МИНИСТЕРСТВО ТРАНСПОРТА РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ АГЕНТСТВО ЖЕЛЕЗНОДОРОЖНОГО ТРАНСПОРТА
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
САМАРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ПУТЕЙ СООБЩЕНИЯ

Приложение 2
к рабочей программе дисциплины

ОЦЕНОЧНЫЕ МАТЕРИАЛЫ ДЛЯ ПРОМЕЖУТОЧНОЙ АТТЕСТАЦИИ

ПО ДИСЦИПЛИНЕ (МОДУЛЮ)

Проектирование пользовательского интерфейса
(наименование дисциплины(модуля)

Направление подготовки / специальность

09.03.03 Прикладная информатика
(код и наименование)

Направленность (профиль)/специализация

Прикладная информатика на железнодорожном транспорте
(наименование)

Содержание

1. Пояснительная записка

Цель промежуточной аттестации – оценивание промежуточных и окончательных результатов обучения по дисциплине, обеспечивающих достижение планируемых результатов освоения образовательной программы.

Перечень компетенций, формируемых в процессе освоения дисциплины

Код и наименование компетенции	Код индикатора достижения компетенции
ПК-2.2 Применяет методы и средства проектирования программного обеспечения, структур данных, баз данных, программных интерфейсов	Знает:принципы проектирования структуры базы данных, которая удовлетворяет требованиямфункциональности АИС современные технологии разработки приложений базы данных, программных интерфейсов, структур данных
	Умеет: использовать современные инструментальные средства и технологии программированияразрабатывать функциональность автоматизированной информационной системыразрабатывать программные компоненты для работы с базами данныхразрабатывать пользовательский интерфейс автоматизированной информационной системы
	Владеет: навыками работы с различными СУБД и их администрированияметодами проектирования структуры базы данныхтехнологией ADO .NET иEntityFramework для доступа к базе данных различных СУБДнавыками создания web-приложений базы данных на основе шаблона проектирования MVC.

Результаты обучения по дисциплине, соотнесенные с планируемыми результатами освоения образовательной программы

Код и наименование индикатора достижения компетенции	Результаты обучения по дисциплине	Оценочные материалы
ПК-2.2 Применяет методы и средства проектирования программного обеспечения, структур данных, баз данных, программных интерфейсов	Знаетпринципы проектирования структуры базы данных, которая удовлетворяет требованиямфункциональности АИС современные технологии разработки приложений базы данных, программных интерфейсов, структур данных	
	Умеет: использовать современные инструментальные средства и технологии программированияразрабатывать функциональность автоматизированной информационной системыразрабатывать программные компоненты для работы с базами данныхразрабатывать пользовательский интерфейс автоматизированной информационной системы	
	Владеет: навыками работы с различными СУБД и их администрированияметодами проектирования структуры базы данныхтехнологией ADO .NET иEntityFramework для доступа к базе данных различных СУБДнавыками создания web-приложений базы данных на основе шаблона проектирования MVC.	

Промежуточная аттестация (зачет) проводится в одной из следующих форм:
1) собеседование;
2) выполнение заданий в ЭИОС СамГУПС.

2. Типовые[1] контрольные задания или иные материалы для оценки знаний, умений, навыков и (или) опыта деятельности, характеризующих уровень сформированности компетенций

2.1 Типовые задания для оценки навыкового образовательного результата

Проверяемый образовательный результат

Код и наименование индикатора достижения компетенции	Образовательный результат
ПК-2.2	Применяет методы и средства проектирования программного обеспечения, структур данных, баз данных, программных интерфейсов

ОРГАНИЗАЦИЯ ИНТЕРФЕЙСА ПОЛЬЗОВАТЕЛЯ

1. Определение пользовательского интерфейса

На ранних этапах развития вычислительной техники пользовательский интерфейс рассматривался как средство общения человека с операционной системой и был достаточно примитивным. В основном он позволял запустить задание на выполнение, связать с ним конкретные данные и выполнить некоторые процедуры обслуживания вычислительной установки.

Со временем по мере совершенствования аппаратных средств появилась возможность создания интерактивного программного обеспечения, использующего специальные пользовательские интерфейсы. В настоящее время основной проблемой является разработка интерактивных интерфейсов к сложным программным продуктам, рассчитанным использование непрофессиональными пользователями. В последние годы были сформулированы основные концепции построения таких пользовательских интерфейсов и предложено несколько методик их создания.

Пользовательский интерфейс представляет собой совокупность программных и аппаратных средств, обеспечивающих взаимодействие пользователя с компьютером. Основу такого взаимодействия составляют диалоги. Под диалогом в данном случае понимают регламентированный обмен информацией между человеком и компьютером, направленный на решение конкретной задачи.

Обмен информацией осуществляется передачей сообщений и управляющих сигналов. Сообщение – порция информации, участвующая в диалоговом обмене. По направлению передачи информации различают:

- входные сообщения, которые генерируются человеком с помощью средств ввода (клавиатуры, мыши и т.п.);
- выходные сообщения, которые генерируются компьютером в виде текстов, звуковых сигналов и/или изображений и выводятся пользователю на экран монитора или другие устройства вывода информации.

В основном пользователь генерирует сообщения следующих типов: запрос информации, запрос помощи, запрос операции или функции, ввод или изменение информации и т.д. В ответ он получает: подсказки или справки, информационные сообщения, не требующие ответа, приказы, требующие действий, сообщения об ошибках, нуждающиеся в ответных действиях, и т.д.

2. Классификация интерфейсов

По аналогии с процедурным и объектным подходом к программированию различают процедурно-ориентированный и объектно-ориентированный подходы к разработке интерфейсов (см. рис. 1).

---

[1]Приводятся типовые вопросы и задания. Оценочные средства, предназначенные для проведения аттестационного мероприятия, хранятся на кафедре в достаточном для проведения оценочных процедур количестве вариантов. Оценочные средства подлежат актуализации с учетом развития науки, образования, культуры, экономики, техники, технологий и социальной сферы. Ответственность за нераспространение содержания оценочных средств среди обучающихся университета несут заведующий кафедрой и преподаватель – разработчик оценочных средств.

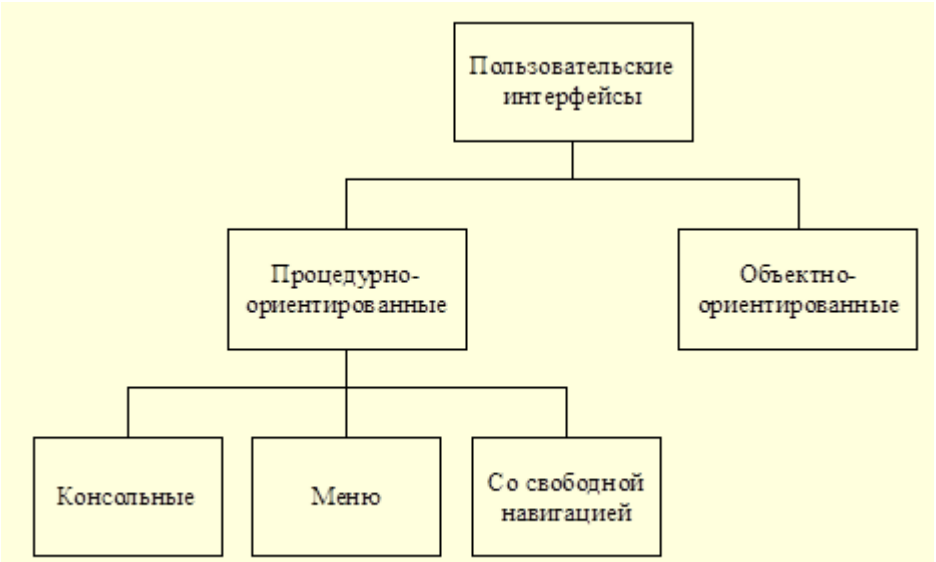

Рис.1 . Типы пользовательских интерфейсов

Процедурно-ориентированные интерфейсы предоставляют пользователю возможность выполнения некоторого набора действий, для которых могут вводиться соответствующие исходные данные. Вся работа с программой сводится к выбору действия, которое надо выполнить (если такой выбор предоставляется), вводу данных (при необходимости) и обработке полученных результатов.

Объектно-ориентированные интерфейсы используют несколько иную модель взаимодействия с пользователем, ориентированную на манипулирование объектами предметной области. Мы не будем подробно останавливаться на объектно-ориентированных пользовательских интерфейсах, поскольку для решения учебных задач достаточно процедурного подхода, значительно более простого в реализации. В качестве примера объектно-ориентированного интерфейса можно привести программу «Проводник» ОС Windows. Объектами предметной области в этом случае являются файлы и папки. Выполнение операции может выглядеть так: пользователь «берет» файл (точнее, объект интерфейса, соответствующий файлу) и «перетаскивает» его в другую папку, инициируя таким образом перемещение «физического» файла на диске.

Процедурно-ориентированные интерфейсы, в свою очередь, можно разделить на несколько подтипов: консольные, меню и со свободной навигацией.

Консольным называют интерфейс, который организует взаимодействие с пользователем на основе последовательного ввода и вывода информации в текстовом режиме по принципу «вопрос-ответ». Обычно такой интерфейс реализует конкретный сценарий работы, например: ввод данных – решение задачи – вывод результата (рис. 2, а). Единственное отклонение от последовательного процесса, которое обеспечивается данным интерфейсом, заключается в организации цикла для обработки нескольких наборов данных (рис. 2, б). Подобные интерфейсы в настоящее время используют только в процессе обучения программированию или в тех случаях, когда вся программа реализует одну функцию, например, в некоторых системных утилитах.

Рис. 2 . Структура программы с консольным интерфейсом

В качестве примера программы с консольным интерфейсом рассмотрим программу решения квадратного уравнения (см. рис. 3).

```
Введите коэффициент a: 4
Введите коэффициент b: 0
Введите коэффициент c: -1

Корни квадратного уравнения 4*x^2-1=0 :
x1=0.5
x2=-0.5
Решить еще одно уравнение? (y,n)_
```

Рис.3 . Внешний вид консольного интерфейса

Данная программа последовательно запрашивает коэффициенты уравнения, после чего производит вычисления, выводит результат и предлагает повторить ввод данных. Никаких средств перехода к предыдущему шагу не существует, и если пользователь ошибся при вводе параметров, то ему придется дойти до решения, а затем вводить параметры еще раз.

Интерфейс-меню, в отличие от консольного интерфейса, позволяет пользователю выбирать необходимые операции из специального списка, выводимого ему программой. В этом типе интерфейсов последовательность действий выбирается самим пользователем. Различают одноуровневые и иерархические меню. Первые используют для сравнительно простых случаев, когда вариантов немного (не более 5–7), и они включают операции одного типа, например, Создать, Открыть, Закрыть и т.п. Вторые применяются при большом количестве вариантов или их очевидных различиях, например, операции с файлами и операции с данными, хранящимися в этих файлах. На рис. 4 показана типичная структура алгоритма программы, организующей одноуровневое меню.

Алгоритм программы с многоуровневым меню обычно строится по уровням, причем выбор команды на каждом уровне осуществляется так же, как для одноуровневого меню.

Интерфейс-меню предполагает, что программа в любой момент времени находится либо в состоянии обслуживания меню (ожидания выбора со стороны пользователя), либо в состоянии

выполнения операции. Пользователь, как правило, вынужден ожидать, пока выполняется выбранное им действие.

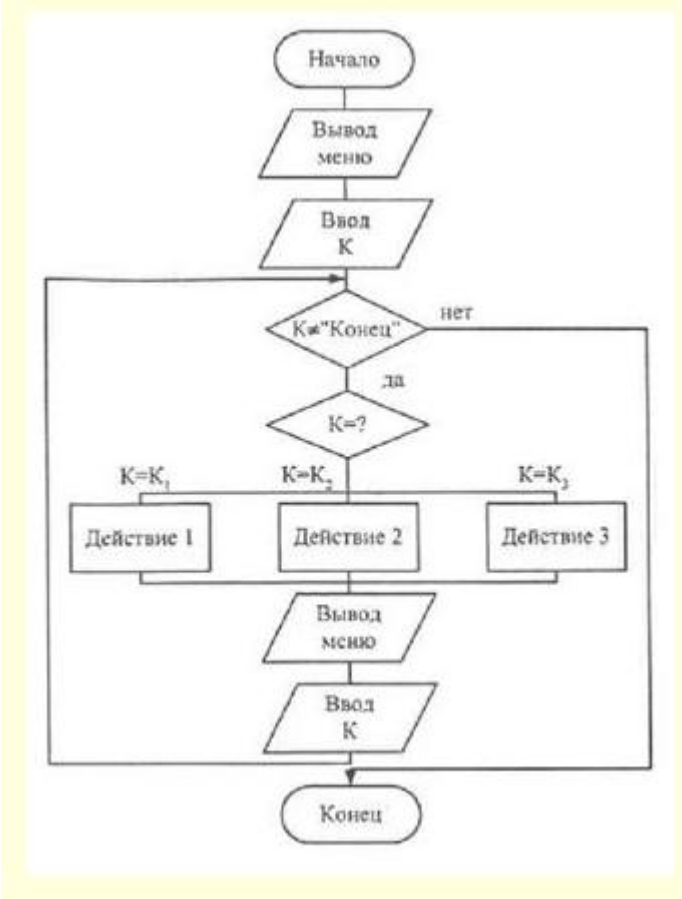

Рис.4 . Структура программы с интерфейсом-меню

Меню может быть построено различными способами. Простейший вариант реализации меню – вывод списка пунктов и предложение ввести номер пункта из этого списка (см. рис. 5, а). Более сложный вариант – список, по которому можно перемещаться с помощью клавиш (обычно клавиши управления курсором). Достоинства этого способа в том, что он удобнее, привлекательнее выглядит, не требует от пользователя соотнесения текста меню с номером пункта и уменьшает вероятность ошибки при выборе за счет того, что текущий пункт меню «подсвечивается». Внешний вид такого меню приведен на рис. 5, б.

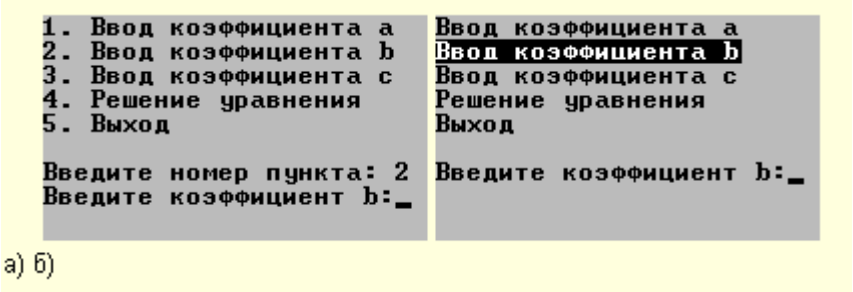

Рис.5 . Внешний вид интерфейса-меню

В отличие от интерфейса-меню интерфейс со свободной навигацией обеспечивает возможность осуществления любых допустимых в конкретном состоянии операций, доступ к которым возможен через различные интерфейсные компоненты. На данный момент сформировался стандартный набор компонент пользовательского интерфейса, которые широко применяются в самых разнообразных программах и поддерживаются многими операционными системами и библиотеками. Поскольку даже разные реализации этих компонент подчиняются

некоторым общим принципам управления, интерфейсы, построенные на их основе, привычны и понятны любому пользователю. Это является несомненным достоинством интерфейсов со свободной навигацией.

Внешний вид некоторых распространенных интерфейсных элементов в системе ОС Windows приведен на рис. 6. Перечислим эти компоненты (в скобках даны устоявшиеся английские названия):

- опция, флажок (checkbox), рис. 6, а;
- поле ввода (editbox), рис. 6, б;
- наборныйсчетчик (spin control, up/down control), рис. 6, в;
- кнопка (button), рис. 6, г;
- индикатор хода выполнения задачи (progressbar), рис. 6, д;
- ползунок (slider), рис. 6, е;
- списки: линейный (listbox, рис. 6, ж), выпадающий (combobox, рис. 6, з), древовидный (treecontrol, рис. 6, и);
- переключатель (radiobutton), рис. 6, к;
- меню (menu), рис. 6, л;
- панель инструментов (toolbar), рис. 6, м.

Существенной особенностью интерфейсов со свободной навигацией является способность изменяться в процессе взаимодействия с пользователем, предлагая выбор только тех операций, которые имеют смысл в конкретной ситуации (например, блокируя ввод в те или иные поля).

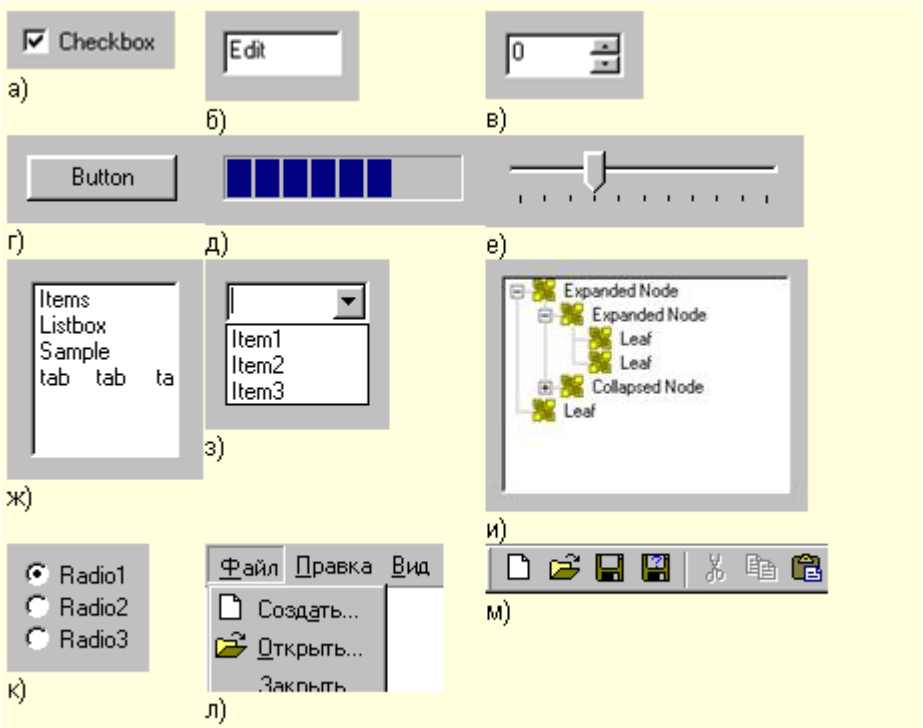

Рис.6 . Компоненты интерфейса со свободной навигацией

Рис.6 . Компоненты интерфейса со свободной навигацией

Как правило, интерфейсы этого типа реализуют, используя событийное программирование и объектно-ориентированные библиотеки, что предполагает применение визуальных сред разработки программного обеспечения. Тем не менее, несложные интерфейсы со свободной навигацией можно реализовать и на процедурно-ориентированном языке (например, Си) в однозадачной операционной системе без событийного управления (например, MS-DOS). Пример такого интерфейса для программы решения квадратного уравнения приведен на рис. 7.

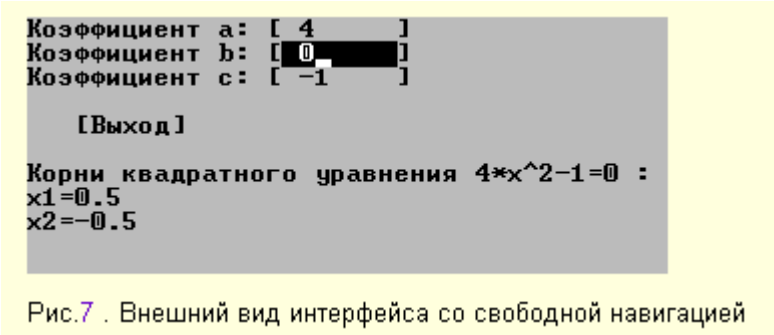

Рис.7 . Внешний вид интерфейса со свободной навигацией

Рис.7 . Внешний вид интерфейса со свободной навигацией

Интерфейс данной программы состоит из трех полей ввода для коэффициентов *a* , *b* , *c* и кнопки «Выход». Пользователь может вводить значения в поля в произвольном порядке. После ввода значения программа автоматически пересчитывает корни квадратного уравнения и обновляет информацию на экране. Текущий элемент управления (поле ввода или кнопка) подсвечивается, как это принято во всех системах, допускающих навигацию с использованием клавиатуры.

3. Текстовый режим работы видеоадаптера

Из предыдущих лекций Вам уже известно, что современные видеоадаптеры могут работать в различных видеорежимах, которые можно подразделить на текстовые и графические. Часть функций текстового вывода также уже рассмотрена. В основном эти функции рассчитаны на потоковый вывод информации и могут быть использованы как для вывода на экран, так и для вывода в файл. В данной лекции приводится описание функций, позволяющих работать с текстовой информацией на экране более сложным образом. Эти функции не могут применяться для вывода в файл, так как они используют особенности видеопамяти, предоставляющей произвольный доступ к содержимому (в отличие от потоковых механизмов работы с файлами).

Рассмотрим подробнее работу видеоадаптера в текстовом режиме. Экран в текстовом режиме разбивается на столбцы и строки символов. Количество столбцов и строк зависит от установленного видеорежима. Размеры экрана для стандартных видеорежимов приведены в табл. 1. По умолчанию программы под MS-DOS работают в цветном видеорежиме 80×25 (C80), однако при запуске их под WindowsNT/2000/XP система предлагает по умолчанию видеорежим 80×50 (C4350). Для обеспечения корректной работы программ нужно либо устанавливать режим самостоятельно (функцией textmode), либо определять размеры экрана (функцией gettextinfo) и осуществлять вывод с учетом полученных значений.

Помимо перечисленных, в современных видеоадаптерах существуют режимы с шириной экрана, равной 132 символа. В этих режимах на экран помещается значительно больше информации. В то же время качество вывода текста практически не ухудшается, так как современные мониторы поддерживают значительно большие разрешения и имеют большие размеры экрана, чем на заре развития ЭВМ. Однако эти режимы не поддерживаются BorlandC++, потому что они появились несколько позже соответствующих библиотек языка Си.

Таблица 1

Характеристики текстовых видеорежимов

Размеры экрана	Количество цветов	Константа Си
40×25	16 оттенков серого	BW40
40×25	16 цветов	C40
80×25	16 оттенков серого	BW80
80×25	16 цветов	C80
80×25	2 (монохромный)	MONO
80×43 (EGA) 80×50 (VGA)	16 цветов	C4350

Под один символ выделяется так называемое знакоместо – область экрана, находящаяся на пересечении строки и столбца. Все знакоместа имеют одинаковый размер и составляют прямоугольную матрицу. В одном знакоместе может быть записан только один символ.

На каждое знакоместо в видеопамяти отводится два байта. В один из них записывается ASCII-код символа, который должен выводиться в соответствующем месте экрана. Другой байт называется байтом атрибутов текста и содержит информацию о цвете символа. Младшие 4 бита атрибутов определяют цвет текста, старшие – цвет фона. Коды цветов приведены в табл. 2. Знакоместа записываются в видеопамяти последовательно слева направо, сверху вниз.

Таблица 2

Коды цветов текстового режима

Код	Цвет	Константа Си	Код	Цвет	Константа Си
0	Черный	BLACK	8	Черный	DARKGRAY
1	Темно-синий	BLUE	9	Светло-синий	LIGHTBLUE
2	Темно-зеленый	GREEN	10	Светло-зеленый	LIGHTGREEN
3	Темно-голубой	CYAN	11	Светло-голубой	LIGHTCYAN
4	Темно-красный	RED	12	Светло-красный	LIGHTRED
5	Темно-розовый	MAGENTA	13	Светло-розовый	LIGHTMAGENTA
6	Коричневый	BROWN	14	Желтый	YELLOW
7	Светло-серый	LIGHTGRAY	15	Белый	WHITE

Поскольку на один символ тратится всего два байта, текстовый режим является очень экономным с точки зрения расходования видеопамяти. Своеобразным побочным эффектом такой экономии является значительное упрощение функций работы с экраном и более высокое быстродействие этих функций. Особенно хорошо это заметно на старых компьютерах без аппаратного ускорения графики.

За формирование изображения на экране из ASCII-кодов и атрибутов текста отвечает видеоадаптер. Программисту не приходится затрачивать для этого практически никаких усилий. Преобразование ASCII-кода в растровый образ символа, выводимый на экран, осуществляется «на лету» в процессе развертки кадра. При этом используется шрифт, который либо «прошит» в ПЗУ видеоадаптера, либо заранее загружен из ОЗУ пользователем. Шрифты, используемые по умолчанию в системе MS-DOS, содержат во второй половине таблицы европейские символы. В связи с этим для вывода кириллицы в текстовом режиме требуется настройка системы на 866 кодовую страницу либо установка русификатора перед запуском программы. И в том, и в другом случае с диска подгружается русифицированный шрифт и устанавливается в качестве текущего шрифта для видеоадаптера.

Помимо символов, в текстовом режиме автоматически формируется изображение текстового курсора, который находится в заданной позиции на экране и имеет заданный внешний вид. Под внешним видом понимается высота курсора: он может выглядеть как черточка под символом, может покрывать все знакоместо или вообще быть отключен.

Система координат на экране выглядит так же, как и в графическом режиме: ось X направлена вправо, ось Y – вниз. Левый верхний угол имеет координаты (1; 1), в отличие от графического режима (где он имеет нулевые координаты). Во многих случаях текстовые функции Си работают не непосредственно с экраном, а с некоторой его прямоугольной областью, называемой окном вывода. По умолчанию окно занимает весь экран, но его координаты могут быть изменены программистом. При выводе в окно содержимое экрана за пределами окна не изменяется, а перенос слишком длинных строк производится по границе окна, а не экрана.

4. Функции текстового режима

Для использования описанных ниже функций необходимо подключить заголовочный файл <conio.h>.

4.1 Общиепараметры

voidtextmode(intnewmode)

Устанавливает заданный текстовый видеорежим (см. табл. 1).

void window(int left, int top, int right, int bottom)

Устанавливает новое текстовое окно. Задание некорректных координат приводит к игнорированию обращения к функции window. Параметры left и top задают координаты экрана

для верхнего левого угла окна, bottom и right определяют координаты экрана для нижнего правого угла окна.

По умолчанию окно занимает весь экран. Например, в режиме 80×25 по умолчанию координаты окна равны 1,1,80,25.

voidgettextinfo(structtext_info *pinfo)

Заполняет структуру text_info, на которую указывает pinfo, видеоинформацией о текущем режиме. С помощью этой функции, в частности, можно узнать текущие координаты окна, установленный видеорежим, размеры экрана.

4.2 Управление курсором

voidgotoxy(int x, int y)

Перемещает курсор в заданную позицию текстового окна. Левый верхний угол окна имеет координаты (1,1). Если координаты по какой-то причине указаны неверно, то вызов данной функции игнорируется. Примером такой ошибки может служить вызов gotoxy (40,30), когда размер окна равен 35×25.

intwherex(); intwherey()

Возвращают координаты текущей позиции курсора (относительно текстового окна).

void _setcursortype(intcur_t)

Задает внешний вид текстового курсора. С помощью этой функции можно отключить курсор (параметр _NOCURSOR), включить сплошной «блочный» курсор (_SOLIDCURSOR) или вернуть стандартный вид курсора (_NORMALCURSOR). Отключение курсора очень часто используется в интерфейсах-меню и интерфейсах со свободной навигацией.

4.3 Управление атрибутами текста

voidhighvideo(); voidlowvideo()

Устанавливают повышенную и пониженную (соответственно) яркость вывода символов. Эти функции не влияют на все символы, отображаемые в данный момент времени на экране. Их вызов влияет только на последующий вывод на консоль в текстовом режиме.

voidnormvideo()

Устанавливает нормальную яркость для символов путем возврата к тем значениям текстовых атрибутов (символов и фона), которые были в момент запуска программы.

voidtextcolor(intnewcolor)

Устанавливает цвет символов для выводимого на экран текста. В качестве аргумента можно передавать константы цвета, приведенные в табл. 2.

voidtextbackground(intnewcolor)

Устанавливает цвет фона для выводимого на экран текста. В качестве аргумента можно передавать константы цвета со значениями 0 ÷ 7, приведенные в табл. 2.

voidtextattr(intnewattr)

Устанавливает атрибуты выводимого текста. Эта функция дает возможность установить цвет фона и текста за один вызов. Для того чтобы сформировать байт атрибутов, необходимо цвет фона сдвинуть на 4 разряда влево и сложить с цветом текста: attr=back<<4+text.

4.4 Очистка текста

Все перечисленные ниже функции работают внутри текстового окна, не изменяя символы за его пределами.

voidclrscr()

Очищает текущее текстовое окно и устанавливает курсор в левый верхний угол (в позицию 1,1).

voidclreol()

Стирает все символы от позиции курсора до конца строки без перемещения курсора.

voiddelline(); voidinsline()

Функция delline удаляет текущую строку (в которой находится курсор) и поднимает все строки, находящиеся ниже курсора, на одну строку вверх. Функция insline вставляет пустую строку в позицию курсора текстового окна, используя при этом текущий цвет фона. Все строки, лежащие ниже данной, сдвигаются на одну строку вниз, а последняя строка в текстовом окне пропадает.

4.5 Копирование текста

Перечисленные далее функции работают без учета текущего окна. Все параметры задаются относительно левого верхнего угла экрана, имеющего координаты (1,1).

intmovetext(int left, int top, int right, int bottom, intdestleft, intdesttop)

Копирует содержимое прямоугольной области на экране, определяемой значениями left (левая граница), top (верхняя граница), right (правая граница) и bottom (нижняя граница), в новую прямоугольную область, определяемую аналогичным образом. Левый верхний угол нового прямоугольника задается парой параметров destleft и desttop. Копирование для перекрывающихся окон выполняется корректно.

intgettext(intleft, inttop, intright, intbottom, void *destin)

Заносит содержимое прямоугольной области на экране, заданной значениями параметров left, top, right, bottom в область памяти, на которую указывает destin.

Функция считывает содержимое прямоугольника в память последовательно, слева направо и сверху вниз. Каждая позиция экрана занимает 2 байта памяти. Первый байт соответствует символу данного знакоместа, а второй – его атрибутам.

Пространство, необходимое для прямоугольника в *w* столбцов шириной и *h* строк высотой определяется следующим образом:

размер в байтах = (*h* строк) × (*w* столбцов) × 2.

intputtext(int left, int top, int right, int bottom, void *source)

Выводит содержимое области памяти, на которую указывает source, в прямоугольник на экране, координаты которого задаются значениями left, top, right и bottom. Функция выводит содержимое области памяти в заданный прямоугольник в последовательности слева направо и сверху вниз.

5. Реализация пользовательского интерфейса в Borland C ++

5.1 Общие принципы

При внимательном рассмотрении перечисленных в разделе 2 видов пользовательского интерфейса можно заметить, что во всех случаях ввод и вывод информации производятся отдельно от основных вычислений в программе. Общий подход к проектированию приложений заключается в том, чтобы отделить интерфейсную часть от вычислительной. Вычислительная часть должна быть построена таким образом, чтобы как можно меньше зависеть от пользовательского интерфейса, от способов ввода и вывода информации. Все исходные данные передаются в вычислительное ядро стандартными методами, как правило, через параметры функций. Ядро возвращает результаты и, возможно, информацию о возникших ошибках. Интерфейсная часть самостоятельно интерпретирует эту информацию и сообщает пользователю результаты выполнения.

Вычислительная часть программы не должна самостоятельно выводить на экран какие-либо сообщения (исключение составляют сообщения, выводимые в отладочных целях, но их в финальной версии программы также быть не должно). Кроме того, она не должна завершать работу программы (за исключением ситуаций, когда нормальное продолжение работы в принципе невозможно, например, при сильных повреждениях служебных структур в памяти).

Для отделения одной части от другой в BorlandC++ их следует просто вынести в отдельные функции. Основное правило при распределении кода по функциям заключается в том, чтобы для каждой функции определить ее принадлежность к интерфейсу или вычислительной части. По возможности следует не допускать появления функций, которые по поведению можно отнести и в ту, и в другую часть.

Основное требование к интерфейсной части – обеспечивать максимальную защиту от ошибок ввода. Контроль сложных зависимостей между вводимыми данными, которые определяются типом выполняемой задачи, обычно выполняется ядром. В то же время интерфейс в состоянии отследить простые ошибки: ввод строки вместо числа, ввод числа за пределами заданного диапазона и т.п. Проверкам «на глупость» должны подвергаться все данные, вводимые пользователем.

5.2 Консольный интерфейс

Консольный интерфейс достаточно просто реализуется с помощью стандартных функций ввода-вывода: printf, scanf, puts, getch. Перед выводом можно очистить экран, чтобы предыдущие сеансы работы с программой не отвлекали пользователя. Для некоторых видов программ, особенно для системных утилит, очистка экрана является нежелательной, т.к.

информация о предыдущих запусках может потребоваться пользователю. Впрочем, к учебным программам это отношения не имеет, и экран лучше все-таки очистить.

Перед каждым запросом данных необходимо вывести пользователю приглашение для ввода этих данных, например: «Введите коэффициент a». Если ввод производится по сравнительно сложным правилам, их необходимо также вкратце описать перед вводом (например, при вводе массива нужно сначала спросить число элементов N, а потом указать, что элементы вводятся N раз).

Поскольку после завершения программы на экране появляется оболочка BorlandC++ и закрывает результаты расчетов, желательно приостанавливать выполнение программы до нажатия клавиши после вывода результатов. При этом пользователю нужно вывести соответствующее сообщение.

Кроме того, ввод данных можно осуществлять не стандартными функциями, а рассмотренными ниже функциями ввода с редактированием. Это позволит сделать интерфейс более дружественным пользователю, особенно в случае, когда расчеты можно повторить (во второй и последующих итерациях можно предлагать пользователю по умолчанию предыдущие значения параметров).

В соответствии с принципами разделения интерфейса и вычислительного ядра ввод, вывод и вычисления следует размещать в разных функциях. Общую логику работы «ввод – вычисления – вывод» можно также вынести в отдельную функцию:

```
void run()
{
input_data();
calculate();
show_results();
}
```

Разумеется, прототипы функций ввода input_data(), вычислений calculate() и вывода show_results() могут быть (и будут) другими, зависящими от конкретной программы и способов передачи данных в ядро. Такой подход позволяет легко организовать циклическое выполнение расчетов. Для этого функцию run() нужно вызывать в do…while-цикле, выход из которого осуществляется при отрицательном ответе на вопрос «продолжить?».

В остальном программирование консольного интерфейса, как правило, не вызывает затруднений.

5.3 Простое меню

Простое меню отличается от консольного интерфейса в первую очередь последовательностью выполнения операций. Программа обязательно зацикливается, условие выхода из цикла – выбор пункта меню «Выход». Внутри цикла размещается запрос номера меню и его обработка. Запрос может производиться функциями scanf, getch и т.п. Обработка обычно реализуется оператором switch, в котором для каждого пункта меню предусмотрена отдельная ветка case, а все остальные значения «тихо» игнорируются. Уведомлять пользователя о выборе неправильного пункта меню необязательно, это может его раздражать (например, при случайном промахе в процессе нажатия клавиши).

Основная интерфейсная функция может выглядеть так:

```
voidrun()
{
//... вывод меню на экран
intcont_menu_loop=1;
do
{
intsel_menu;
//... вводsel_menuлюбымудобнымспособом
switch (sel_menu)
{
case0: //вводзначения a
a=input_float();
break;
case1: //вводзначения b
```

```
b=input_float();
break;
//...
case4: //вычисления и вывод результатов
calculate();
show_results();
break;
case5: //выход
cont_menu_loop=0;
break;
}
} while (cont_menu_loop);
}
```

Если меню используется как часть более сложного пользовательского интерфейса, то вывод меню и выбор пункта можно вынести в отдельную функцию. Этой функции можно передавать массив строк с названиями отдельных пунктов, а возвращаемое значение (номер выбранного пункта) анализировать в операторе switch. Такой подход в дальнейшем позволяет легко перейти от меню одного типа к меню другого типа, например, от простого (см. рис. 5, а) к меню с перемещением курсора (см. рис. 5, б).

5.4 Меню с перемещением курсора

Основное отличие этого варианта меню от простого заключается в способе ввода переменной sel_menu. Если в простом меню она вводится непосредственно пользователем, то в данном варианте ввод осуществляется косвенно с помощью клавиш управления курсором. Выбор пункта можно организовать следующим образом:

```
voidhighlight_menu(intitem,inthighlight);
void run()
{
//...
int selected=0;
intsel_menu=0;
do
{
highlight_menu(sel_menu,1); // подсвечиваем пункт меню
int key=getch();
highlight_menu(sel_menu,0); // временно гасим пункт меню
switch (key)
{
case0:
key=getch();
switch (key)
{
case UP:
sel_menu=(sel_menu-1+max_menu)%max_menu;
break;
case DOWN:
sel_menu=(sel_menu+1)%max_menu;
break;
}
break;
case ENTER:
selected=1;
break;
}
} while (!selected);
//...
}
```

Вприведенномфрагментекодафункцияhighlight_menuвыводитэлементменюсзаданнымномеромсвключенной (highlight != 0) иливыключенной (highlight = = 0) подсветкой. Данную функцию можно реализовать различными способами. Как правило, подсветка производится с помощью установки соответствующих атрибутов текста или вызовом функций highvideo и lowvideo. В зависимости от реализации в highlight_menu может потребоваться передача дополнительных параметров, описывающих меню (например, массив строк с названиями пунктов, размеры меню на экране и т.п.).

5.5 Ввод строки с редактированием

Стандартные средства Си для ввода данных (функция scanf) предоставляют пользователю очень мало возможностей. Редактирование вводимого текста возможно только путем удаления символов с конца строки, начальное значение установить нельзя (по умолчанию вводится пустая строка), прокрутки строки, если она не влазит в окно, не предусмотрено (точнее, строка просто переносится по границе окна, что не всегда желательно).

В связи с этим очень часто возникает необходимость в разработке своих средств ввода текста, которые обеспечили бы удобный и привычный пользовательский интерфейс. При этом достаточно организовать ввод текстовой строки, а ввод чисел и более сложных значений (например, интервалов) организуется на основе текстового ввода с последующей обработкой введенной строки (например, чтением числа с помощью функции strtol).

Рассмотрим следующую функцию ввода текста с редактированием:

```
intinputstr(char* str,intmaxlen)
{
int x=wherex(),y=wherey(),curlen=strlen(str);
intpos=curlen;
charfirstkey=1;
highvideo();
cprintf("%-*s",maxlen,str);
lowvideo();
gotoxy(x+pos,y);
_setcursortype(_NORMALCURSOR);
intexitcode=-1;
while (exitcode<0)
{
int key=getch();
switch (key)
{
case 0:
key=getch();
switch (key)
{
case LEFT:
if (pos>0)
--pos;
break;
case RIGHT:
if (pos<curlen)
++pos;
break;
case DELETE:
if (pos<curlen)
movmem(str+pos+1,str+pos,curlen-pos);
break;
case HOME:
pos=0;
break;
case END:
pos=curlen;
```

```
break;
}
break;
case BACKSPACE:
if (pos>0)
{
--pos;
movmem(str+pos+1,str+pos,curlen-pos);
}
break;
case ESC:
exitcode=0;
break;
case ENTER:
exitcode=1;
break;
default:
if (key>=' ')
{
if (firstkey)
{
pos=0;
str[0]=0;
}
if (curlen<maxlen)
{
movmem(str+pos,str+pos+1,curlen-pos+1);
str[pos]=key;
++pos;
}
}
}
curlen=strlen(str);
firstkey=0;
gotoxy(x,y);
cprintf("%-*s",maxlen,str);
gotoxy(x+pos,y);
}
_setcursortype(_NOCURSOR);
gotoxy(x,y);
returnexitcode;
}
```

Данная функция организует ввод строки ограниченной длины с возможностью перемещения курсора клавишами Left, Right, Home, End и удаления символов клавишами Delete и Backspace. Ввод завершается при нажатии Enter либо Esc, при этом возвращаемое значение показывает, какая из клавиш была нажата. Это позволяет в вызывающей функции определить, отказался пользователь от ввода или подтвердил его. Если первая нажатая клавиша приводит к вводу символа, то старое содержимое строки очищается. Такая особенность позволяет легко вводить новые данные вместо старых. В то же время, если первой нажатой клавишей была клавиша редактирования, старое содержимое сохраняется.

Перед началом ввода функция устанавливает нормальный вид курсора, а после окончания ввода скрывает текстовый курсор. Это позволяет использовать функцию в интерфейсах-меню и интерфейсах со свободной навигацией, не прибегая к дополнительным вызовам _setcursortype.

При редактировании строки ее содержимое в буфере str перемещается функцией movmem, которая предназначена для копирования участка памяти. Для использования этой функции необходимо подключить заголовочный файл <mem.h>.

Приведенная функция inputstr может использоваться в качестве замены scanf, при этом интерфейс становится более дружественным, что благоприятно сказывается на впечатлении пользователя от программы. Для ввода чисел различных типов можно написать отдельные функции, преобразующие число в строку, вызывающие inputstr и затем преобразующие полученную строку обратно в число.

2.2. Перечень вопросов для подготовки обучающихся к промежуточной аттестации ЗАДАНИЕ (практическое) к зачету:

1. Дать определение понятиям: HCI, эргономика, функциономика. Перечислить факторы, значимые для HCI.
2. Датьопределениепонятиям:Эргономическоеобеспечение, юзабилити, интерфейс. Охарактеризовать виды интерфейсов и привести примеры.
3. Дать определение понятию: Интерфейс пользователя. Перечислитьосновныесоставляющиеинтерфейсапользователя. Перечислить видыинтерфейсовпользователя.
4. Датьопределение и охарактеризоватьграфические и командныеинтерфейсы. Привести примеры.
5. Датьопределение и охарактеризоватьинтерфейскомандной строки и многооконныйинтерфейс. Привести примеры.
6. Перечислить главные функции проектировщика интерфейсов.
7. Перечислить, дать определения и охарактеризовать познавательные процессы пользователя, наиболее значимые с точки зрения разработки интерфейса.
8. Перечислить, дать определения и охарактеризовать информационные процессы пользователя, наиболее значимые с точки зрения разработки интерфейса.
9. Перечислить, дать определения и охарактеризовать виды памяти пользователя.
10. Дать определение понятию: мышление.Охарактеризовать виды мышления. Привести примеры.
11. Охарактеризовать цвет как визуальный атрибут отображаемой информации. Охарактеризовать воздействие каждого из основных цветов на пользователя. Дать рекомендации, где лучше всего использовать тот или иной цвет.
12. Основные рекомендации по использованию цвета с точки зрения физиологии (по Мерчу).
13. Основные рекомендации по использованию цвета с точки зрения восприятия (по Мерчу).
14. Основные рекомендации по использованию цвета с точки зрения познавательности (по Мерчу).
15. Перечислить визуальные атрибуты отображаемой информации. Охарактеризовать основные действия разработчика интерфейса при проектировании размещения данных на экране. Описать общие принципы расположения информации на экране.
16. Охарактеризовать основные методы выделения информации.
17. Перечислить основные принципы композиции и организации информации.
18. Охарактеризовать иерархический принцип организации информации и принцип визуального выделения наиболее важных элементов.
19. Охарактеризовать принцип сбалансированности структуры экрана и принцип визуального объединения логически взаимосвязанных элементов.
20. Охарактеризовать принцип удобочитаемости, логической согласованности и интеграции.
21. Охарактеризовать шрифт как атрибут визуального отображения информации.
22. Основные способы пространственного размещения. Дать определения понятиям группирование, дискрета окна, выравнивание.
23. Охарактеризовать диалог типа “вопрос-ответ”. Привести примеры.
24. Охарактеризовать диалог на основе меню. Привести примеры.
25. Охарактеризовать диалог на основе экранных форм. Привести примеры.
26. Охарактеризовать диалог на основе командного языка. Описать принципы передачи параметров. Привести примеры.

27. Охарактеризовать процесс разработки сценария диалога, его цели. Перечислить основные способы представления сценариев.
28. Дать определения понятиям: процесс, задание. Охарактеризовать процессы ввода-вывода. Перечислить основные факторы, которые учитываются при описании процессов ввода-вывода.
29. Охарактеризовать сообщения как средства осуществления диалога. Перечислить и охарактеризовать типы сообщений.
30. Охарактеризовать методы разработки гибкого интерфейса. Перечислить и охарактеризовать виды адаптации.
31. Охарактеризовать темп ведения диалога. Дать определение понятиям: Время ответа (отклика) системы, клауза, закрытие. Дать рекомендации по допустимому времени ответа системы.
32. Перечислить и охарактеризовать модели построения интерфейса.
    Модели построения интерфейса.
33. Перечислить виды окон приложения и охарактеризовать их структуру.
    Окна. Их виды и структура.
34. Перечислить и охарактеризовать виды вторичных окон приложения.
35. Охарактеризовать основные достоинства и недостатки MDI-интерфейса.
36. Перечислить и охарактеризовать альтернативные средства технологии MDI.
37. Описание закона Фиттса.
38. Математическая модель закона Фиттса.
39. Правило размера цели как частное применение закона Фиттса.
40. Физическое и виртуальное позиционирование как частное применение закона Фиттса.
41. Правило бесконечной границы как частное применение закона Фиттса.
42. Закон Хика.
43. Пример выбора системы меню на основе вычислений по закону Хика. Количество меню и элементов меню взять самостоятельно.
44. Дать определение понятию “Элемент управления”. Перечислить и охарактеризовать возможные состояния элементов управления.
45. Охарактеризовать меню и выпадающие меню как элементы управления с точки зрения удобства использования. Привести примеры.
    Меню как элемент управления.
46. Охарактеризовать всплывающие и каскадные меню как элементы управления с точки зрения удобства использования. Привести примеры.
47. Основные требования к оформлению меню приложения.
48. Охарактеризовать кнопки, флажки и чекбоксы как элементы управления с точки зрения удобства использования. Привести примеры.
49. Охарактеризовать обычные, раскрывающиеся, пролистываемые списки как элементы управления с точки зрения удобства использования. Привести примеры.
50. Охарактеризовать поля ввода, ползунки, полосы прокрутки как элементы управления с точки зрения удобства использования. Привести примеры.
51. Охарактеризовать строку состояния, индикатор состояния процесса как элементы индикации с точки зрения удобства использования. Привести примеры.
52. Дать определение понятию “юзабилити-тестирование”. Перечислить и охарактеризовать основные способы проведения тестирования.
53. Перечислить и охарактеризовать факторы, определяющие удобство применения ПО.
54. Перечислить основные рекомендации по проведению тестирования.

Описать особенности свойств окна в LabVIEW (VIProperties) – представление окна, выполнение и пр.

3. Методические материалы, определяющие процедуру и критерии оценивания сформированности компетенций при проведении промежуточной аттестации

Критерии формирования оценок по ответам на вопросы, выполнению тестовых заданий

- оценка **«отлично»** выставляется обучающемуся, если количество правильных ответов на вопросы составляет 100 – 90% от общего объёма заданных вопросов;

- оценка **«хорошо»** выставляется обучающемуся, если количество правильных ответов на вопросы – 89 – 76% от общего объёма заданных вопросов;

- оценка **«удовлетворительно»** выставляется обучающемуся, если количество правильных ответов на тестовые вопросы –75–60 % от общего объёма заданных вопросов;

- оценка **«неудовлетворительно»** выставляется обучающемуся, если количество правильных ответов – менее 60% от общего объёма заданных вопросов.

Критерии формирования оценок по результатам выполнения заданий

«Отлично/зачтено» – ставится за работу, выполненную полностью без ошибок и недочетов.

«Хорошо/зачтено» – ставится за работу, выполненную полностью, но при наличии в ней не более одной негрубой ошибки и одного недочета, не более трех недочетов.

«Удовлетворительно/зачтено» – ставится за работу, если обучающийся правильно выполнил не менее 2/3 всей работы или допустил не более одной грубой ошибки и двух недочетов, не более одной грубой и одной негрубой ошибки, не более трех негрубых ошибок, одной негрубой ошибки и двух недочетов.

«Неудовлетворительно/не зачтено» – ставится за работу, если число ошибок и недочетов превысило норму для оценки «удовлетворительно» или правильно выполнено менее 2/3 всей работы.

Виды ошибок:

- грубые ошибки: незнание основных понятий, правил, норм; незнание приемов решения задач; ошибки, показывающие неправильное понимание условия предложенного задания.

- негрубые ошибки: неточности формулировок, определений; нерациональный выбор хода решения.

- недочеты: нерациональные приемы выполнения задания; отдельные погрешности в формулировке выводов; небрежное выполнение задания.

Критерии формирования оценок по зачету с оценкой

«Отлично/зачтено» – студент приобрел необходимые умения и навыки, продемонстрировал навык практического применения полученных знаний, не допустил логических и фактических ошибок

«Хорошо/зачтено» – студент приобрел необходимые умения и навыки, продемонстрировал навык практического применения полученных знаний; допустил незначительные ошибки и неточности.

«Удовлетворительно/зачтено» – студент допустил существенные ошибки.

«Неудовлетворительно/не зачтено» – студент демонстрирует фрагментарные знания изучаемого курса; отсутствуют необходимые умения и навыки, допущены грубые ошибки.

Экспертный лист
оценочных материалов для проведения промежуточной аттестации по дисциплине «Проектирование пользовательского интерфейса»

Направление подготовки / специальность

09.03.01. «Информатика и вычислительная техника»
(код и наименование)

Направленность (профиль)/специализация

_____
(наименование)

Бакалавр
квалификация выпускника

1. Формальное оценивание			
Показатели		Присутствуют	Отсутствуют
Наличие обязательных структурных элементов:		+	
–титульный лист		+	
–пояснительная записка		+	
– типовые оценочные материалы		+	
–методические материалы, определяющие процедуру и критерии оценивания		+	
Содержательное оценивание			
Показатели	Соответствует	Соответствует частично	Не соответствует
Соответствие требованиям ФГОС ВО к результатам освоения программы	+		
Соответствие требованиям ОПОП ВО к результатам освоения программы	+		
Ориентация на требования к трудовым функциям ПС (при наличии утвержденного ПС)	+		
Соответствует формируемым компетенциям, индикаторам достижения компетенций	+		

Заключение: ФОС рекомендуется/ не рекомендуется к внедрению; обеспечивает/ не обеспечивает объективность и достоверность результатов при проведении оценивания результатов обучения; критерии и показатели оценивания компетенций, шкалы оценивания обеспечивают/ не обеспечивают проведение всесторонней оценки результатов обучения.