МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ
БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ
Кафедра экономической информатики

А.В.Бахирев, Е.Н. Живицкая,
В.Н. Комличенко, С.А. Соколов

МЕТОДИЧЕСКОЕ ПОСОБИЕ
для выполнения курсового проектирования
по дисциплине
«Основы информатики и вычислительной техники»
для студентов специальности Э.01.03.00

МИНСК 2000

УДК (075.8)
ББК 22.1 Я 73
М

Методическое пособие для выполнения курсового проектирования по дисциплине «Основы информатики и вычислительной техники» для студентов специальности Э.01.03.00. А.В.Бахирев, Е.Н. Живицкая, В.Н. Комличенко, С.А. Соколов. -Мн.: БГУИР, 2000.- с.47.ил.

1. Цель и задачи курсового проектирования

Целью курсового проектирования является освоение методов проектирования и программирования и получение навыков самостоятельного решения простейших экономических и управленческих задач с использованием ПЭВМ.

Задачи курсового проектирования:

1. Углубить и закрепить теоретические знания по дисциплине «Основы информатики и вычислительной техники».
2. Научиться искать, анализировать и использовать документальные источники научной информации.
3. Научиться пользоваться исходными данными и нормативно-справочными материалами.
4. Приобрести навыки самостоятельной познавательной деятельности, умение формулировать суждения и выводы, логически последовательно и доказательно их излагать.
5. Получить навыки самостоятельной разработки программного обеспечения.
6. Приобрести навыки оформления отчетной документации.
7. Выработать умение публичной защиты подготовленного материала (делать доклад, отвечать на вопросы, формулировать свое мнение и др.)

Ведущим требованием к курсовому проекту является наличие достаточно высокого уровня теоретического обоснования и использование новых подходов к проектированию.

2.Общие положения

Курсовой проект выполняется студентом самостоятельно под руководством преподавателя профилирующей кафедры в соответствии с индивидуальным заданием. Задание на курсовой проект должно быть выдано не позже третьей рабочей недели соответствующего семестра для студентов дневного обучения и в период проведения установочных занятий по предмету для студентов заочного отделения. При этом разъясняется принцип выбора варианта задания; уточняется тема и формулируется цель проектирования, исходные данные по проектированию, акцентируются моменты, на которые необходимо обратить особое внимание; обсуждаются возникающие вопросы, определяется время сдачи проекта и график консультаций. Задание оформляется и детализируется в соответствии с разделами специального бланка, образец которого и возможный вариант его заполнения приведен в прил. 4.

В ходе выполнения курсового проекта руководитель консультирует студента, контролирует своевременность и правильность выполнения курсового проекта, а также оформления его пояснительной записки. График консультаций согласуется между преподавателем и руководителем в индивидуальном порядке.

Студенты- заочники могут консультироваться в «дни заочника», проводимые кафедрой обычно в 1-ю и 3-ью субботу каждого месяца, в назначенное время, определяемое на текущий семестр. График консультаций вывешивается на доске объявлений кафедры и доске объявлений деканата заочного факультета.

По результатам курсового проектирования составляется отчет, который содержит описание всех этапов проектирования и программной реализации, а также приложения, включающие распечатку его компьютерной реализации, диаграммы функциональной и структурной схем, возможно, иерархическую схему классов, схему объектов, таблицы, иллюстрации и другие вспомогательные или дополнительные материалы, которые загромождают текст основной части.

Отчет должен содержать правильно оформленный титульный лист, который включает название кафедры, фамилию и инициалы студента, номер группы, название предмета, тему и др.. Образец заполнения титульного листа приведен в прил.3.

Студенты- заочники привозят оформленный отчет в деканат заочного факультета или отправляют по почте в установленные сроки. Созданную программу размещают на отдельной дискете и предоставляют ее при защите дипломного проекта.

К защите допускаются студенты, вовремя представившие пояснительную записку для проверки. Проверка курсового проекта осуществляется в течение не более одной недели с момента ее получения. Проверка заключается в прочтении

пояснительной записки, контроле наличия в отчете всех пунктов задания курсового проекта, а также полноты и качества содержания пунктов.

Защита курсового проекта. Завершенная работа представляется руководителю для окончательной проверки. Решение о возможности допуска ее к защите определяется кафедрой на основании представления руководителя и изучения самого курсового проекта (КП) и доводится до сведения обучаемых не менее чем за неделю до защиты.

Защита КП проводится до начала зачетной сессии.

За 2-3 дня до защиты работы могут быть розданы для возможного исправления недостатков и учета замечаний.

Порядок защиты. Защита КП проводится на заседаниях кафедры или комиссии, состоящей из 2-3-х преподавателей с демонстрацией разработанной программы на компьютере.

Первое слово предоставляется студенту, который в своем кратком выступлении (до 5 мин.) должен осветить следующие моменты:

- каковы были цель и задачи курсового проектирования;
- какая литература рассматривалась и показалась наиболее содержательной и интересной, какие выводы были сделаны в результате анализа литературных источников;
- какие методы использовались для решения поставленных задач;
- какие трудности возникли в процессе проектирования, какие решения были приняты и краткое их обоснование;
- анализ недостатков и преимуществ представленного программного продукта и предложения по его совершенствованию.
- выводы и результаты исследования и проектирования.

При докладе следует свободно излагать материал исследования без заглядывания в отчет по КП, который в это время может находиться у членов комиссии.

Студенту могут быть заданы вопросы с целью проверки знания предмета, разработанной им программы и самостоятельности выполнения курсового проекта. Затем излагается краткая характеристика работы, оценка ее качества и самостоятельности выполнения.

Оценка работы обсуждается членами кафедры и выставляется после завершения защиты всех работ, допущенных в этот день.

3. Содержание и объем пояснительной записки

Рекомендуемое содержание пояснительной записки КП и содержание каждого пункта:

1. Введение (1...2 листа) – обычно описываются преимущества использования объектно-ориентированного программирования, цель, назначение и задачи, решаемые при помощи созданной программы.
2. Анализ литературных источников (не менее 5 источников с указанием ссылок) по теме проектирования (1…2 листа).
3. Описание автоматизируемой предметной области, обоснование выбора языка программирования, используемой операционной и инструментальной сред, предлагаемой структуры классов .
4. Описание структуры наследования классов (2...4 листа) – определяется структура будущих классов.
5. Описание назначения, свойств и методов каждого класса (1...2 листа) – описывается текущая реализация классов.
6. Описание программы (2...3 листа) – описываются логически законченные части программы и принцип функционирования программы (т.е. какие действия должен осуществлять пользователь для управления программой).
7. Выводы (1...2 листа) – описываются функциональные возможности программы: возможность ввода \ вывода, характерные особенности программы, введенные новшества, требования к программному обеспечению компьютера, на котором будет осуществляться демонстрация.
8. Литература (не менее 5 источников).
9. Приложение - содержит распечатку программы или программу, переписанную с экрана компьютера от руки.

Требования к образуемым классам:

1. Каждый класс должен иметь собственный конструктор.
2. Элементы данных класса должны быть описаны с ограничением на доступ private, или protected.
3. Назначение каждого свойства и метода класса должны быть снабжены комментариями.

Для повышения оценки необходимо:

1. Применить цвет при выводе на экран текста.
2. Применить символы псевдографики для построения таблиц или реализовать программу в графическом режиме.
3. Использовать подсветку и другие возможности для идентификации выбранного пункта при построении меню.

4.Требования к оформлению пояснительной записки

4.1.Общие положения

4.1.1. Пояснительная записка оформляется в соответствии с требованиями, принятыми в БГУИР, и единой системой программной документацией.
4.1.2. Все листы пояснительной записки, кроме титульного, должны иметь сквозную нумерацию. Номер листа пишется на верхнем поле в центре листа.
4.1.3. Листы пояснительной записки должны быть сброшюрованы в скоросшивателе или в обложке из плотной чертежной бумаги (лист формата А3 согнут пополам). Одна половина обложки используется в качестве титульного листа.
4.1.4. Пояснительная записка должна содержать:

- титульный лист;
- содержание;
- задание по курсовому проектированию;
- введение;
- основные главы пояснительной записки;
- выводы;
- список источников, использованных при разработке;
- приложение.

4.1.5. Титульный лист пояснительной записки оформляется в соответствии с требованиями стандарта БГУИР. Пример оформления титульного листа пояснительной записки приведен в прил. 3.
4.1.6. Содержание пояснительной записки размещают на отдельной (пронумерованной) странице (страницах), снабжают заголовком "СОДЕРЖАНИЕ", не нумеруют как раздел и включают в общее количество страниц пояснительной записки.

В содержание пояснительной записки включают номера разделов, подразделов, пунктов и подпунктов, имеющих заголовок, их наименования и номера страниц; номера и наименования (при наличии) приложений пояснительной записки и номера страниц.

Наименования, включенные в содержание, записывают строчными буквами. Прописными буквами должны записываться аббревиатуры.
Пример оформления содержания пояснительной записки приведен в приложении.
4.1.7. Задание по курсовому проектированию записывается на специальном бланке и утверждается заведующим кафедрой или его заместителем. Задание подписывается руководителем курсовой работы и студентом, принявшим задание к исполнению.

Задание по курсовому проектированию является отчетным документом, без которого пояснительная записка на проверку не принимается, а работа к защите не допускается.

Пример оформления задания по курсовому проектированию приведен в приложении.

4.1.8. Текст пояснительной записки при необходимости разбивается на пункты, а пункты - на подпункты, независимо от того, разделен документ на разделы и подразделы или нет.

4.1.9. Структурными элементами текста пояснительной записки являются разделы, подразделы, пункты, подпункты и перечисления.

Раздел - первая ступень деления, обозначенная номером и снабженная заголовком.

Подраздел - часть раздела, обозначенная номером и имеющая заголовок.

Пункт - часть раздела или подраздела, обозначенная номером. Может иметь заголовок.

Подпункт - часть пункта, обозначенная номером, может иметь заголовок.

Абзац - логически выделенная часть текста, не имеющая номера.

При отсутствии разделов в тексте документа его первым структурным элементом является пункт.

Допускается помещать текст между заголовками раздела и подраздела, между заголовками подраздела и пункта.

Внутри подразделов, пунктов и подпунктов могут быть даны перечисления, которые рекомендуется обозначать арабскими цифрами со скобкой: I), 2) и т.д. Допускается выделять перечисления простановкой дефиса перед текстом.

Не рекомендуется делать ссылки на элементы перечисления.

Каждый структурный элемент начинается с нового абзаца или нового листа.

4.1.10. Заголовки разделов пишут прописными буквами и размещают симметрично относительно правой и левой границ текста.

Заголовки подразделов записывают с абзаца строчными буквами (кроме первой прописной).

Переносы слов в заголовках не допускаются. Точку в конце заголовка не ставят.

Если заголовок состоит из двух предложений, их разделяют точкой.

Каждый раздел рекомендуется начинать с нового листа.

4.1.11. Расстояние между заголовком и последующим текстом, а также между заголовками раздела и подраздела должно быть равно:

- при выполнении документа рукописным способом - 10 мм;
- при выполнении машинописным способом - двум интервалам.

Для разделов и подразделов, текст которых записывают на одной странице с текстом предыдущего раздела, расстояние между последней строкой текста и последующим заголовком должно быть равно:

- при выполнении документа рукописным способом - не менее 15 мм;
- при выполнении машинописным способом - трем интервалам.

Расстояние между основаниями строк заголовка принимают таким, как в тексте.

4.1.12. Разделы, подразделы, пункты и подпункты следует нумеровать арабскими цифрами с точкой.

Разделы должны иметь порядковый номер (1,2 и т.д.).

В пределах раздела должна быть сквозная нумерация по всем подразделам, пунктам и подпунктам, входящим в данных раздел.

Нумерация подразделов включает номер раздела и порядковый номер подраздела, входящего в данный раздел, разделенные точкой (2.1, 3.1 и т.д.).

При наличии разделов и подразделов к номеру подраздела после точки добавляют порядковый номер пункта и подпункта (3.1., 3.1.1, и т.д.).

4.2. Текст пояснительной записки

4.2.1. Текст пояснительной записки должен быть кратким, четким, исключающим возможность неоднозначного толкования.

Термины и определения должны быть едиными и соответствовать установленным стандартам, а при их отсутствии - общепринятым в научно-технической литературе.

4.2.2. Для выделения отдельных понятий допускается изменять интервалы между словами, а также писать отдельные слова или части текста шрифтом, отличным от основного текста.

4.2.3. Необходимые пояснения к тексту пояснительной записки могут оформляться сносками. Сноска обозначается цифрой со скобкой, вынесенными над уровнем основного шрифта. Например: "печатающее устройство $^{2)}$…" или "бумага$^{5)}$…".

Если сноска относится к отдельному слову, знак сноски помещается непосредственно у этого слова, если же к предложению в целом, то в конце предложения. Текст сноски располагают в конце страницы и отделяют от основного текста линией длиной 3 см, проведенной в левой части страницы.

4.3. Иллюстрации

4.3.1. Иллюстрации могут быть расположены в тексте пояснительной записки и (или) в приложениях. Иллюстрации, если их в пояснительной записке более одной, нумеруют арабскими цифрами в пределах всего документа. В приложениях

иллюстрации нумеруются в пределах каждого приложения в порядке, установленном для основного текста документа. Ссылки на иллюстрации дают по типу " рис. 12" или "(рис.12)". Ссылки на ранее упомянутые иллюстрации дают с сокращенным словом "смотри", например, "см. рис. 12".
Номер иллюстрации и ее название помещают последовательно под иллюстрацией.

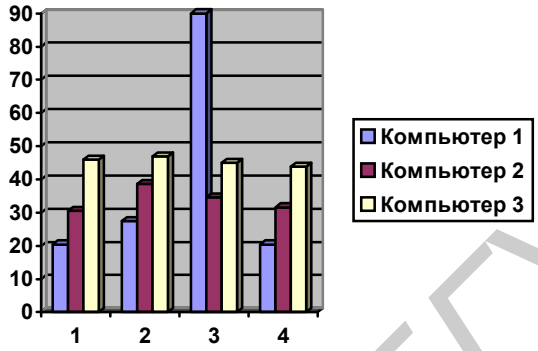


Рис.4.3.1. Пример оформления изображения

Иллюстрации должны быть расположены так, чтобы их было удобно рассматривать без поворота пояснительной записки или с поворотом по часовой стрелке.

4.4. Формулы

4.4.1. Формулы в пояснительной записке, если их более одной, нумеруются арабскими цифрами, номер ставят с правой стороны страницы в скобках, на уровне формулы. В пределах всего документа формулы имеют сквозную нумерацию. Ссылки в тексте на порядковый номер формулы дают в скобках, например: "в формуле (3)".
4.4.2. Значения символов и числовых коэффициентов, входящих в формулу, должны быть приведены непосредственно под формулой. Значения каждого символа пишут с новой строки в той последовательности, в какой они приведены в формуле. Первая строка расшифровки должна начинаться со слова "где", без двоеточия после него.
4.4.3. Размерность одного и того же параметра в пределах документа должна быть постоянной.

$$A = \sum_{i=0}^{N} B_i$$

Рис. 4.4.1. Пример оформления формул

4.4.4.Уравнения и формулы следует выделять из текста свободными строками. Выше и ниже каждой формулы должно быть оставлено не менее одной свободной

строки. Если уравнение не умещается в одну строку, оно должно быть перенесено после знака равенства (=) или после знаков плюс (+), минус (-), умножения (*) и деления (:).

4.5. Ссылки

4.5.1. В пояснительной записке допускаются ссылки на стандарты, технические условия и другие документы. При ссылках на стандарты и технические условия указывают их обозначения. Ссылаться следует на документ в целом или его разделы и приложения (с указанием обозначения и наименования документа, номера и наименования раздела или приложения). При повторных ссылках на раздел или приложение указывают только номер.

При ссылках на документ допускается проставлять в квадратных скобках его порядковый номер в соответствии с перечнем ссылочных документов. Допускается указывать только обозначение документа и (или) разделов без указания их наименований. Ссылки на отдельные подразделы, пункты и иллюстрации другого документа не допускаются. Допускаются ссылки внутри пояснительной записки на пункты, иллюстрации и отдельные подразделы. (например, если в строке пояснительной записки курсового проекта содержится указание [3], то это значит, что более подробная информация содержится в книге или статье, указанной под номером 3 в списке литературы пояснительной записки).

4.6. Таблицы

4.6.1. Цифровой материал для достижения лучшей наглядности и сравнимости показателей, как правило, следует оформлять в виде таблицы.
4.6.2. Сноски к таблицам располагают непосредственно под таблицей.
Примеры оформления таблиц - см. табл.4.6.1. и табл.4.6.2.

Таблица 4.6.1

Пример таблицы

Ф.И.О.	Возраст	Пол
Иванов И.И.	20	м
Петрова П.П.	40	ж

Таблица 4.6.2

Пример разрывающейся таблицы

Ф.И.О.	Возраст	Пол
Иванов И.И.	20	м
Петрова П.П.	40	ж

Продолжение таблицы 4.6.2

Ф.И.О.	Возраст	Пол
Сидоров С.С.	50	м
Сидорова Р.П.	45	ж

4.6.3.Каждая таблица должна иметь заголовок, который располагают над таблицей и печатают в центре строки. Заголовок и слово «Таблица» начинают с прописной буквы. Заголовок не подчеркивают.

4.6.4.Заголовки граф должны начинаться с прописных букв, подзаголовки – со строчных, если они составляют одно предложение с заголовком, и с прописных, если они самостоятельные. Делить головки таблицы по диагонали не допускается. Высота строк должна быть не менее 8 мм. Графу «№ п.п.» в таблицу включать не следует.

4.6.5.Таблицу размещают после первого упоминания о ней в тексте таким образом, чтобы её можно было читать без поворота пояснительной записки или с поворотом по часовой стрелке.

4.6.6.Таблицу с большим количеством строк допускается переносить на другой лист. При переносе части таблицы на другой лист (страницу) слово «Таблица» и номер её указывают один раз справа над первой частью таблицы, над другими частями пишут слово «Продолжение». Если в КП несколько таблиц, то после слова «Продолжение» указывают номер таблицы. При переносе таблицы на другой лист (страницу) заголовок помещают только над её первой частью. Таблицу с большим количеством граф допускается делить на части и помещать одну часть под другой в пределах одной страницы. Если строки или графы выходят за формат страницы, то в первом случае в каждой части таблицы повторяется её головка, во втором случае – боковик.

4.6.7.Если повторяющийся в разных строках графы таблицы текст состоит из одного слова, его после первого написания допускается заменять кавычками со словами « То же» , а далее – кавычками. Ставить кавычки вместо повторяющихся цифр, марок, знаков, математических символов не допускается. Если цифровые или иные данные в какой–либо строке таблицы не приводят, то в ней ставят прочерк.

4.7. Примечания

4.7.1. В примечаниях к тексту и таблицам указывают только справочные и пояснительные данные. Одно примечание не нумеруется. После слова "Примечание" ставят точку. Несколько примечаний следует нумеровать по порядку арабскими цифрами с точкой. После слова "Примечания" ставят двоеточие.

4.7.2. Примечаний допускается печатать через один интервал.

Пример: данная строка содержит примечание[1].

4.8. Сокращения

4.8.1. Сокращения слов в тексте и надписях под иллюстрациями не допускаются, за исключением:
1)сокращений, установленных в ГОСТ 2.316-68 и общепринятых в русском языке;
2)сокращений, применяемых для обозначения программ, их частей и режимов работы, в языках управления заданиями, в средствах настройки программы и т.п., в том числе и обозначаемых буквами латинского алфавита.

Если в пояснительной записке принята особая система сокращения слов или наименований, то в записке должен быть приведен перечень принятых сокращений.

4.9. Приложения

4.9.1. Иллюстрированный материал, таблицы или вспомогательный текст допускается оформлять в виде приложений. Приложения оформляют как продолжение пояснительной записки на последующих страницах.
4.9.2. Каждое приложение должно начинаться с новой страницы с указанием в правом верхнем углу слова "ПРИЛОЖЕНИЕ" прописными буквами и иметь тематический заголовок, который записывают симметрично тексту прописными буквами.
При наличии в пояснительной записке более одного приложения все приложения нумеруют арабскими цифрами (без знака №). Например: ПРИЛОЖЕНИЕ 1, ПРИЛОЖЕНИЕ 2 и т.д.
4.9.3. Содержание каждого приложения при необходимости разбивают на разделы, подразделы пункты, нумеруемые отдельно по каждому приложению.
4.9.4. Нумерация страниц пояснительной записки и приложений, входящих в состав пояснительной записки, должна быть сквозной.
4.9.5. На приложения должны быть даны ссылки в основном тексте пояснительной записки.

Все приложения должны быть перечислены в листе "Содержание ".

Пример оформления приложений смотри в конце методического пособия.

---

[1] – здесь содержится текст примечания

5.Методика решения задачи

5.1.Основные этапы решения задачи

1. Постановка задачи, выбор метода решения задачи - определяет требования к будущей программе и ее характеристик (данный пункт определяется заданием на курсовой проект).
2. Проектирование программы - определение структуры программы, ее основных составных частей (функций) и принципов их взаимодействия.
3. Классификация объектов и определение иерархии классов - определяется набор классов, необходимых для решения данной задачи и определения иерархии классов, обоснование выбранного решения.
4. Определение свойств и методов каждого класса - проектирование классов.
5. Разработка алгоритмов методов классов и функций,составление блок-схем алгоритма.
6. Разработка программы на алгоритмическом языке и ее отладка на компьютере.
7. Разработка документации - создание описания принципов работы программы, описание ее составных частей, принципов их взаимодействия, классов.
8. Анализ и обработка результатов решения задачи - определение соответствия основных характерик программы с требованиями задания и составление заключения (выводов) о проделанной работе.

5.2. Этапы проектирования и структура программы

Процесс проектирования объекта связан с созданием, преобразованиями и представлением в принятой (установленной стандартами или некоторыми протокольными соглашениями) форме образа этого объекта. Образ объекта или его составных частей может создаваться в воображении человека в результате творческого процесса или генерироваться по некоторым алгоритмам. Проектирование начинается с задания на проектирование. Это задание представляется в виде тех или иных документов и является исходным (первичным) описанием объекта. Результатом проектирования, как правило, служит полный комплект документации, содержащий достаточные сведения для изготовления объекта в заданных условиях. Эта документация представляет собой окончательное описание объекта.

Проектирование – процесс, заключающийся в преобразовании исходного описания объекта в окончательное описание на основе выполнения комплекса работ исследовательского, расчетного и конструкторского характера.

Преобразование исходного описания в окончательное порождает промежуточные описания, которые являются предметом рассмотрения с целью определения окончания проектирования или выбора путей его продолжения. Такие описания называются проектными решениями.

Возможности проектирования сложных объектов обусловлены использованием ряда принципов, основными из которых являются **декомпозиция** (расчленение объекта на подобъекты, с возможностью их раздельного дальнейшего проектирования) **и иерархичность описаний объектов**(организация повторяющейся декомпозиции в виде иерархических уровней), **многоэтапность и итерационность проектирования** (в силу сложности рассматриваемых объектов), **типизация и унификация проектных решений и средств проектирования**.

Иерархические уровни описаний проектируемых объектов. Описание технических объектов должны быть по сложности согласованы с возможностями восприятия человеком и возможностью оперирования описаниями в процессе их преобразования с помощью имеющихся средств проектирования. Однако выполнить это требование в рамках некоторого единого описания, не расчленяя его на составные части, удается лишь для простых изделий. Как правило, требуется структурирование описаний и соответствующее расчленение представлений о проектируемых объектах на иерархические уровни и аспекты.

Разделение описаний по степени детализации отображаемых свойств и характеристик объекта лежит в основе блочно-иерархического подхода к проектированию и приводит к появлению иерархических уровней (уровни абстрагирования) в представлениях об объекте.

Принцип иерархичности означает структурирование представлений об объектах проектирования по степени детальности описания, а принцип декомпозиции (блочности)– разбиение представлений каждого уровня на ряд составных частей (блоков) с возможностями раздельного (поблочного) проектирования объектов на каждом уровне.

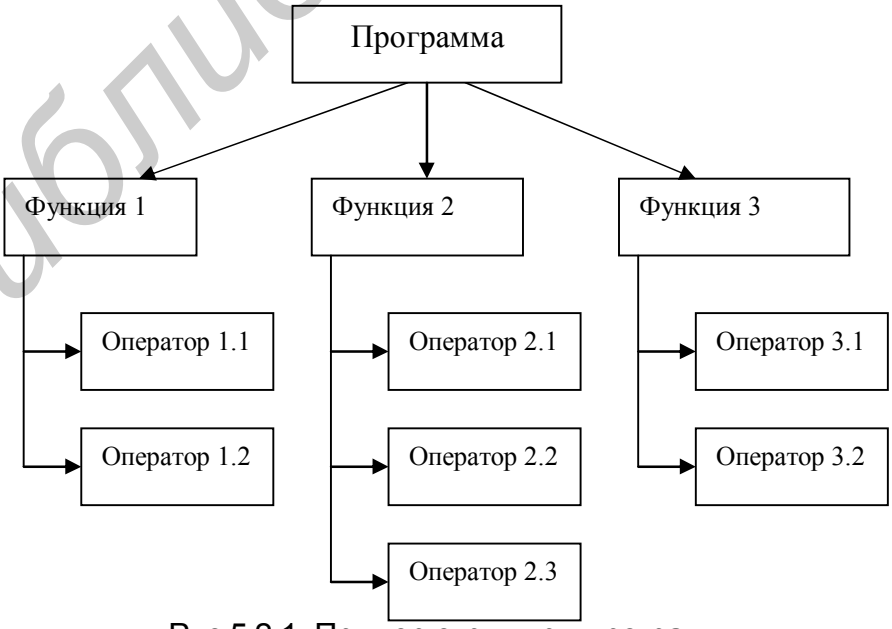


Рис.5.2.1. Пример структуры программы

Принципы декомпозиции и иерархичности описаний можно продемонстрировать на примере структуры программы, приведенной на Рис. 5.2.1, где описание программы представляется вершиной иерархического дерева, программа состоит из функций (программа декомпозируется на функции), функции состоят из операторов (функции декомпозируются на операторы) и т.д.. В больших программах сложные функции могут декомпозированы (разделены) на более простые функции, которые в свою очередь могут быть декомпозированы на еще более простые и т.д.

Вообще говоря, процесс проектирования не является линейным. В сложных проектах, детализация описаний приводит к необходимости возврата и уточнения уже пройденных этапов проектирования. В целом, это и предполагают принципы многоэтапности и итерационности проектирования.

Понятия типизация и унификация проектных решений и средств проектирования связаны с совершенствованием технологии проектирования и изготовления изделия и в рамках данного курсового проекта не рассматриваются.

5.3. Понятие алгоритма и его свойства

Определение алгоритма и его свойства. Алгоритм – это точное, т.е. сформулированное на определенном языке конечное описание последовательности действий, необходимых для выполнения некоторой работы или для решения конкретной задачи. Алгоритм обычно составляется для функций и методов, в которых осуществляются относительно сложные вычисления и преобразования (сортировка).

Свойства алгоритма

1. Дискретность – разбиение алгоритма на ряд отдельных законченных действий (шагов).
2. Точность – указание последовательности шагов.
3. Понятность – каждый исполнитель алгоритма должен однозначно написать и быть в состоянии выполнить каждый шаг алгоритма.
4. Результативность – получение результата за конечное число шагов.
5. Массовость – применимость алгоритма к решению целого класса задач.
6. Детерминированность (определенность) – однозначность результата процесса решения при заданных исходных данных.

Способы представления алгоритма

1. Формульная запись y = (2-x)+(3x-5).
2. Табличная запись.
3. Словесная запись.

Рассмотрим на примере описание алгоритма сортировки с помощью прямого включения. При этом алгоритме элементы массива мысленно делятся на уже отсортированную последовательность $a_1,\ldots,a_{i-1}$ и исходную не отсортированную последовательность. При каждом шаге, начиная с i=2 и увеличивая i каждый раз на

единицу, из исходной последовательности извлекается i-й элемент и перекладывается в нужное место готовой, т.е. отсортированной последовательности. В процессе поиска подходящего места удобно, чередуя сравнения и движения по последовательности, как бы просеивать i-й элемент, т.е. сравнивать его с очередным элементом a_j, а затем либо этот элемент вставляется на свободное место, либо a_j сдвигается (передается) вправо и процесс “уходит” влево. Для работы этого алгоритма понадобится дополнительный элемент массива, поэтому необходимо расширить диапазон индекса в описании переменной a.

Запись на алгоритмическом языке:

```
ПРОЦЕДУРА Прямое Включение.
ПЕРЕМЕННЫЕ a[0..n]: массив элементов. i, j: индексы. x: элемент.
НАЧАЛО
  ДЛЯ  i=2 ДО n ДЕЛАТЬ
    x = a[i]. a[0] = x. j = i.
    ПОКА x < a[j-1] ДЕЛАТЬ
      a[j] = a[j-1]. j = j – 1.
    КОНЕЦ.
    a[j] = x.
  КОНЕЦ
КОНЕЦ Прямое Включение.
```

5.4. Графический способ представления алгоритма (схема алгоритма)

Схема алгоритма программы – графическое представление алгоритма, в котором каждое элементарное действие представляется в виде специальной графической фигуры (блока). Последовательность выполнения действий изображается линиями и стрелками, соединяющими эти блоки.

Под блоком понимается любой конечный этап из всего вычислительного процесса.

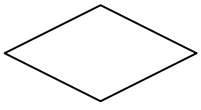
Некоторые типы блоков:

 - начальный и конечный блок;

 - информационный блок;

 - функциональный блок;

 - соединительный блок;

 - блок проверки условия.

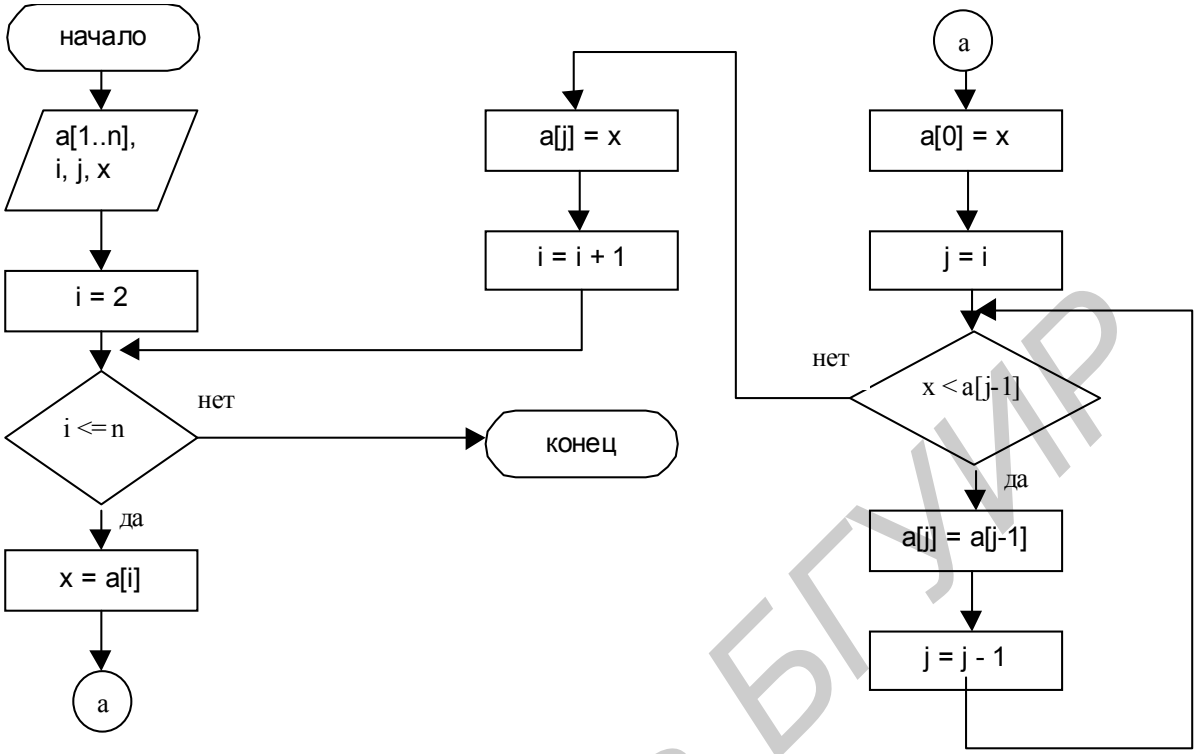


Рис.5.4.1. Пример алгоритма сортировки массива с помощью прямого включения

6. Пример разработки программы на C++

Задание. В страховой фирме ведется учет страховых полисов клиентов. Разработать интерфейсные средства (программу) поддержки ведения учета и соответствующего файла данных, используя понятия: «человек», «клиент», «страховая фирма», «страховой полис». Программа должна :
а) добавлять объекты в массив, удалять и редактировать их;
б) осуществлять сохранение результатов выполнения программы в файле и считывание их из файла;
в) выводить результаты на экран в виде таблицы.
г) осуществлять сортировку страховых полисов по свойству "код клиента", под которым он числится в архиве страховой фирмы.

Общие рекомендации: Последовательно и подробно описывать в пояснительной записке все этапы проектирования. Использовать методы структурного и объектно-ориентированного программирования. Использовать в пояснительной записке структурные диаграммы классов и объектов, блок схемы реализуемых алгоритмов.
Комментировать исходные тексты разрабатываемых программ, выделять существенные программные блоки.

Замечание. Нельзя использовать русские символы в названии переменных и функций. Символы русского алфавита можно использовать только для сообщений на экране компьютера. Допустимо выводить русские слова латинскими буквами.

6.1. Определение структуры программы, определение основных функций в программе

Определим основные функции в программе. Основные функции - это действия, определеные заданием на курсовой проект:
1)добавление нового страхового полиса,
2)реадактирование страхового полиса,
3)удаление страхового полиса,
4)вывод данных в виде таблицы,
5)сортировка страховых полисов,
6)сохранение данных в файле,
7)считывание данных из файла.

Управление программой удобнее всего реализовать через меню, которое можно создать в теле функции main. В меню будут вызываться основные функции программы.

6.2. Разработка диаграммы классов

Цель данного этапа - построение диаграммы на основе абстракции указанных понятий («человек», «клиент», «страховая фирма», «страховой полис») в виде набора соответствующих классов и определение отношений наследования между ними.

Основные задачи:

- определить набор классов, необходимых для реализации программы, на основе указанных понятий и функциональных требований к разрабатываемой программе;
- определить отношения наследования между классами таким образом, чтобы классу «страховой полис» была доступна вся информация, необходимая для решения поставленных задач;
- представить полученные результаты в виде диаграммы.

Под классом можно понимать некое множество объектов, имеющих общую структуру и общее поведение. Так как, каждое из введенных понятий определяет множество таких объектов, то для каждого понятия можно определить класс с аналогичным названием: «человек», «клиент», «страховая фирма», «страховой полис». Теперь любой конкретный объект, относящийся к одному из понятий, будет являться экземпляром соответствующего класса.

В простейшем случае наследование одного класса от другого идет по принципу "от общего к частному", с использованием отношения в родитель/потомок (иерархия is-a), когда класс (производный) заимствует структурную или функциональную часть одного или нескольких классов (базовых). Наследование вводится на основании общности рассматриваемых множеств сущностей.

Рассмотрим предложенные классы для проектирования. Сразу бросается в глаза общность классов «человек», «клиент». Клиент – это есть человек, отличающийся от всех остальных людей наличием, в данном случае, желанием застраховаться. Класс «клиент» – это частность по отношению к классу «человек», т.е. клиент всегда принадлежит множеству, определяемому понятием человек, но не каждый человек является клиентом. Наследование идет от класса «человек» к классу «клиент».

Сущность «страховой полис» образуется на основании сущностей «клиент» и «страховая фирма», т.к. для образования класса «страховой полис» необходима информация о клиенте, который страхуется, и о фирме, которая страхует этого клиента. Следовательно, производный класс «страховой полис» (класс потомок)

может быть образован от двух базовых классов-родителей (предков): «клиент» и «страховая фирма», через механизм наследования.

На основании вышесказанного можно построить диаграмму наследования классов, которая изображена на рис.6.1.

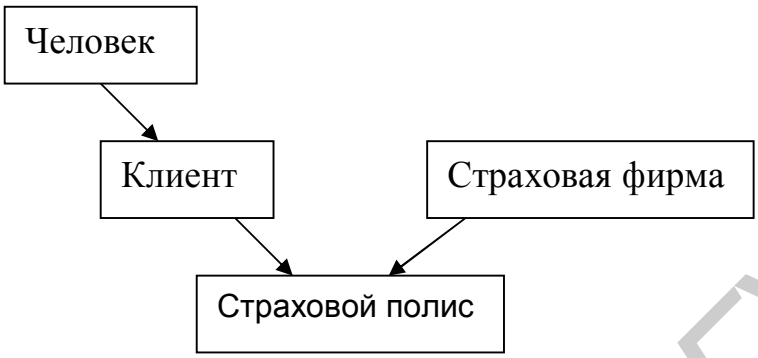


рис.6.2.1. Архитектура наследования классов

6.3. Проектирование классов, определение набора свойств и методов каждого класса

Класс ***человек*** должен содержать информацию, необходимую для решения сформулированных в задании задач. Такой информацией можно считать его имя и фамилию, место жительство (адрес), возраст. Этой информации вполне достаточно в рамках поставленной задачи. Нет необходимости учитывать всю теоретически возможную информацию типа: семейное положение, количество детей и т.д., т.к. эта информация не имеет никакого отношения к учету страховых полисов.

Класс ***клиент***, помимо информации о человеке, должен нести информацию, позволяющую отличать одного клиента от другого. Для отличия клиентов, как правило, в страховых фирмах присваивают код. Код может представляться в виде числа, но и может содержать буквенные значения (сокращенное название штата, области или района).

Класс ***страховая фирма*** должен нести информацию об отличии одной страховой фирмы от другой. Эта информация о названии фирмы, юридический адрес, номер лицензии.

Класс ***страховой полис*** должен нести информацию о конкретном страховом полисе. Эта информация о ***клиенте*** страховой фирмы, ***страховой фирме***, выдавшей данный страховой полис, сумме страховки, страховом агенте, выдавшем данный страховой полис.

Выполним проектирование классов: Класс *человек* должен содержать информацию об имени и фамилии (текст), адрес (текст), возраста (целое число).

Этой информации вполне достаточно. Предположим, что имя и фамилия будут находиться в одном поле. Тогда класс на C++ будет выглядеть:

```
class Man{
public:
char Name[8];
char Address[10];
int Age;
};
```

Класс ***клиент*** должен обладать информацией о коде клиента в архиве страховой фирмы (предположим, что это простое целое число). Основную информацию наследуем от класса Man:

```
class Client : public Man{
public:
int ClientCode;
};
```

Класс ***страховая фирма*** должен обладать информацией о названии фирмы (текст), юридическом адресе (текст), номере лицензии (целое число). Тогда:

```
class Firm{
public:
char FirmName[8];
char FirmAddress[10];
int LicNumber;
};
```

Класс ***страховой полис*** помимо информации о человеке и страховой фирме, наследуемой от классов Man и Firm, должен обладать информацией о сумме страховки (предположим целое число), о страховом агенте, выдавшем данный страховой полис (текст). Тогда:

```
class Document : public Client, public Firm{
public:
int  Summa;
char Agent[8];
}
```

Класс ***страховой полис*** будет иметь методы, выполняющие ввод значений свойств объектов с клавиатуры и вывод этих значений в виде строки, характеризующей объект для построения таблицы.

Нет необходимости в классах ***человек***, ***клиент***, ***страховая фирма***, ***страховой полис*** использовать деструкторы, т.к. в этих классах не используются динамические переменные. Конструктор может обнулять или задавать начальные значения свойств объекта.

Введем дополнительный класс ***список страховых полисов***, управляющий массивом объектов типа ***страховой полис***. Это позволит выделить каждую операцию с массивом объектов в виде отдельного метода.

Для объекта типа ***список страховых полисов*** в конструкторе может происходить загрузка результатов предыдущей работы, а в деструкторе сохранение результатов текущей работы. Объект ***список страховых полисов*** будет создаваться в начале выполнения программы и разрушаться в конце.

Определим, какие методы будут иметь используемые классы. Классы Man,Client и Firm введены как образующие информационную структуру класса Document, поэтому нет необходимости введения в них каких-либо методов. В классе Document вводим два метода InputDocument и PrintLineFromTable соответственно для поэлементного ввода и построчного формирования выводимой таблицы. Для управления массивом объектов класса Document вводится класс ListDocument. В нем создается массив Records, содержащий объекты класса Document. Количество определенных объектов (с которыми можно работать) находится в переменной countRecord.

Класс ListDocument содержит следующие методы:

1)LoadFromFile – считывает сохраненные ранее данные;
2)PrintTable – распечатывает на экране таблицу;
3)AddDocument – добавляет в массив новый страховой полис;
4)EditDocument – редактирует в массиве выбранный страховой полис;
5)DeleteDocument - удаляет из массива выбранный страховой полис;
6)Sort – сортирует массив страховой полисов;
7)SavetoFile – сохраняет данные для последующего использования.

Метод LoadFromFile вызывается из конструктора, а метод SavetoFile – из деструктора, поэтому происходит запоминание результатов предыдущей работы. Меню будет реализовано в функции **main**.

6.4. Непосредственное кодирование функций класса ListDocument на языке С и создание программы

Непосредственное кодирование функций LoadFromFile, AddRecord, PrintTable, SavetoFile и т.д. на языке С выполняется при помощи текстового редактора Borland C++. Описание оболочки редактора Borland C++ приведено в прил. 5.

Кодирование начинается с разработки алгоритма программы, непосредственного создания классов и функции main, в которой реализуется алгоритм управления программой (в данном случае меню). Затем рекомендуется создавать методы (функции) класса ListDocument в следующей последовательности.

1. **AddDocument, PrintTable** – это позволит осуществить ввод данных и их вывод и проконтролировать правильность структуры классов и набор свойств базового класса (самого сложного). Правильность работы данных функций определяется по принципу: какая информация вводится, такая же должна выводиться на экран в виде таблицы. Причем каждый новый документ должен добавляться в нижнюю строчку таблицы.
2. **EditDocument, DeleteDocument** – это сервисные функции. Проконтролировать редактирование можно, сравнивая порядковый номер редактируемого документа (должен редактироваться только выбранный документ, данные о всех остальных документах должны быть неизменны). Проконтролировать удаление можно по уменьшению количества строчек в таблице. При удалении документа не должно происходить удаления других строчек таблицы или изменения их содержимого.
3. **SavetoFile** – это функция сохраняет данные о документах в текстовом файле «1.dat». Проконтролировать правильность выполнения данной функции можно по содержанию данного файла.
4. **LoadFromFile** - это функция считывает данные о документах из текстового файла «1.dat». Проконтролировать правильность выполнения данной функции можно, сравнив содержание данного файла и содержание выводимой на экран таблицы.
5. **Sort** – это сервисная функция. Контролировать выполнение данной функции можно по значениям колонки «код клиента» в таблице. Все значения должны располагаться в порядке возрастания.

В зависимости от используемых стандартных функций к тексту программы подключаются файлы-библиотеки с описанием прототипов (с описанием стандартных функций) при помощи оператора #include.

Полный текст примера программы содержится в прил. 2. В программе "//"- это признак строчного комментария в Borland C++.

Литература

1. Фигурнов В.Э. Программное обеспечение персональных ЭВМ. – М.: Наука, 1988г.
2. Гукин Д. Word for Windows для начинающих: Пер. с англ. – Киев.: Диалектика, 1994.
3. Нортон П. Программно- аппаратная организация IBM PC. Пер.с англ.-Москва, : Радио и связь, -1992.
4. Керниган Б., Ритчи Д. Язык программирования Си. – М.: Финансы и статистика, 1985.
5. Уэйт М., Прата С., Мартин Л., Язык Си. – М.: Мир, 1988.
6. Бруно Бабе. Просто и ясно о Borland C++: Пер. с англ. – М. Бином.,1997.
7. Касаткин А.И., Вальвачев А.Н. От TURBO C к Borland C++. Мн.: Выш. шк., 1992.

Приложение 1

Варианты задания на курсовой проект

(Добавить условия на определение конструктора и деструктора и усложнить структуру классов в задании, например, введением дополнительного класса агент фирмы, который осуществлял страхование, продавца магазина, продавшего товар и т.п.)

ЗАДАНИЕ. Разработать экономическую программу, использующую наследование классов (как простое, так и множественное). Программа должна иметь возможности:

а) добавлять объекты в массив, удалять и редактировать их;

б) осуществлять сохранение результатов выполнения программы в файле и считывания их из файла;

в) выводить результаты на экран в виде таблицы;

г)осуществлять сортировку по указанному полю.

Программа предназначена для:

1)учета покупок ювелирного магазина (использовать классы: тип ювелирного украшения, ювелирное украшение, ювелирная фирма, покупка). Итоговая информация должна сохраняться в файле в виде таблицы, отсортированной по стоимости ювелирного украшения;

2)учета жилищного фонда (использовать классы: дом, квартира, человек, жилищный договор). Итоговая информация должна сохраняться в файле в виде таблицы, отсортированной по номеру жилищного договора;

3)учета стройматериалов (использовать классы: тип строй материалов, продавец, покупатель, договор о купле/продаже). Итоговая информация должна сохраняться в файле в виде таблицы, отсортированной по номеру договора;

4)учета посадок на участке в ботаническом саду (использовать классы: участок, делянка, растение, посадка). Итоговая информация должна сохраняться в файле в виде таблицы, отсортированной по номеру участка;

5)расчета закупки сырья промышленного предприятия (использовать классы: предприятие, поставщик, тип сырья, закупка). Программа должна обеспечивать расчет суммы, необходимой для закупки сырья. Итоговая информация представляется на экране в виде таблицы, отсортированной по типу сырья;

6)расчета прибыли от выполняемых работ по ремонт офиса многофилиального концерна (использовать классы: фирма, фирма по ремонту офисов, фирма заказчик, выполненный объем работ). Программа должна обеспечивать расчет прибыли с учетом налоговых выплат. Итоговая информация представляется на экране в виде таблицы, отсортированной по сумме выполненных работ;

7)расчета себестоимости изделия (использовать классы: блок, деталь, производитель детали, сборщик). Программа должна обеспечивать вывод списка деталей, используемых в данном изделии в виде таблицы, отсортированной по

стоимости и расчет суммарной стоимости всех деталей, используемых в данном изделии;

8)расчета закупки сырья промышленного предприятия (использовать классы: предприятие, поставщик, тип сырья, закупка). Программа должна обеспечивать расчет суммы необходимой для закупки сырья и выводить в виде таблицы список закупленного сырья, отсортированного по номеру накладной;

9)определения затрат рабочего времени на выполнение строительных работ (использовать классы: человек, рабочий, заказ, выполненная работа). Программа должна обеспечивать вывод результатов работы на экран и файл в виде таблицы, отсортированной по номеру заказа;

10)определения пробега автомобиля на основе путевых листов (использовать классы: водитель, автомобиль, заказ, путевой лист). Программа должна обеспечивать вывод результатов работы на экран и файл в виде таблицы, отсортированной по номеру путевого листа;

11)определения величины таможенных сборов на базе контрактов коммерческой фирмы (использовать классы: фирма, контракт, таможенный сбор, финансовые отчисления). Программа должна обеспечивать вывод результатов работы на экран и файл в виде таблицы, отсортированной по номеру контракта;

12)определения процента выхода годных изделий на основе актов приема ОТК (использовать классы: рабочий, заказ, контролер, акт). Программа должна обеспечивать вывод результатов работы на экран и файл в виде таблицы, отсортированной по номеру заказа;

13)оценки экспорта фирмы (использовать классы: фирма, покупатель, продавец, контракт). Программа должна обеспечивать вывод результатов работы на экран и файл в виде таблицы, отсортированной по номеру контракта;

14)оценки роста промышленного предприятия по данным за последние годы (использовать классы: год, вид продукции, прибыль, финансовый документ). Программа должна обеспечивать вывод результатов работы на экран и файл в виде таблицы, отсортированной по номеру финансового документа;

15)оценки продаж театральных билетов от времени года (использовать классы: время года, спектакль, театральный день, прибыль). Программа должна обеспечивать вывод результатов работы на экран и файл в виде таблицы, отсортированной по величине прибыли;

16)определения суммарной продажи проездных билетов за определенный месяц (использовать классы: месяц, тип билета, финансовый документ, прибыль). Программа должна обеспечивать вывод результатов работы на экран и файл в виде таблицы, отсортированной по величине прибыли;

17)учета выдачи стипендий студентам дневных отделений вузов (использовать классы: месяц, студент, результат предыдущей сессии, стипендия). Программа

должна обеспечивать вывод результатов работы на экран и файл в виде таблицы, отсортированной по месяцам;

18)определения величины выплат фирмы по больничным листам (использовать классы: месяц, человек, справка, больничный лист). Программа должна обеспечивать вывод результатов работы на экран и файл в виде таблицы, отсортированной по месяцам;

19)определения величины затрат фирмы на разработку новой номенклатуры выпускаемых изделий (использовать классы: подразделение фирмы, начальник, тип изделия, документация на новое изделие). Программа должна обеспечивать вывод результатов работы на экран и файл в виде таблицы, отсортированной по типу изделия;

20)управления затрат рабочего времени персонала коммерческой фирмы (использовать классы: месяц, человек, задание, результат). Программа должна обеспечивать вывод результатов работы на экран и файл в виде таблицы, отсортированной по месяцам;

21)учета амортизации промышленного оборудования (использовать классы: завод, цех, тип станка, станок). Программа должна обеспечивать вывод результатов работы на экран и файл в виде таблицы, отсортированной по типам станка;

22)управления финансовыми затратами на ремонт производственных помещений (использовать классы: завод, корпус завода, ремонтная организация, договор на ремонт помещения). Программа должна обеспечивать вывод результатов работы на экран и файл в виде таблицы, отсортированной по номерам договора;

23)учета покупок по кредитной карточке (использовать классы: человек, банка, владелец карточки, покупка). Программа должна обеспечивать вывод результатов работы на экран и файл в виде таблицы, отсортированной по дате совершения покупки;

24)расчета премиальных по результатам года (использовать классы: человек, сотрудник, организация, премия). Программа должна обеспечивать вывод результатов работы на экран и файл в виде таблицы, отсортированной по величине премии.

Приложение 2

Окончательный вариант программы

```
#include <dos.h>            //  библиотека, содержащая функцию delay
#include <stdio.h>          // библиотека, содержащая функции ввода-вывода
#include <conio.h>          // библиотека, содержащая функцию clrscr
#include <iostream.h>       // библиотека, содержащая операторы работы с классами
class Man{                  // класс Man
public:
        char Name[8];       // поле “имя”
        char Address[10];   // поле “адрес”
        int Age;            // поле “возраст”
};
class Client : public Man{  // класс “клиент” с наследованием
public:                     // от класса “человек”
        int ClientCode;     // поле “код клиента”
};
class Firm{                 // класс “фирма”
public:
        char FirmName[8];       // поле “название фирмы”
        char FirmAddress[10];   // поле “адрес фирмы”
        int LicNumber;          // поле “номер лицензии”
};
class Document : public Client, public Firm{
// класс “документ” образованного путем множественного наследования
// от классов “клиент” и “фирма”
public:
        int  Summa;             // поле “сумма страховки”
        char Agent[8];          // поле “имя страхового агента”
        void InputDocument(){   // функция ввода значений полей документа
        printf(“input Name: “);       // приглашение к вводу поля “имя”
        scanf(“%s“,Name);             // ввод значения поля “имя”
        printf(“input Address: “);    // приглашение к вводу поля “адрес”
        scanf(“%s“,Address);          // ввод значения поля “адрес”
        printf(“input Age: “);        // приглашение к вводу поля “возраст”
        scanf(“%d“,&Age);             // ввод значения поля “возраст”
        printf(“input ClientCode: “); // приглашение к вводу поля “код клиента”
        scanf(“%d“,&ClientCode);      // ввод значения поля “код клиента ”
        printf(“input FirmName: “);   // приглашение к вводу поля “имя фирмы”
```

```
        scanf("%s",FirmName);       // ввод значения поля "название фирмы"
        printf("input FirmAddress: "); // пригл. к вводу поля "адрес фирмы"
        scanf("%s",FirmAddress);    // ввод значения поля "адрес фирмы"
        printf("input LicNumber: ");  // пригл. к вводу поля "номер лицензии"
        scanf("%d",&LicNumber);     // ввод значения поля "номер лицензии"
        printf("input Summa: "); // приглашение к вводу поля "сумма страховки"
        scanf("%d",&Summa);   // ввод значения поля "сумма страховки"
        printf("input Agent: ");  // приглашение к вводу поля "имя страх. агента"
        scanf("%s",Agent);       // ввод значения поля "имя страх. агента"
        }
        void PrintLineFromTable(){ // функция вывода значений
                                   // полей документа в виде строки таблицы
        printf("%8s",Name);         // вывод значения поля "имя"
        printf("%10s",Address);   // вывод значения поля "адрес"
        printf("%5d",Age);          // вывод значения поля "возраст"
        printf("%5d",ClientCode); // вывод значения поля "код клиента"
        printf("%8s",FirmName);  // вывод значения поля "название фирмы"
        printf("%10s",FirmAddress); // вывод значения поля "адрес фирмы"
        printf("%5d",LicNumber);  // вывод значения поля "номер лицензии"
        printf("%5d",Summa);     // вывод значения поля "сумма страховки"
        printf("%8s\n",Agent);     // вывод значения поля "имя страх. агента"
        }
};
class ListDocument{          // класс "список документов"
public:
        Document Records[20];  // объявление массива классов "документ"
        int countRecord;           // количество определенных элементов в массиве
        ListDocument(){     // декларация конструктора класса "список документов"
        LoadFromFile();   // вызов функции чтения данных из файла
        };
        ~ListDocument(){    // декларация деструктора класса "список документов"
        SavetoFile(); // вызов функции записи данных в файла
        };
        void LoadFromFile(){ // функция чтения данных из файла
        FILE *in; // объявление логического имени файла,
                        // представляющего собой указатель на экземпляр
                        // структуры FILE, описанный в библиотеке stdio.h
        countRecord=0;
        if ((in = fopen("c:\\work\\cursov\\1.DAT", "rt"))== NULL){ // проверка
```

```
        printf("Cannot open input file.\n");            // на возможность
        return;                                         // открытия файла
        }
        while (fscanf(in, "%s", Records[countRecord].Name)!=EOF){
                // считывание полей класса, в случае неудачи выход из функции
        if(!fscanf(in, "%s", Records[countRecord].Address))
        break;
        if(!fscanf(in, "%d", &Records[countRecord].Age))
        break;
        if(!fscanf(in, "%d", &Records[countRecord].ClientCode))
        break;
        if(!fscanf(in, "%s", Records[countRecord].FirmName))
        break;
        if(!fscanf(in, "%s", Records[countRecord].FirmAddress))
        break;
        if(!fscanf(in, "%d", &Records[countRecord].LicNumber))
        break;
        if(!fscanf(in, "%d", &Records[countRecord].Summa))
        break;
        if(!fscanf(in, "%s", Records[countRecord].Agent))
        break;
        countRecord++;
        }
        fclose(in);
        }
void SavetoFile(){   // функция записи данных в файла
        FILE *out; // объявление логического имени файла,
                        // представляющего собой указатель на экземпляр
                        // структуры FILE, описанный в библиотеке stdio.h
        if ((out = fopen("c:\\work\\1.DAT", "wt"))== NULL){ // открытие файла
                                                            // и проверка на
                printf("Cannot open input file.\n");      //  УСПЕШНОСТЬ
                                                    // ЗАВЕРШЕНИЯ ОПЕРАЦИИ
                return ;                                    //  выход
        }
        for(int i=1;i<=countRecord;i++){    // цикл записи информации в файл
                                            // из массива элементов
       fprintf(out, "%s", Records[i-1].Name);
              fprintf(out, "%s", Records[i-1].Address);
```

```
        fprintf(out, "%d", Records[i-1].Age);
fprintf(out, "%d", Records[i-1].ClientCode);
        fprintf(out, "%s", Records[i-1].FirmName);
        fprintf(out, "%s", Records[i-1].FirmAddress);
        fprintf(out, "%d", Records[i-1].LicNumber);
        fprintf(out, "%d", Records[i-1].Summa);
        fprintf(out, "%s\n", Records[i-1].Agent);
        }
fclose(out);
}
void PrintTable(){   // функция вывода данных на экран в виде таблицы
clrscr();
// создание шапки таблицы
        printf("\n                              ");
        printf("                          Table 1");
printf("\n****************************************");
printf("****************************************");
printf("\n  Name  Address  Age  Client  Firm");
        printf("   Address  Lic  Summa  Agent\n");
// цикл формирования строк таблицы
for(int i=1;i<=countRecord;i++)
Records[i-1].PrintLineFromTable();
// нижняя часть рамка таблицы
printf("\n****************************************");
printf("****************************************\n");
printf("Press anny key for continue...");
getch();
}
void AddDocument(){ // функция добавления нового документа
clrscr();     // очистка экрана
Records[countRecord].InputDocument(); // ввод нового документа
countRecord++; // увеличение количества записей на один
}
void EditDocument(){ // функция редактирования документа
clrscr();  // очистка экрана
int i;
// выбор номера редактируемого документа
printf("input number of edit Document (1...%d) :",countRecord);
scanf("%d",&i);
```

```
        Records[i-1].InputDocument(); // ввод новых значений полей документа
}
        void DeleteDocument(){ // функция удаления документа
        clrscr(); // очистка экрана
        int i;
        // выбор номера удаляемого документа
        printf("input number of delete Document (1...%d) :",countRecord);
        scanf("%d",&i);
        //  заполнение освободившейся позиции
        if((countRecord>1)&&(i!=countRecord))
        for(int j=i-1;j<countRecord;j++)
        Records[j]=Records[j+1];
        countRecord--;// уменьшение количества записей на один
        }
        void Sort(){  // функция сортировки документа
        Document copyRecord;
        for(int j=0;j<(countRecord-1);j++)
    for(int i=j+1;i<countRecord;i++)
        if(Records[j].ClientCode>Records[i].ClientCode){
            copyRecord=Records[j];
            Records[j]=Records[i];
    Records[i]=copyRecord;
                }
        }
};
void main(void){
        ListDocument R; // создание класса "список документов"
        int i=100;
        while(i>0){
        clrscr();
        // вывод меню на экран
        printf("1: ReRead data from file\n");
        printf("2: Print Table\n");
        printf("3: Add Document\n");
        printf("4: Edit Document\n");
        printf("5: Delete Document\n");
        printf("6: Sort Document\n");
        printf("7: ReWrite data to file\n");
        printf("0: Exit \n\n");
```

```
// приглашение для ввода и ввод управляющего символа
printf("inpit location of menu: ");
scanf("%d",&i);
// обработка результатов выбора элемента меню
switch(i){
case 1 : R.LoadFromFile();  // вызов функции считывания
                            // информации из файла
break;
case 2 : R.PrintTable();  // вызов функции вывода данных
                          //на экран в виде таблицы
break;
case 3 : R.AddDocument();// вызов функции добавл. нового документа
break;
case 4 : R.EditDocument();// вызов функции редактирования документа
break;
case 5 : R.DeleteDocument();// вызов функции удаления документа
break;
case 6 : R.Sort();  // вызов функции сортировки документов
break;
case 7 : R.SavetoFile();// вызов функции записи информации в файл
break;
}
}
                                                                    }
```

Приложение 3

Образец оформления титульного листа

Министерство Образования Республики Беларусь
БГУИР
Заочный факультет
Кафедра экономической информатики

Курсовой проект
по курсу
«Основы информатики и вычислительной техники»
задание
«Разработать экономическую программу учета страховых полисов на языке С++»

Исполнитель:

Иванов И.И.
гр.901400
1 курса ЗФ,

Руководитель:

Петров П.П.

Минск 2000

Приложение 4

Образец заполнения бланка задания

БГУИР.

(название ВУЗа)

Факультет заочный .

«УТВЕРЖДАЮ»

Заведующий кафедры_____

(подпись)

«___»_____20___г.

З А Д А Н И Е

по курсовому проектированию

Студенту_____

1.Тема проекта Разработать экономическую программу учета страховых полисов (использовать классы: человек, клиент, страховая фирма, страховой полис).

2.Срок сдачи студентом законченного проекта _____

3.Исходные данные для проекта_____

Программа должна иметь возможности:

а) добавлять *объекты (экземпляры класса)* в массив, удалять и редактировать их;

б)осуществлять сохранение результатов выполнения программы в файле и считывания их из файла;

в) выводить результаты на экран в виде таблицы;

г)осуществлять сортировку по коду клиента, под которым он числится в архиве страховой фирмы.

4.Содержание пояснительной записки (перечисление вопросов, которые необходимо осветить)_____

Введение

1.Описание структуры наследования классов

2.Описание назначения, свойств и методов каждого класса

3.Описание программы .

4.Выводы .

Литература .

Приложение 1 .

5.Перечисление графического материала (с указанием необходимых чертежей и графиков)

Структура наследования классов (А4) .

Структурная схема программы (А4) .

Блок-схема алгоритма сортировки (А4) .

6.Консультант по проекту (с указанием разделов проекта) Ф.И.О. руководителя .

7.Дата выдачи задания чило, месяц и год .

8.Календарный график работ над проектом на весь период проектирования (с указанием сроков выполнения и сложности отдельных этапов)

Руководитель_____

(подпись)

Задание принял к исполнению_____

(дата и подпись студента)

Приложение 5

Работа с (IDE) Borland C++.

Интегрированная среда программирования (IDE) Borland C++.

Интегрированная среда (IDE) – это программа, имеющая встроенный редактор текстов, подсистему работы с файлами, систему помощи, встроенный отладчик, подсистему управления компиляцией и редактированием связей, а также компилятор и редактор связей. Другими словами, IDE дает возможность получить EXE-файл, не используя другие программы. IDE запускается файлом BC.EXE.

После запуска на исполнение файла запуска IDE (файл BC.EXE) на экране отображается основное окно IDE (Рис 1).

Верхняя строка окна – это главное меню. Опции меню позволяют обратиться к подменю и выбрать соответствующую команду.

Нижняя строка экрана отведена под строку состояния, где выделены назначения «горячих» клавиш, воспринимаемых на данном этапе работы.

Выбрать любую из команд меню можно одним из трех способов:

1)нажать клавишу F10 и с помощью клавиш со стрелками выбрать необходимую команду;

2)установить курсор мыши на любое ключевое слово меню и нажать левую кнопку мыши;

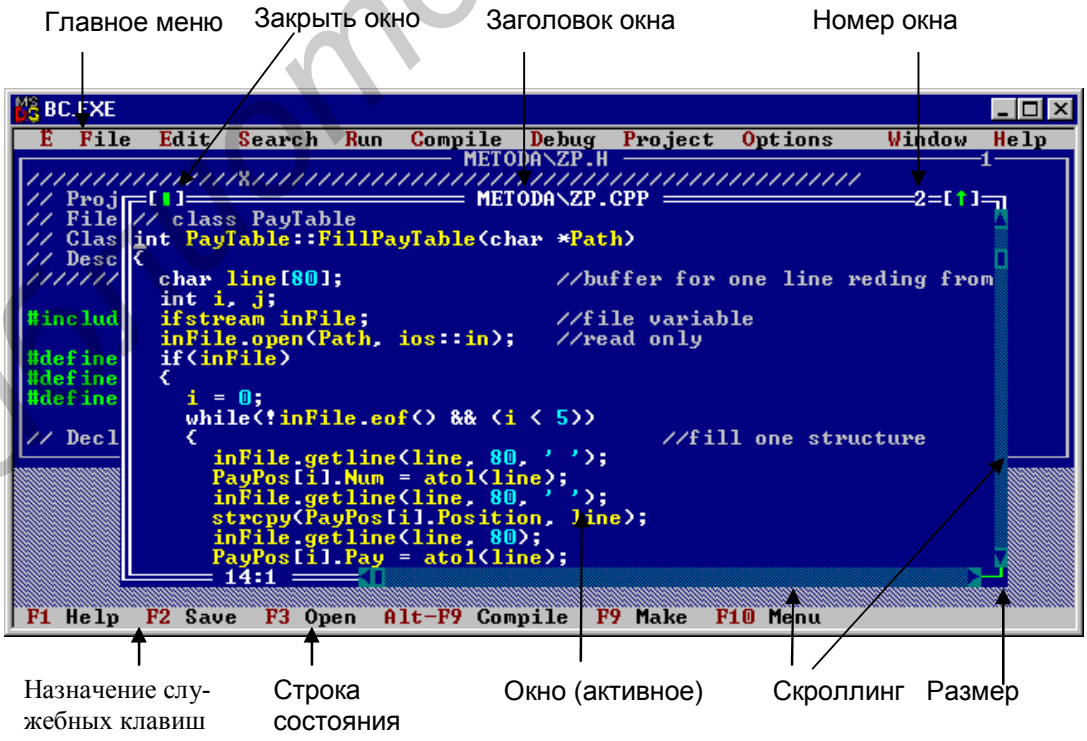


Рис. 1. Основное окно Borland C++ с двумя открытыми окнам

3)использовать «горячие» клавиши (метод скорейшего вызова команды). Одновременное нажатие клавиши Alt и «горячей» (клавиша подсвеченная другим цветом).

Окно – это ограниченная рамкой область экрана. Его можно открыть, переместить, покрыть другими окнами, изменить размеры, закрыть. Активное окно (то, которое воспринимает нажатия клавиш) обозначается двойной линией. Активное окно изображается всегда поверх других окон. В любой момент только одно окно может быть активным. Каждое окно имеет заголовок и номер, показанные в его верхней строке. Для активного окна там же расположены два управляющих поля, заключенных в квадратные скобки: поле справа используется для раскрытия окна мышью на полный экран, а поле слева – для закрытия окна. Многие окна для управления положением текста имеют по правой и нижней границам вертикальную и горизонтальную полосы прокрутки (скроллинга). Поля изменений размеров окна – это символы нижних углов рамки окна.

Операции с окнами могут выполняться тремя способами:

1)через команды главного меню Window;

2)с помощью манипулятора мышь;

3)при помощи «горячих» клавиш.

Закрыть можно только активное окно. Для этого либо выбирается в меню Windows команда Close, либо нажимается «горячая» клавиша Alt-F3, либо мышью выбирается поле [•] на окне.

Переключение между окнами выполняется так. Выбирается команд List… из меню Window или нажимается «горячая» клавиша Alt-0. Открывается окно диалога, в котором можно выбрать любое из открытых и ранее закрытых окон. Если на экране отображена хотя бы небольшая часть необходимого окна, достаточно в эту область установить мышь и нажать ее левую кнопку, чтобы окно стало активным. Циклический просмотр окон возможен при помощи клавиши F6.

Активное окно может быть раскрыто на весь экран либо выбором Window-Zoom, либо нажатием клавиши F5, либо выбором мышью поля [↑].

Для просмотра результатов выполнения программы, если вывод выполняется в текстовом режиме, используется переключение в окно вывода (Window - Output). Для просмотра результатов как в текстовом, так и графическом режимах, следует активизировать окно экрана пользователя (Window – User screen) или воспользоваться . одновременным нажатием клавиши – Alt-F5. Возврат в среду происходит при нажатии любой клавиши.

Переключение в режим редактирования выполняется автоматически при выборе команды New в меню File или при открытии файла. Для возврата из меню в режим редактирования достаточно нажать клавишу Esc.

Команды вставки и удаления (под блоком понимается выделенное подсветкой подмножество символов):

Ins – режим вставки/замены;

Del – удалить символ в позиции курсора;

Backspace – удалить символ слева от курсора;

Ctrl-Y – удалить строку;

Ctrl-N – вставить строку.

Команды работы с блоками:

Shift+клавиши со стрелками – выделение блока текста;

Ctrl-Ins – копировать блок в буффер обмена;

Shift-Ins – копировать блок из буффера обмена в текущую позицию курсора;

Ctrl-Del – удалить блок.

Shift -Del – вырезать блок в буфер обмена.

Этапы создания программы в инструментальных средах фирмы Borland.

Основные этапы создания программы в IDE Borland C++:

1)настройка опций среды программирования;

2)набор исходного текста программы;

3)компиляция программы;

4)компоновка программы;

5)отладка программы;

6)запуск программы на исполнение.

Система программирования Borland C++ включает:

1)интегрированную среду программирования (Integrated Development Environment - IDE);

2)компилятор исходного текста программы;

3)редактор связей (компоновщик);

4)библиотеки заголовочных файлов;

5)библиотеки функций;

6)программы-утилиты.

Задание опций интегрированной среды

Первым шагом при работе с IDE является настройка нужных опций (дополнительных параметров). Все опции имеют значения по умолчанию. Рассмотрим основные опции, настраиваемые с помощью команд меню Options.

Для того чтобы начать работу с IDE, прежде всего требуется задать директории, используемые текстовым тедактором, компилятором и компоновщиком рис. 2. Для этого используется команда Options\Directories (мы будем использовать формат записи Меню\Меню\…\Команда для экономии места). Поле ввода Include Directories

используется для задания директориев заголовочных файлов. В поле ввода разрешается указывать несколько директориев, разделяемых символом “;”.
Поле ввода Library Directories задает директории, содержащие объектный файл загрузчика (CO?.OBJ, где ? – это буква M, S, H, T, L, C в зависимости от используемой модели памяти) и файлы библиотек функций (.LIB). Поле ввода Output Directory задает директорий, в котором помещаются файлы с расширениями .OBJ, .EXE, .MAP. Если в поле – пустая строка, используется текущий директорий.

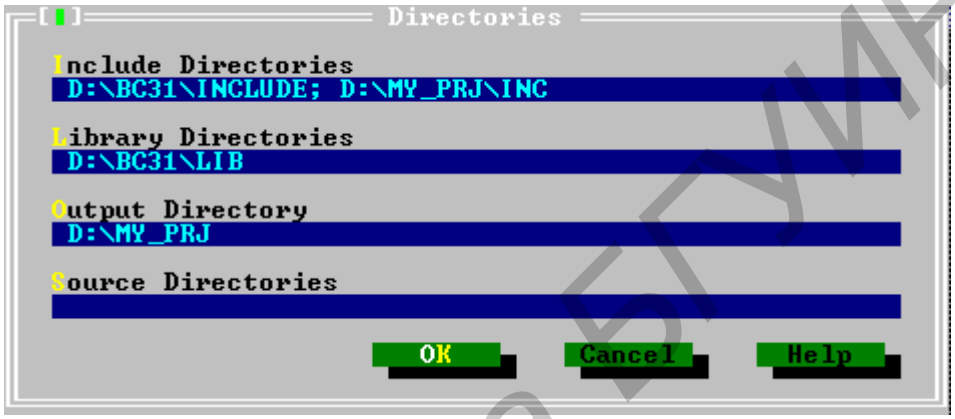


Рис. 2. Окно диалога для установки директориев

При выборе строки Options\Compiler открывается еще одно меню для настройки опций компилятора. Наиболее важные опции задаются при выборе команды Code generation. Опция считается выбранной, если она помечена символом (•) и включенной, если она помечена символом [x]. Самым важным пунктом в окне Code generation является выбор модели памяти. Для большинства программ, разрабатываемых для ОС MS-DOS, нужно выбрать SMALL модель памяти.

Набор текста программы

Следующим шагом является ввод программы с использованием текстового редактора и сохранение исходного текста программы в файле.
Набор текста программы можно выполнить встроенным или любым другим текстовым редактором. По традиции файлы, содержащие исходные тексты программ на языке C, имеют расширение имени файла .C, а на языке C++ - .CPP.

Не следует начинать компиляцию, компоновку или запуск программы без сохранения сделанного набора! Запущенная на выполнение программа может вызвать «зависание» компьютера, и сделанный набор будет потерян.
К программам-утилитам относят ассемблер, препроцессор, отладчик, программу профилирования и многие другие полезные программы.

Компиляция, редактирование связей, запуск программы на выполнение

Borland C++ включает богатейшие библиотеки функций для управлениями файлами, выполнения ввода-вывода и многих других действий. Прототипы (заголовки функций с описанием типов формальных параметров и типа возвращаемого функцией значения), символические константы и другие макро, связанные с библиотечными функциями, объединяют в заголовочные файлы, которые по традиции имеют расширение .H. Необходимые для компиляции файлы включаются в текст программы при помощи препроцессорной директивы #include. При запуске на компиляцию текст программы сначала обрабатывается препроцессором, который обрабатывает только «свои» директивы (в частности, вместо директивы «#include имя_файла» встраивается из библиотеки INCLUDE файл, имя которого задано в директиве), а затем текст программы передается непосредственно на обработку компилятору.

Компиляция исходного текста программы инициируется либо через команду Compile\Compile to OBJ, либо нажатием «горячей» клавиши Alt-F9. Команда Make EXE file также запускает программу на компиляцию и при отсутствии синтаксических ошибок автоматически запускает компоновщик для получения .EXE-файла. Еще одна возможность для запуска программы на компиляцию – команда Run\Run (Ctrl-F9). После успешной компиляции и компоновки запускается полученный .EXE-файл на выполнение.

Все сообщения об ошибках и предупреждения IDE помещает в окно по имени Message. Это окно активно после завершения компиляции. Если в программе обнаружены ошибки, включаются средства трассировки ошибок, которые связывают строки текста программы в окне редактора со строками окна Message. Перемещение высвечивания клавишами со стрелками в окне Message синхронно сопровождается высвечиванием соответствующих ошибочных строк в тексте программы. При нажатии клавиши Enter активизируется окно редактора и курсор устанавливается на ошибочную строку. Нажатие клавиши F6 (переход или активизация следующего окна) вновь делает активным окно Message.

Многофайловая компиляция

При модульном программировании не обойтись без многофайловой компиляции. При работе с большими программами намного удобнее размещать части программы не в одном, а в нескольких файлах. Каждый файл должен включать целиком одну или несколько функций. Имена этих файлов записываются в специальный файл – файл проекта, из которого IDE узнает, какие из текстовых файлов следует объединять в исполняемый (.EXE) файл.

Все необходимые для работы с файлами проектов команды включены в меню Project.

Для организации файла проекта необходимо открыть файл проекта. Для этого выполняется команда Project\Open Project… IDE активизирует специальное окно Project в нижней части экрана и открывает окно диалога, позволяющее загрузить нужный файл проекта или создать новый с заданным именем.

Если создан новый файл проекта, окно Project первоначально будет пустым. Включение файлов в проект и их удаление выполняются либо через команды Project\Add item… и Project\Delete item, либо нажатием клавиш Ins и Del, в случае если курсор размещен в окне Project. При добавлении файлов в проект открывается окно диалога, позволяющее выбрать нужный файл.

Окно Project упрощает переход от одного файла, включенного в проект, к другому при их редактировании. Для этого высвечивание перемещается на нужную строку окна Project и нажимается клавиша ENTER.

При работе со средой Borland C++ рекомендуется использовать проект, даже если программа состоит из одного файла.

Отладка программы

В процессе отладки вы можете:

1)осуществлять пошаговое выполнение программы. После прохода каждой ее строки будет производиться приостановка, позволяющая проанализировать промежуточные результаты;

2)проверять значение и местоположение (адрес) некоторой переменной в ходе выполнения программы;

3)просмотреть последовательность вызова функций в программе.

Существует два режима пошагового выполнения программы:

1)трассировка с заходом в тело функции, при встрече ее вызова в тексте программы (F7);

2)пошаговое выполнение функции (как обычной команды без захода в тело функции), при встрече ее вызова в тексте программы (F8).

Команда Run\Trace into (F7) запускает программу на отладку. Интегрированная среда высвечивает строку программы, содержащую точку входа main(). После этого нажатием клавиши F7 вызывается выполнение кода, соответствующего одной строке текста программы. Если в строке записана ссылка на функцию, начинается трассировка по тексту тела функции. При необходимости выполнения строки функции за один шаг, используется клавиша F8 (команда Run\Step over).

Для ускорения процесса отладки используется команда Run\Go to cursor (F4). Программа выполняется до строки, в которой в данный момент располагается текстовый курсор. Можно также задать режим выполнения до точки останова (через опцию подменю “Debug\Toggle breakpoint” или одновременным нажатием клавиш Ctrl

и F8, в дальнейшем будем использовать запись “Ctrl+F4”). При этом строка в точке останова подсвечивается обычно красным фоном. Снять установку точку останова можно повторным выполнением описанной команды, размещая курсор на подсвеченной строке останова.

Для наблюдения за изменением значений переменных в ходе выполнения программы используется подменю Debug\Watches\Add watch или “Ctrl+F7”. В появившемся окне Add Watch (вызов окна Add Watch можно также получить, если нажать клавишу Ins, предварительно сделав активным окно Watch) необходимо ввести имя переменной, значение которой необходимо просмотреть и нажать Ввод. Указанная переменная размещается в окне Watch, создаваемом в нижней части экрана, и в процессе отладки через это окно можно наблюдать за изменением размещенных в нем переменных. Удалить переменную из окна Watch можно при помощи клавиши “Del”, предварительно выделив ее подсветкой.

Используя опцию меню Evaluate/modify или “Ctrl+F4”, можно изменить значение переменной в процессе выполнения отладки, чтобы протестировать алгоритм с новым заданным значением. Окно этой опции “Evaluate and Modify” можно использовать и в качестве калькулятора, если записать выражения с переменными в строке “Expression” и нажать клавишу “Evaluate’ для получения результата в строке Result.

Св. план 1999, поз. 94

Учебное издание

Бахирев Андрей Владимирович,
Живицкая Елена Николаевна,
Комличенко Виталий Николаевич,
Соколов Сергей Александрович

МЕТОДИЧЕСКОЕ ПОСОБИЕ
для выполнения курсового проектироекта
по дисциплине
«Основы экономической информатики»
для студентов специальности Э.01.03.00

Редактор Е.Н. Батурчик

Подписано в печать Формат 60x84 1/16.
Бумага Печать офсетная. Усл. печ. л.
Уч.-изд. л. 4,6. Тираж 120 экз. Заказ

Белорусский государственный университет информатики и радиоэлектроники
Отпечатано в БГУИР. Лицензия ЛП №156. 220027, Минск, П. Бровки, 6