доступ ко всем модулям программы, ее переменным и другим элементам.

Для открытия окна обозревателя используется пункт меню **Вид/Обозреватель** кода (пункт меню Вид, подпункт – Обозреватель кода).

5. Для выполнения программы следует использовать кнопку быстрого вызова **Запуск** (зеленый треугольник на панели инструментов) или пункт меню **Запуск/Запустить** иликлавишу ***F*9**.

Если при вводе программы вы допустили синтаксические ошибки, то компилятор выдаст соответствующие сообщения в окне **Сообщения**. При этом курсор будет установлен в тексте программы на месте первой обнаруженной компилятором ошибки (рисунок 8).

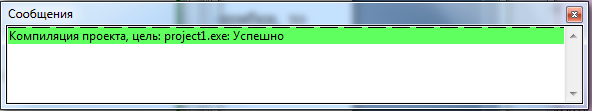


**Рисунок 8** – Вывод сообщения об ошибке компиляции

Перейти на строку, содержащую следующую ошибку, можно щелкнув мышью по соответствующему сообщению в окне Сообщения.

После исправления ошибок вновь запустите программу.

Если при вводе программы вы не допустили ошибок, то при ее запуске в окне Сообщения получите подтверждение успешной компиляции (рисунок 9).



**Рисунок 9** – Сообщение об успешном выполнении компиляции программы

После этого на экране появится окно Консоль или Терминал, в которое будет выведен запрос на ввод чисел. Числа вводятся через пробел или каждое число в своей строке, нажимая после ввода числа клавишу *Enter*. В результате выполнения программы вы должны получить ответ (рисунок 10).

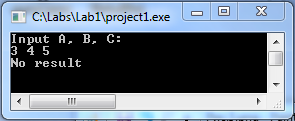


Рисунок 10 – Вид окна с результатом вычислений

**Задание 2**

**Изучить диагностические сообщения Turbo Delphi.**

Поочередно внося ошибки в программу, фиксируйте сообщения об ошибках в специальной таблице, представленной ниже. Классифицируйте ошибку, расшифруйте сообщение системы и определите этап выполнения программы (компиляция, компоновка или выполнение), на котором была обнаружена данная ошибка.

**Таблица –** Диагностические сообщения

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Ошибка | Физический смысл ошибки | Проявление  ошибки | Расшифровка сообщения | Этап |
| 1. | *Var AB, C,*  *X1, X2, D, E;* | Вместо переменных A и B описана переменная AB | Получено сообщение: Identifier not found «A» и др. | Не найдена переменная A | Компиля-ция |
| 2. | Readln(A,B,C) |  |  |  |  |
| 3. | E=2\*A; |  |  |  |  |
| 4. | *D:=spr(B)-4\*A\*C;* | Вызвана несуществующая функция |  |  |  |
| 5. | Исходные  данные: 0 1 3 |  |  |  |  |
| 6. | Исходные  данные: 1 1 3 |  |  |  |  |
| 7. | { *E:=2\*A;* } | Пропущена строка вычислений |  |  |  |
| 8. | *Read(A,C);* | Не определено (не введено) значение B |  |  |  |

В процессе работы в среде создаются сле­ду­ю­щие файлы и каталоги:

<Имя проекта>**.*lpr*** – исходный файл;

<Имя проекта>**.*lpi*, *lps* –** файлы описания проекта;

<Имя проекта>**.*exe*** – исполняемый файл (для *Windows*);

Кроме этого создается папка ***lib*** с вложенной папкой ***x86\_64-win64***, в которой появятся файлы:

<Имя проекта>**.*compiled* –** настройки компилятора;

<Имя проекта>**.*o*** –объектный модуль – результат компиляции.

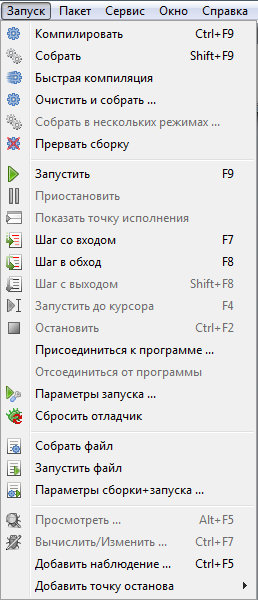
И папка ***backup*,** которая будет хранить копии измененных файлов с расширением ***.bak***.

Если в процессе работы создаются модули (*Unit*), то они будут иметь имя с расширением **.*pas***.

Для переноса консольной программы с машины на машину достаточно переписать на носитель файлы с расширениями *.****lpr*** и ***.pas*** (если они создавались), остальные будут пересозданы в процессе загрузки и запуска приложения**.**

# Часть 2. Средства отладки среды *Lazarus*

Компилятор и компоновщик находят не все ошибки программы. После них в программе могут остаться логические ошибки, ошибки в записи формул и т.д. Для локализации этих ошибок обычно осуществляют пошаговое выполнение программы с начала или с заданной точки с одновременным контролем значений всех или выбранных переменных.

**Управление пошаговым выполнением программы** выполняется с помощью подпунктов меню **Запуск** (рисунок 11):

**Шаг со входом** (***F*7**) – зайти в подпрограмму, если она вызывается на данном шаге;

**Шаг в обход** (***F*8**) – выполнить шаг, не заходя в подпрограмму, вызываемую на данном шаге (если на выполняемом шаге подпрограммы не вызываются, то Шаг со входом и Шаг в обход неразличимы);

**Шаг с выходом** – завершить выполнение под­программы;

**Остановить** (***Ctrl+F*2**) – прекратить процесс отладки.

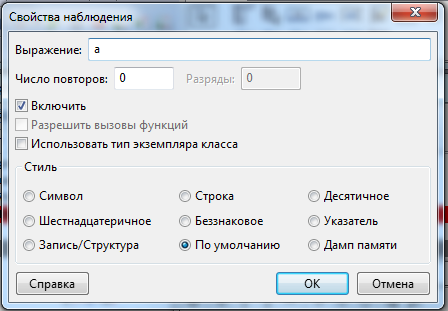
Для входа в пошаговый режим также можно использовать подпункт меню **Запустить до курсора** (***F*4**), при выборе которого программа выполняется до строки, в которой установлен курсор.

**Просмотр значений переменных в окне Список наблюдения.** В режиме пошагового выполнения существует возможность просмотра значений переменных. Это позволяет контролировать процесс выполнения программы.

Для открытия окна **Список наблюдений** используют подпункт меню **Вид/Окна отладки/Окно наблюдений**. Открывшееся окно не будет содержать имен переменных, значение которых должно высвечиваться в этом окне.

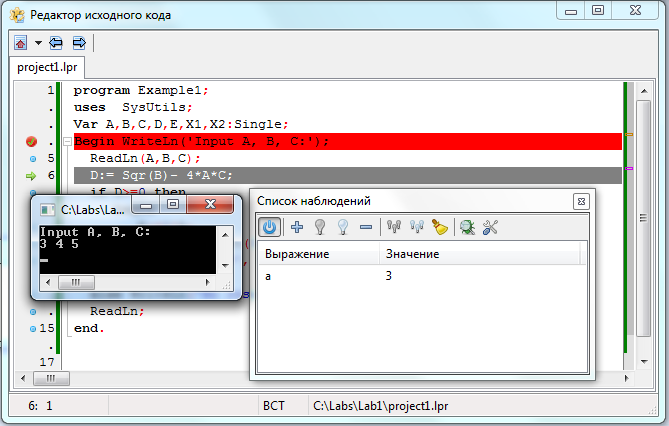
**Рисунок 11 –** Пункт меню **Запуск**

Для добавления имени переменной в окно наблюдения можно вызвать контекстное меню, щелкнув правой кнопкой мыши в области окна, или использовать инструмент «+» на локальной панели инструментов окна. этом на экране появляется окно **Свойства наблюдения** для настройки. В этом окне вводим имя переменной, например, «a» (рисунок 12). Тип переменной среда определяет автоматически, однако, если переменную необходимо показывать в каком-либо особом виде, например, шестнадцатеричном, то тип следует определить, поставив точку напротив выбранного среди указанных в окне настройки типов.



**Рисунок 12** – Добавление отслеживаемой переменной в окно наблюдений

Теперь при остановке в любой точке программы в окне Список наблюденийможно видеть значение переменной «a» в этот момент (рисунок 13).



**Рисунок 13** – Использование окна Список наблюдений в процессе отладки программы

В процессе отладки значения переменных просчитывают вручную и сравнивают со значениями, по­лу­ча­емыми в процессе работы программы.

**Установка контрольных точек.** Для установки контрольных точек необходимо щелкнуть мышью по левому полю окна программы напротив оператора, перед выполнением которого необходимо остановить процесс вычислений. В этом место появляется точка, а строка оператора выделяется красным (см. рисунок 13).

Теперь, если запустить программу, то ее выполнение остановится в указанной точке. Далее для поиска ошибки используют управление по­шаговым выполнением, описанное в предыдущем пункте. Для отмены точки останова необходимо повторно щелкнуть по той же точ­ке.

**Просмотр текущих значений переменных.** Текущие значения пере­менных во время выполнения программы можно просматривать, указывая во время остановки на соответствующую переменную курсором (рисунок 14).

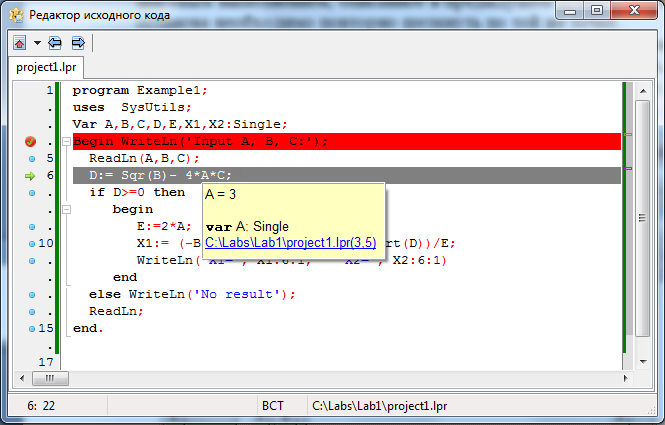


Рисунок 14 – Просмотр значений переменных, указанных курсором

**Задание 3**

**Изучить средства отладки программ в среде *Lazarus*.**

Порядок работы:

1. Внесите в программу ошибку 7. Выполните пошаговую трассировку программы, отслеживая значения переменных *A*, *B*, *C*, *D*, *E*, *x*1, *x*2 в окне Список наблюдений. Какое значение имеет переменная E в момент вычисления корней уравнения? Почему?
2. Внесите в программу ошибку 6. Выполните пошаговую трассировку программы, отслеживая значение переменных *A*, *B*, *C*, *D*, *E*, *x*1, *x*2. Какое значение имеет переменная *D*? В какой момент обнаруживается ошибка?
3. Установите точку останова перед вычислением дискриминанта. Выполните программу до точки останова. Просмотрите значения переменных, подводя к ним курсор мыши.

3. Ответы на вопросы и выводы занести в отчет.

**Часть 3. Создание схем алгоритмов средствами Microsoft Visio и OpenOffice Draw**

Теоретический материал и задания по третьей части лабораторной работы см. в методических указаниях Р.С. Самарев. Создание схем алгоритмов средствами Microsoft Visio и OpenOffice Draw.

**Приложение А**

**Правила поведения во время выполнения лабораторных работ по дисциплине «Основы программирования»**

1. Посещение лабораторных работ является обязательным. В компьютерный зал не допускаются студенты в верхней одежде, с едой и напитками.

1.1. В случае пропуска лабораторной работы по уважительной причине студент обязан предъявить оправдательный документ (справку о болезни, повестку в военкомат и т. д.)

1.2. Разрешается пропуск занятий студентам, которые сдали программы и отчеты по всем предыдущим лабораторным работам, включая пропускаемую.

2. Во время занятий студент выполняет лабораторную работу, составляет отчет и показывает работающую программу и отчет преподавателю. Не разрешается в учебное время ходить по залу, громко разговаривать и играть в компьютерные игры.

3. Студенты, не получившие зачет по всем лабораторным работам, к сдаче экзамена не допускаются. Студентам, сдающим лабораторные работы после срока по неуважительной причине, зачет задачи по результатам рейтинга аннулируется, а оценка за экзамен снижается.

4. Студенты, не выполняющие настоящие правила, от занятий отстраняются и будут допущены до следующих занятий только при наличии объяснительной записки с визой заместителя декана I курса.

**Требования к отчетам по практике**

1. Задание по лабораторной работе считается выполненным, если преподаватель принял программу и отчет.
2. Отчет должен выполняться в электронном виде с использованием текстовых и графических редакторов.
3. Каждый отчет должен включать титульный лист, где указывается:

а) фамилия, имя и номер группы студента;

б) номер варианта;

в) номер и название лабораторной работы;

г) фамилия, имя и отчество преподавателя.

1. Отчет по лабораторной 1 должны содержать все результаты практической работы
2. Начиная со 2-й лабораторной работы, отчет, кроме указанного ранее, должен содержать **(строго обязательно):**

a) текст задания;

б) схему алгоритма, выполненную в графическом редакторе;

в) текст программы;

г) результаты тестирования;

д) вывод.

Результаты тестирования должны быть оформлены в виде таблицы вида.

|  |  |  |
| --- | --- | --- |
| Исходные данные | Ожидаемый результат | Полученный результат |
|  |  |  |

6. Если студент имеет более 2-х несданных работ, то следующие лабораторные у него приниматься не будут, пока предыдущие лабораторные работы не будут сданы, а отчеты по ним – подписаны.