МИНОБРНАУКИ РОССИИ
федеральное государственное бюджетное образовательное учреждение
высшего образования
«Кузбасский государственный технический университет имени Т. Ф. Горбачева»
Институт профессионального образования
Кафедра информационных и автоматизированных производственных систем

Составитель О.С. Семенова

ОП.04 Основы алгоритмизации и программирования

Методические указания к самостоятельной работе

Рекомендовано цикловой методической комиссией
информационных систем и программирования
в качестве электронного издания для использования
в образовательном процессе

Кемерово
2018

Рецензенты:

И.В. Чичерин, к.т.н., доцент кафедры информационных и автоматизированных производственных систем

Семенова Ольга Сергеевна
Основы алгоритмизации и программирования: методические указания к самостоятельной работе [Электронный ресурс]: для студентов специальности СПО 09.02.07 «Информационные системы и программирование» / сост. О.С. Семенова, КузГТУ. – Электрон. издан. – Кемерово, 2018.

Приведенные методические указания к самостоятельной работе по курсу «Основы алгоритмизации и программирования» позволяют углубить знания, полученные в ходе аудиторных занятий; способствуют закреплению теоретических положений; развивают навыки по их практическому применению.

Часть №1. МЕТОДИЧЕСКИЕ РЕКОМЕНДАЦИИ ПО САМОСТОЯТЕЛЬНОЙ ПОДГОТОВКЕ К ИЗУЧЕНИЮ ТЕОРЕТИЧЕСКОГО МАТЕРИАЛА

Раздел 1. Основы алгоритмизации

Тема 1.1. Алгоритмизация вычислительного процесса

Содержание темы

1. Понятие алгоритма.
2. Способы описания алгоритмов.
3. Линейный разветвляющийся, циклический алгоритм.

Литература [1-3].

Методические рекомендации

В процессе изучения данного раздела студент должен усвоить, что ***программное обеспечение (ПО)*** – это совокупность программных средств для обеспечения нормальной работы вычислительной системы, подразделяющееся на общее и прикладное программное обеспечение. Все программное обеспечение можно разделить на три вида:

- системное ПО
- средства разработки
- прикладные программы

Системное программное обеспечение – это операционные системы, различные программы-утилиты для диагностики ресурсов компьютера (например, тестирования оперативной памяти), предоставления пользователю удобного способа работы взаимодействия с компьютером (например, командная строка), а также обслуживания ресурсов компьютера (например, разметка диска).

К ***средствам программирования*** относятся множество языков программирования, средства для автоматизации процесса создания программ, компиляторы и интерпретаторы. Языки и системы программирования являются по своему назначению инструментами для создания действительно полезного ПО. С их помощью создается как прикладное так и системное программное обеспечение, а также новые средства разработки.

Огромную долю в ПО занимают ***прикладные программы***, которые в свою очередь делят на ***универсальные*** и ***специализированные***.

Перед созданием любого программного обеспечения необходимо построить алгоритм его функционирования. ***Алгоритм*** – это понятное и точ-

ное предписание совершить определенную последовательность действий, направленную на достижение указанной цели или решение поставленной задачи. ***Алгоритм применительно к вычислительной машине*** – точное предписание, т. е. набор операций и правил их чередования, при помощи которого, начиная с некоторых исходных данных, можно решить любую задачу фиксированного типа.

Любой алгоритм характеризуется следующими свойствами:

· ***Дискретность*** - алгоритм должен быть разбит на шаги (отдельные законченные действия).

· ***Определенность*** - у исполнителя не должно возникать двусмысленностей в понимании шагов алгоритма (исполнитель не должен принимать самостоятельные решения).

· ***Результативность (конечность)*** - алгоритм должен приводить к конечному результату за конечное число шагов.

· ***Понятность*** - алгоритм должен быть понятен для исполнителя.

· ***Эффективность*** - из возможных алгоритмов выбирается тот алгоритм, который содержит меньше шагов или на его выполнение требуется меньше времени.

В процессе алгоритмизации исходный алгоритм разбивается на отдельные связанные части, называемые шагами, или ***частными алгоритмами***. Различают четыре основных типа частных алгоритмов:

· линейный алгоритм;

· алгоритм с ветвлением;

· циклический алгоритм;

· вспомогательный, или подчиненный, алгоритм.

Линейный алгоритм – набор инструкций, выполняемых последовательно во времени друг за другом.

Алгоритм с ветвлением – алгоритм, содержащий хотя бы одно условие, в результате проверки которого ЭВМ обеспечивает переход на один из двух возможных шагов.

Циклический алгоритм – алгоритм, предусматривающий повторения одного и того же действия над новыми исходными данными. Необходимо заметить, что циклический алгоритм легко реализуется посредством двух ранее рассмотренных типов алгоритмов.

Вспомогательный, или подчиненный, алгоритм – алгоритм, ранее разработанный и целиком используемый при алгоритмизации конкретной задачи.

В настоящее время различают несколько способов описания алгоритмов:

1. ***Словесный***, то есть записи на естественном языке, описание словами последовательности выполнения алгоритма.

Например: Записать алгоритм нахождения наибольшего общего делителя (НОД) двух натуральных чисел. Алгоритм может быть следующим: задать два числа; если числа равны, то взять любое из них в качестве ответа и остановиться, в противном случае продолжить выполнение алгоритма; определить большее из чисел; заменить большее из чисел разностью большего и меньшего из чисел; повторить алгоритм с шага

2. ***Формульно-словесный***, аналогично пункту 1, плюс параллельная демонстрация используемых формул.

В качестве примера можно привести ведение лекций преподавателем (словесный способ) с одновременной записью формул на доске (формульный).

3. ***Графический***, то есть с помощью блок-схем (рис. 1).

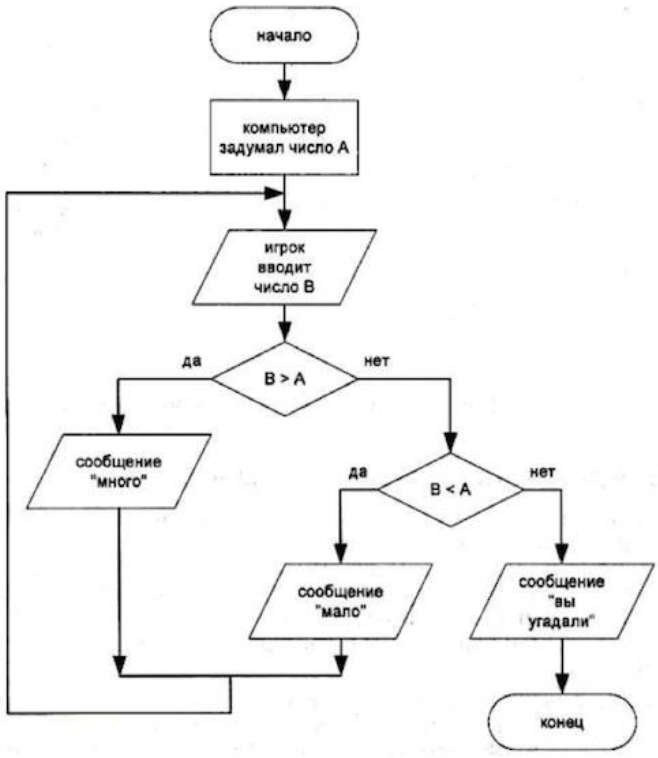

Рис. 1 – Описание алгоритма с помощью блок-схемы

Графический способ представления алгоритмов является более компактным и наглядным по сравнению со словесным. При графическом исполнении алгоритм изображается в виде последовательности связанных между

собой блочных символов, каждый из которых соответствует выполнению одного из действий. Такое графическое представление называется схемой алгоритма или блок-схемой. В блок-схеме каждому типу действий (вводу исходных данных, вычислению значений выражений, проверке условий, управлению повторением действий, окончанию обработки и т.п.) соответствует геометрическая фигура, представленная в виде блочного символа. Блочные символы соединяются линиями переходов, определяющими очередность выполнения действий.

4. ***Программный***, то есть тексты на языках программирования, предназначенные для исполнения на компьютере.

Контрольные вопросы:

1. Дайте определение алгоритму.
2. Свойства алгоритма.
3. Способы записи алгоритма.
4. Основные элементы блок-схемы.
5. Виды алгоритмов.
6. Отличительные особенности алгоритмов с предусловием и постусловием.
7. Какие цели преследуются при разработке прикладного программного обеспечения?
8. Зачем нужна алгоритмизация вычислительного процесса?
9. Какие способы описания алгоритмов вы знаете?
10. Какие блоки используются при графическом способе описания алгоритма?

Раздел 2. Введение в программирование

Тема 2.1. Языки программирования

Содержание темы

1. Развитие языков программирования.
2. Обзор языков программирования. Области применения языков программирования. Стандарты языков программирования. Среда проектирования. Компиляторы и интерпретаторы.
3. Жизненный цикл программы. Программа. Программный продукт и его характеристики.
4. Основные этапы решения задач на компьютере.

Тема 2.2. Типы данных

Содержание темы

1. Типы данных. Простые типы данных. Производные типы данных. Структурированные типы данных.

Литература [1-3].

Языки программирования

Существуют различные классификации языков программирования. По наиболее распространенной классификации все языки программирования, в соответствии с тем, в каких терминах необходимо описать задачу, делят на языки ***низкого*** и ***высокого уровня***.

Если язык близок к естественному языку программирования, то он называется языком высокого уровня, если ближе к машинным командам, – языком низкого уровня.

В группу языков низкого уровня входят машинные языки и языки символического кодирования: Автокод, Ассемблер. Операторы этого языка – это те же машинные команды, но записанные мнемоническими кодами, а в качестве операндов используются не конкретные адреса, а символические имена. Все языки низкого уровня ориентированы на определенный тип компьютера, то есть являются машинно-зависимыми.

К языкам программирования высокого уровня относят Фортран (переводчик формул – был разработан в середине 50-х годов программистами фирмы IBM и в основном используется для программ, выполняющих естественно-научные и математические расчеты), Алгол, Кобол (коммерческий язык – используется, в первую очередь, для программирования экономических задач), Паскаль, Бейсик (был разработан профессорами Дармутского колледжа Джоном Кемени и Томасом Курцом.), Си, Пролог (в основе языка лежит аппарат математической логики) и т.д.

Эти языки машинно-независимы, так как они ориентированы не на систему команд той или иной ЭВМ, а на систему операндов, характерных для записи определенного класса алгоритмов. Однако программы, написанные на языках высокого уровня, занимают больше памяти и медленнее выполняются, чем программы на машинных языках.

Программу, написанную на языке программирования высокого уровня, ЭВМ не понимает, поскольку ей доступен только машинный язык. Поэтому для перевода программы с языка программирования на язык машинных кодов используют специальные программы-трансляторы.

Существует три вида транслятора: ***интерпретаторы*** (это транслятор, который производит пооператорную обработку и выполнение исходного кода программы), ***компиляторы*** (преобразует всю программу в модуль на машинном языке, после чего программа записывается в память компьютера и лишь потом исполняется) и ***ассемблеры*** (переводят программу, записанную на языке ассемблера, в программу на машинном языке).

Языки программирования также можно разделять на поколения:

– *языки первого поколения*: машинно-ориентированные с ручным управлением памяти на компьютерах первого поколения.

– *языки второго поколения*: с мнемоническим представлением команд, так называемые автокоды.

– *языки третьего поколения*: общего назначения, используемые для создания прикладных программ любого типа. Например, Бейсик, Кобол, Си и Паскаль.

– *языки четвертого поколения*: усовершенствованные, разработанные для создания специальных прикладных программ, для управления базами данных.

– *языки программирования пятого поколения*: языки декларативные, объектно–ориентированные и визуальные. Например, Пролог, ЛИСП (используется для построения программ с использованием методов искусственного интеллекта), Си++, Visual Basic, Delphi.

Языки программирования также можно классифицировать на процедурные и непроцедурные.

В ***процедурных языках*** программа явно описывает действия, которые необходимо выполнить, а результат задается только способом получения его при помощи некоторой процедуры, которая представляет собой определенную последовательность действий. Среди процедурных языков выделяют в свою очередь *структурные* и *операционные языки*. В структурных языках одним оператором записываются целые алгоритмические структуры: ветвления, циклы и т.д. В операционных языках для этого используются несколько операций. Широко распространены следующие структурные языки: Паскаль, Си, Ада, ПЛ/1. Среди операционных известны Фортран, Бейсик, Фокал.

Непроцедурное (декларативное) программирование появилось в начале 70-х годов 20 века. К непроцедурному программированию относятся *функциональные* и *логические языки*.

В *функциональных* языках программа описывает вычисление некоторой функции. Обычно эта функция задается как композиция других, более простых, те в свою очередь делятся на еще более простые задачи и т.д. Один из основных элементов функциональных языков – рекурсия. Оператора присваивания и циклов в классических функциональных языках нет.

В *логических* языках программа вообще не описывает действий. Она задает данные и соотношения между ними. После этого системе можно задавать вопросы. Машина перебирает известные и заданные в программе данные и находит ответ на вопрос. Порядок перебора не описывается в программе, а неявно задается самим языком. Классическим языком логического программирования считается Пролог. Программа на Прологе содержит, набор предикатов–утверждений, которые образуют проблемно–ориентированную базу данных и правила, имеющие вид условий.

Можно выделить еще один класс языков программирования – объектно–ориентированные языки высокого уровня. На таких языках не описывают подробной последовательности действий для решения задачи, хотя они содержат элементы процедурного программирования. Объектно–ориентированные языки, благодаря богатому пользовательскому интерфейсу, предлагают человеку решить задачу в удобной для него форме. Первый объектно-ориентированный язык программирования Simula был создан в 1960-х годах Нигаардом и Далом.

Контрольные вопросы:

1. Опишите семантику языков программирования.
2. Чем отличаются компилируемые и интерпретируемые языки?
3. Первые языки программирования высокого уровня.
4. Алгоритмические языки программирования.
5. Какие методы используются при создании программного кода?
6. Перечислите основные виды ПО и его назначение.
7. Перечислите основные этапы разработки прикладного программного обеспечения.
8. В чем заключается постановка задачи оптимизации?
9. Какие языки программирования вы знаете?

Раздел 3. Объектно-ориентированное программирование

Тема 3.1. Основные принципы объектно-ориентированного программирования (ООП)

Содержание темы

1. История развития ООП. Базовые понятия ООП: объект, его свойства и методы, класс, интерфейс.
2. Основные принципы ООП: инкапсуляция, наследование, полиморфизм.
3. Классы объектов. Компоненты и их свойства.
4. Событийно-управляемая модель программирования. Компонентно-ориентированный подход.

Тема 3.2. Интегрированная среда разработчика.

Содержание темы

1. Требования к аппаратным и программным средствам интегрированной среды разработчика.
2. Интерфейс среды разработчика: характеристика, основные окна, инструменты, объекты. Форма и размещение на ней управляющих элементов.
3. Панель компонентов и их свойства. Окно кода проекта.
4. Состав и характеристика проекта. Выполнение проекта. Настройка среды и параметров проекта.
5. Панель компонентов и их свойства. Окно кода проекта. Состав и характеристика проекта. Выполнение проекта. Настройка среды и параметров проекта.
6. Настройка среды и параметров проекта.

Тема 3.3. Визуальное событийно-управляемое программирование

Содержание темы

1. Основные компоненты (элементы управления) интегрированной среды разработки, их состав и назначение.
2. Дополнительные элементы управления. Свойства компонентов. Виды свойств. Синтаксис определения свойств. Назначения свойств и их влияние на результат. Управление объектом через свойства.
3. События компонентов (элементов управления), их сущность и назначение. Создание процедур на основе событий.

Литература [1-3].

Методические рекомендации

В процессе изучения данного раздела студент должен усвоить основы работы в среде Visual Basic. Для этого студенту необходимо самостоя-

тельно изучить теоретический материал:

1. Интегрированная среда разработки приложений MS Visual Basic

- Изучение среды и основ работы в Visual Basic
- Запуск Visual Basic и создание нового проекта
- Основные компоненты интерфейса Visual Basic
- Текстовое меню и панель инструментов
- Окна: конструктора форм и панель элементов управления
- Создание формы приложения и сохранение проекта
- Запуск приложения и работа с окном редактора исходного кода
- Этапы создания простейшего приложения

2. Элементы управления (ЭУ)

- Виды элементов управления

В данном подразделе студенту необходимо обратить внимание на наиболее часто используемые ЭУ, приведённые в таблице 1.

Таблица 1 – Элементы управления

Название элемента управления	**Описание элемента управления**
Командная кнопка CommandButton	Используется для выполнения действий. К каждой кнопке присоединяется процедура, которая выполняется тогда, когда пользователь щелкает по кнопке
Ярлык (надпись) Label	Ярлык добавляет к форме любой текст
Текстовое окно TextBox	Служит для ввода пользователем любых данных
Кнопки–переключатели OptionButton	Обеспечивают пользователю возможность выбрать одну опцию из нескольких
Контрольные индикаторы CheckBox	Обеспечивают пользователю возможность выбрать любое количество опций
Рамка Frame	Позволяет объединять элементы управления в группы, которые будут работать независимо друг от друга
Линейки прокрутки	Используются для вертикальной или горизонтальной прокрутки данных в текстовых окнах
Окна списков ListBox	Позволяет пользователю выбирать из списка возможностей
Комбинированные списки ComboBox	Комбинация текстового окна и окна списка. Элемент может быть выбран из списка либо

	щелчком по нему, либо впечатывание имени в текстовое поле
Изображение Image	Выводит на экран содержимое графического файла
Данные Data	Осуществляет доступ к данным
Контур Shape	Создает на форме прямоугольник, квадрат, овал, окружность, прямоугольник и квадрат со скругленными углами
Рисунок Picture	Предназначен для отображения графических изображений; в качестве контейнера для других элементов управления; в виде графического окна для вывода текста, графических элементов
Линия Line	Добавляет на форму линию
ТаймерTimer	Обрабатывает данные системных часов

- Свойства элементов управления.

В данном подразделе студенту необходимо обратить внимание на то, что свойства любого объекта можно установить во время проектирования в окне Properties, или во время выполнения, написав соответствующий программный код: Объект.Свойство=значение.

Каждый из размещаемых в форме элементов управления определяется собственным набором свойств. Но есть свойства, присущие большинству объектов. К ним относятся такие свойства, как Name, Height, Width, Left, Top, Visible, Caption и т.д.

- Методы

В данном подразделе студент должен усвоить, что помимо свойств, объект имеет методы, определяющие выполняемые им действия. Вызов метода осуществляется по правилам:

- не требующего ввода аргумента: объект.метод
- требующего ввода аргумента: объект.метод аргумент1, аргумент2,…, аргумент n
- возвращающего значение:

переменная=объект.метод (аргумент1, аргумент2,…,аргумент n).

Каждый объект имеет некоторое количество доступных ему методов.

- События

В данном подразделе студент должен уяснить, что помимо свойств и методов, для объектов можно задать программные коды, написанные на

языке Visual Basic и выполняемые при наступлении связанных с ними событий. Наиболее часто используемые события приведены в таблице 2.

Таблица 2 – События

Событие	Описание
Click (щелчок)	Происходит при щелчке пользователя кнопкой мыши по объекту
Double Click (двойной щелчок)	Происходит при двойном щелчке пользователя кнопкой мыши по объекту
Change	Происходит при изменении объекта. Например, при вводе текста.
Activate	Происходит при загрузке формы

Контрольные вопросы:

1. Какие основные объекты интегрированной среды программирования Visual Basic вы знаете?
2. Какие стандартные объекты используются при создании прикладной программы?
3. Перечислите наиболее часто используемые свойства объектов.
4. Что такое “методы”?
5. Перечислите наиболее часто используемые события.
6. Какие типы данных существуют в Visual Basic?

Раздел 4. Операторы языка программирования

Тема 4.1. Операторы языка программирования

Содержание темы

1. Операции и выражения. Правила формирования и вычисления выражений. Ввод и вывод данных. Оператор присваивания. Составной оператор.
2. Условный оператор. Оператор выбора.
3. Цикл с постусловием. Цикл с предусловием. Цикл с параметром. Вложенные циклы.
4. Строки. Стандартные процедуры и функции для работы со строками.
5. Массивы. Двумерные массивы.

Литература [1-3].

Методические рекомендации

В процессе изучения данного раздела студент должен усвоить ос-

новные операторы языка программирования. Для этого студенту необходимо самостоятельно изучить теоретический материал

1. Работа с переменными

- Объявление переменных
- Присвоение значения переменным

В данном подразделе студент должен уяснить, что переменная – эта поименованная ячейка памяти, хранящая какое-либо одно значение (одно число, один фрагмент текста). Имя переменной – это строка символов, которая отличает ее от других переменных и объектов программы (элементов управления). Значение переменной – это данные, которые в ней хранятся. Тип данных (тип переменной) обуславливает то, как хранятся и обрабатываются данные.

Кроме того, студент должен разобраться с синтаксисом операторов, используемых для объявления переменных: *DIM, STATIC, PUBLIC, PRIVATE*; уяснить необходимость использования того или иного типа данных (INTEGER, LONG, SINGLE, DOUBLE, BOOLEN, BYTE, STRING, VARIANT).

2. Разработка приложений с разветвляющимися структурами алгоритмов

- Разветвляющиеся алгоритмы
- Программирование условий в Visual Basic
- Логические выражения

В данном подразделе студент должен научиться реализовывать разветвляющийся алгоритм для выполнения различных фрагментов программы в зависимости от выполнения некоторого условия. Разветвляющийся алгоритм реализуется с помощью условного оператора If . . . Then, который может иметь простую однострочную или блочную структуру.

3. Разработка приложений с циклическими структурами алгоритмов

- Циклические алгоритмы
- Виды циклов
- Программирование циклов в Visual Basic

В данном разделе студент должен научиться работать с циклами FOR…NEXT (циклы с заданным количеством повторений, которые позволяют выполнять группу операторов заданное число раз) и с циклами Do While...Loop, которые предназначены для ситуаций, когда количество проходов цикла заранее не известно, но зато известно условие выхода из цикла.

4. Массивы

- Основные сведения о массивах
- Одномерные массивы
- Двумерные массивы
- Динамические массивы

В данном разделе студент должен научиться объявлять одномерные и многомерные массивы с помощью операторов *DIM, STATIC, PUBLIC, PRIVATE*; заполнять массивы данными; выводить массивы на экран; выполнять с массивами различные операции (нахождение максимального элемента, суммы всех элементов, количества элементов и т.д.).

5. Строковые данные

- Функции обработки строк
- Работа со строковым типом данных.

В данном разделе студент должен научиться работать с функциями обработки строк: LEN, MID, RIGHT и т.д.

Контрольные вопросы:

1. Какие функции работы со строками вы знаете?
2. Как осуществляется сложение строк?
3. Что такое массив?
4. Какие массивы вы знаете?
5. Какие способы формирования (заполнения) массивов вы знаете?
6. Как обратиться к элементу массива?
7. Какие основные математические операторы Visual Basic вы знаете?
8. Какие основные функции, существующие в Visual Basic, вы знаете?
9. Чем отличаются циклы с заданным повторением от циклов по условию?
10. Какие виды условий существуют?
11. Какие операторы используются при написании условных выражений?
12. Какие операции можно совершать с условными выражениями?
13. Зачем используются массивы данных в организации дорожного движения?

Раздел 5. Структурное и модульное программирование

Тема 5.1. Процедуры и функции

Содержание темы

1. Общие сведения о подпрограммах. Определение и вызов подпрограмм. Область видимости и время жизни переменной. Механизм передачи параметров. Организация функций.

2. Рекурсия. Программирование рекурсивных алгоритмов.

Тема 5.2. Структурное и модульное программирование

1. Основы структурного программирования. Методы структурного программирования.
2. Модульное программирование. Понятие модуля. Структура модуля. Компиляция и компоновка программы.
3. Стандартные модули.

Литература [1-3].

Методические рекомендации

При изучении данного раздела студент должен уяснить, что ***модульное программирование*** – это независимое программирование каждого модуля. Программирование обычно ведут начиная с верхнего уровня. Вначале составляется программа для ограниченного числа модулей верхнего уровня и производится тестирование всей задачи, при этом остальные модули не программируются, а для них делают так называемые заглушки.

Каждый новый модуль включается в основную программу только после того, как проведено его полное тестирование.

Структурное программирование (кодирование) – процесс программирования на алгоритмическом языке с использованием определенных конструкций. Основные положения структурного программирования:

1. любая программа составляется на базе основных алгоритмических структур трех видов: линейного, разветвляющего и циклического.
2. между этими структурами производится передача управления только вперед, что соответствует в блок-схеме направлению линий сверху вниз.
3. недопустимо пользоваться в программе специальной командой безусловной передачи управления из одного места программы в другое (например, GOTO).

Структурное кодирование применяется для программирования отдельных модулей. Вначале на основе положений 1, 2, 3 разрабатывается алгоритм, а затем базовые алгоритмические конструкции заменяются соответствующими конструкциями конкретного языка. Структурный контроль необходим при разработке сложных задач, включающих десятки и сотни модулей. Это новая форма контроля процесса разработки программ, в которой участвует несколько человек. По мере продвижения в разработке задачи периодически определенной группе сотрудников раздаются рабочие материалы по данной задаче. Они знакомятся с ними, а затем на совмест-

ном обсуждении выявляются недостатки данного этапа, которые разработчик программы должен в ближайшее время устранить.

Процедуры – это самые важные функциональные блоки языка VBA. В VBA вы можете выполнить только программный код, который содержится в какой-либо процедуре (обычной в стандартном модуле, событийной для элемента управления на форме и т.п.). Иногда начинающие пользователи пытаются записать команды прямо в область объявлений стандартного модуля и не могут понять, почему они не выполняются (сообщений о ошибке при этом не выдается — просто этот код становится "невидим" для компилятора). Причина проста — в разделе объявлений модуля (когда в верхних списках показываются значения (General) и (Declarations) могут быть только объявления переменных уровня модуля и некоторые специальные инструкции для компилятора. Весь остальной программный код должен находится внутри процедур.

В VBA предусмотрены следующие типы процедур:

Процедура типа Sub (подпрограмма) — универсальная процедура для выполнения каких-либо действий:

```
Sub Farewell()
MsgBox “Goodbye”
End Sub
```

Процедура типа Function (функция) — тоже набор команд, которые должны быть выполнены. Принципиальное отличие только одно: функция возвращает вызвавшей ее программе какое-то значение, которое там будет использовано. Пример процедуры:

```
Function Tomorrow()
Tomorrow = DateAdd("d", 1, Date())
End Function
```

и пример ее вызова:

```
Private Sub Test1()
Dim dDate
dDate = Tomorrow
MsgBox dDate
End Sub
```

В тексте функции необходимо предусмотреть оператор, который присваивает ей какое-либо значение. В нашем случае это строка Tomorrow = DateAdd(" d", 1, Date()).

В VBA предусмотрены сотни встроенных функций (и гораздо большее число предусмотрено в объектных моделях приложений Office). Даже в нашем примере используются две встроенные функции: Date(), которая возвращает текущую дату по часам компьютера и DateAdd(), которая умеет прибавлять к текущей дате определенное количество дней, недель, месяцев, лет и т.п. Про встроенные функции будет рассказано ниже.

В VBA имеются также процедуры обработки событий (event procedure) – процедуры типа Sub специального назначения, которые выполняется в случае возникновении определенного события. Например, *Private Sub UserForm_Click()*

…..

End Sub

Контрольные вопросы:

1. Что такое подпрограмма?
2. Какие параметры называются фактическими?
3. Какие параметры называются формальными?
4. Как связаны между собой формальные и фактические параметры?
5. Какие переменные называются глобальными?
6. Какие переменные называются локальными?

Раздел 6. *Решение прикладных задач*

Тема 6.1. *Постановка задачи*

1. Анализ, формальная постановка и выбор метода решения задачи.
2. Разработка алгоритма решения задачи.

Тема 6.2. *Проектирование объектно-ориентированного приложения*

1. Создание интерфейса приложения.
2. Разработка программных модулей.
3. Тестирование, отладка приложения.

Литература [4, 5].

Методические рекомендации

В процессе изучения данного раздела студент должен усвоить, что при создании программного обеспечения необходимо пройти несколько этапов:

- *Постановка задачи*. На этом этапе осуществляется формулирование цели решения задачи, описывается входная и выходная информация. Вход-

ная информация по задаче – данные, поступающие на вход задачи и используемые для её решения. Выходная информация может быть представлена в виде документов, кадров на экране монитора, информации в базе данных, выходного сигнала устройству управления. Постановка задачи разрабатывается организацией, разработчиком программной продукции, на основании технического задания совместно с заказчиком. Главный исполнитель – это разработчик.

- *Математическое описание задачи*. На этом этапе выражаются существующие соотношения между исследуемыми величинами посредством математических формул, указываются допущения и ограничения.

Математическая постановка включает в себя:

а) выбор, обоснование и описание модели исследуемого объекта;

б) определение цели исследования;

в) выбор и описание методов решения поставленных задач (достижения цели исследования).

Для построения математической модели на основании словесной постановки задачи выбираются переменные, подлежащие определению, записываются ограничения и выявляются связи между переменными.

В общем случае, математические модели исследуемых объектов могут быть числовыми (с конкретными числовыми значениями характеристик), логическими (в основе логические выражения) и графическими (графики, диаграммы, рисунки, которые легко преобразуются в математические формулы). Разрабатываемая математическая модель может отражать внутреннюю структуру объекта (отношения между составляющими этот объект элементами) или функционирование объекта (зависимости между воздействиями на объект и его состояниями.

При построении математической модели может преследоваться цель определения такого состояния объекта, которое является наилучшим или допустимым в каком-либо смысле. Модель может быть предназначена для объяснения наблюдаемых фактов или прогноза поведения объекта.

Примерная структура математической постановки задачи выглядит стандартным образом: дано; требуется получить; решение; выводы.

- *Выбор и обоснование метода решения задачи*. Выбор методов исследования осуществляется исходя из построенной модели и сформулированной цели. При этом учитывается точность вычислений, время решения задачи на ЭВМ, требуемый объем памяти и т.д.

- *Алгоритмизация вычислительного процесса*. На этом этапе строится детальный алгоритм программы. При этом процесс обработки данных разбивается на отдельные блоки, устанавливается последовательность выполнения блоков, разрабатывается блок-схема алгоритма программы.
- *Проектирование базы данных*.
- *Написание программного кода*. На этом этапе создаётся программный код на заданном языке программирования в соответствии с составленным алгоритмом.
- *Отладка программы, поиск и устранение синтаксических и логических ошибок*. Отладка – этап разработки компьютерной программы, на котором обнаруживают, локализуют и устраняют ошибки. Чтобы понять, где возникла ошибка, приходится:

а) узнавать текущие значения переменных;
б) выяснять, по какому пути выполнялась программа.
Существуют две взаимодополняющие технологии отладки:
а) Использование отладчиков — программ, которые включают в себя пользовательский интерфейс для пошагового выполнения программы: оператор за оператором, функция за функцией, с остановками на некоторых строках исходного кода или при достижении определённого условия;
б) Вывод текущего состояния программы с помощью расположенных в критических точках программы операторов вывода — на экран, принтер, громкоговоритель или в файл. Вывод отладочных сведений в файл называется журналированием.

«Отладка сложна и может занимать непредсказуемо долгое время, поэтому цель в том, чтобы миновать большую её часть. Технические приёмы, которые помогут уменьшить время отладки, включают хороший дизайн, хороший стиль, проверку граничных условий, проверку правильности исходных утверждений и разумности кода, защитное программирование, хорошо разработанные интерфейсы, ограниченное использование глобальных переменных, автоматические средства контроля и проверки. Грамм профилактики стоит тонны лечения» (Брайан Керниган и Роб Пайк).

То есть, при написании программного кода проще сделать так, чтобы отладка нужна была как можно реже. Для этого применяются следующие методы:

- статический анализ кода. На этой фазе программа сканер ищет последовательности в исходном тексте, соответствующие небезопасным вызовам

функций и т. Д. Фактически идет сканирование исходного текста программы на основе специальной базы правил, которая содержит описание небезопасных образцов кода;

- фаззинг. Это процесс подачи на вход программы случайных или некорректных данных и анализ реакции программы;
- Reverse engineering (Обратная инженерия). Этот случай возникает, когда независимые исследователи ищут уязвимости и недокументированные возможности программы.

Контрольные вопросы:

1. Опишите этапы постановки задачи.
2. В чем заключается анализ, формальная постановка и выбор метода решения задачи?
3. Опишите принципы разработки алгоритмов решения задачи.
4. Опишите процесс программирования модулей, тестирования и отладки программы
5. Опишите цели создания прикладного программного обеспечения.
6. Опишите любую прикладную задачу в области организации перевозок с помощью математических формул.
7. Опишите процесс создания базы данных в Excel, Access.
8. Каким способом лучше описать алгоритм работы подвижного состава на линии?
9. Опишите назначение блоков, входящих в блок-схему алгоритма прикладной программы.
10. Проанализируйте результаты, полученные в результате решения задачи на ЭВМ.

Часть №2 МЕТОДИЧЕСКИЕ РЕКОМЕНДАЦИИ ПО ВЫПОЛНЕНИЮ ДОМАШНИХ ЗАДАНИЙ

Домашние задания по курсу «Основы алгоритмизации и программирования» включают в себя подготовку к практическим работам. Проработка учебников, изучение конспекта лекций, ознакомление с методическими указаниями по выполнению практических работ позволит студенту всесторонне изучить разбираемую на занятиях тему. Это позволит студенту не только качественно выполнить практические работы, но и успешно пройти текущие и промежуточную аттестации.

Список литературы

1. Семакин, И. Г. Основы алгоритмизации и программирования [Текст] : учебник для образовательных организаций, реализующих программы среднего профессионального образования по специальностям "Информационные системы и программирование", "Сетевое и системное администрирование", "Обеспечение информационной безопасности автоматизированных систем", "Обеспечение информационной : [для студентов СПО] / И. Г. Семакин, А. П. Шестаков. – Москва : Академия, 2017. – 304 с. – Доступна электронная версия: http://www.academia-moscow.ru/catalogue/4831/345450/

2. Колдаев, В. Д. Основы алгоритмизации и программирования. – Москва : НИЦ ИНФРА-М, 2015. – 416 с. – Режим доступа: http://znanium.com/go.php?id=484837. – Загл. с экрана. (14.12.2018)

3. Колдаев, В. Д. Основы алгоритмизации и программирования. – Москва : НИЦ ИНФРА-М, 2019. – 414 с. – Режим доступа: http://znanium.com/go.php?id=980416. – Загл. с экрана. (14.12.2018)

4. Цветкова, М. С. Информатика [Электронный ресурс] : учебник для использования в учебном процессе образовательных учреждений СПО на базе основного общего образования с получением среднего общего образования / М. С. Цветкова, И. Ю. Хлобыстова. – Москва : Академия, 2017. – 352 с. – Режим доступа: http://academia-moscow.ru/reader/?id=227485#copy. – Загл. с экрана. (14.12.2018)

5. Гаврилов, М. , В. Информатика и информационные технологии 4-е изд., пер. и доп.[электронный ресурс]. – Москва : Юрайт, 2018. – 383 с. – Режим доступа: https://biblio-online.ru/book/informatika-i-informacionnye-tehnologii-413451. – Загл. с экрана. (14.12.2018)

Содержание