Андрей Григорьев

C для PC

Выпуск 0

Автор серии "С для PC"
АНДРЕЙ ГРИГОРЬЕВ
работает в секторе математического моделирования биологических процессов ВНИИГенетики (Москва). Выпускник Московского инженерно-физического института. Обладатель золотой медали Первой Всемирной компьютерной Олимпиады (Лондон, 1989 год). Один из организаторов Всесоюзной компьютерной Олимпиады. Автор ряда публикаций в зарубежных компьютерных журналах, таких как "Computer Applications in the Biosciences" и "Advantage" (Великобритания), TUG Lines (США). С 1989 года ведет постоянную рубрику в журнале "С Vu" (С Users Group, Великобритания), освещающем различные вопросы программирования на языке С.

C для PC

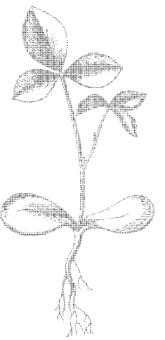
АНДРЕЙ ГРИГОРЬЕВ

ПРОГРАММИРОВАНИЕ НА ЯЗЫКЕ С ДЛЯ ПЕРСОНАЛЬНЫХ КОМПЬЮТЕРОВ

Выпуск 0

Компиляторы Microsoft C 5.0 и QuickC 1.0

Москва
Издательство "Наука"
Главная редакция
физико-математической литературы
Центр МИФИ СП "Диалог"
1990



ББК 22.18
Г83
УДК 519.85

Серия "С для PC. Программирование на языке С для персональных компьютеров" издается с 1990 года

Григорьев А. Компиляторы Microsoft C 5.0 и QuickC 1.0.—М.: Наука. Гл. ред. физ.-мат. лит., 1990. —80с.— (Серия "С для PC. Программирование на языке С для персональных компьютеров"; Вып. 0)
ISBN 5-02-014896-2 (Вып. 0)

Первая книга серии "С для PC" посвящена программированию с помощью современных версий компиляторов Microsoft C и QuickC, приведены практические примеры компиляции и отладки программ, рекомендации по обнаружению часто встречающихся ошибок и их исправлению, описание стандартных моделей памяти.

Для широкого круга пользователей персональных компьютеров.

Г $\frac{1404000000-089}{053(02)-90}$ Без объявл.

ISBN 5-02-014896-2 (Вып. 0)
ISBN 5-02-014897-0

СТОИТ ЛИ ПОКУПАТЬ ЭТУ КНИГУ ?

Что это такое?

"С для PC" - это серия выпусков, освещающих разнообразные вопросы программирования на языке *С* для персональных компьютеров. В выпусках *"С для PC"* вы сможете найти наряду с учебной и полезную справочную информацию.

Кому это нужно?

Материал, предлагаемый вашему вниманию в серии *"С для PC"*, рассчитан на широкий круг читателей. Наиболее полезным, по-видимому, он окажется для программистов, имеющих некоторые начальные знания как о самом языке *С*, так и об устройстве персонального компьютера фирмы IBM®.

Впрочем, и новички, и профессионалы наверняка найдут в каждом выпуске новую и полезную для себя информацию.

Что содержится в этом выпуске?

Существенная часть серии *"С для PC"* будет посвящена программированию с помощью компилятора *Microsoft® C 5.0* (и *QuickC®*) и ориентирована на использование его библиотечных функций.

Поэтому в нулевом выпуске основное внимание уделено некоторому "нулевому циклу" работы с *Microsoft C* и *QuickC*: их установке, предназначению отдельных программ, правильному заданию окружения, обзору стандартных моделей памяти и ряду других вопросов.

Здесь приведены практические примеры компиляции и отладки программ, а также собраны некоторые полезные советы и наиболее часто встречающиеся ошибки вместе с рекомендациями по их исправлению.

О чем будут рассказывать следующие выпуски серии "С для PC"?

Следующие выпуски *"С для PC"* будут посвящены преимущественно программированию, в частности решению задач, связанных с экранной графикой, управлением таймером и громкоговорителем, использованием различных периферийных устройств, созданием интерфейса пользователя и т. д. Кроме того, будут подробно описаны особенности практической работы с компилятором и другими утилитами комплекта *Microsoft C 5.0*.

Каждый выпуск серии *"С для PC"* будет снабжен обширным приложением, содержащим необходимую программистам информацию, сведенную воедино из разнообразных руководств по языку *C*, операционной системе *MS-DOS®* и технических описаний персональных компьютеров и периферийных устройств.

Каким образом можно приобрести следующие выпуски?

Вместо того чтобы искать очередные выпуски серии *"C для PC"* в книжных магазинах, вы можете заказать их, воспользовавшись информацией, приведенной в конце этого выпуска.

Стоит ли покупать эту книгу?

Теперь, я думаю, вам не составит труда сделать окончательный выбор.

НЕМНОГО О ВЫБОРЕ КОМПИЛЯТОРА

Среди программистов в Советском Союзе в настоящее время наиболее известны два *C*-компилятора для IBM PC®-совместимых персональных компьютеров: *Microsoft C* фирмы Microsoft Corporation и *Turbo C*® фирмы Borland International. Можно было бы добавить к этой паре *QuickC*, также производимый фирмой Microsoft, но я склонен рассматривать этот продукт скорее в качестве чрезвычайно полезного дополнения к основному оптимизирующему компилятору, нежели как самостоятельный полноценный компилятор.

Года четыре назад ситуация на рынке *C*-компиляторов была совершенно иной. Оба теперешних лидера (*Turbo C* тогда еще назывался *Wizard C*) в один голос объявляли о своей полной совместимости с компилятором *Lattice C*, который к сегодняшнему дню уже почти забыт, а тогда был, наверное, самым популярным в мире.

Фирме Borland удалось прорваться к лидерству, когда она перенесла свою замечательную оболочку компилятора *Turbo Pascal*® в компиляторы для других языков (как похожи друг на друга *Turbo BASIC*® и *Turbo Prolog*®, *Turbo C* и *Turbo Assembler*®). Это был один из значительных шагов на пути к созданию современных сред (и средств) программирования.

Фирма Microsoft, со своей стороны, совершила скачок к первому месту, выпустив символьный отладчик программ в исходных текстах (symbolic source level debugger, debug - "ловля блох" - эквивалентен русскому термину "отладка") - *CodeView*®, который произвел настоящую революцию в индустрии программирования. Сейчас даже

трудно представить себе, каких дополнительных усилий стоила в те времена отладка программ разработчику, действовавшему буквально вслепую и щедро "увешивавшему" свои тексты дополнительным вводом-выводом, сообщавшим о возможных ошибках.

Современный *C*-компилятор уже обязательно включает в себя удобную оболочку, совмещающую текстовый редактор с символьным отладчиком. Люди, которым эти вещи не кажутся необходимыми и ощущающие некоторую стесненность в средствах, могут приобрести самый дешевый (но не самый плохой) компилятор - *Power C* (MIX Software). Однако, если вы еще не сделали выбора между *Microsoft C* и *Turbo C*, я рекомендую остановиться на первом, а точнее - на *Microsoft C* версии 5.0 (или 5.2), включающем в себя и упомянутое дополнение - *QuickC 1.0* (или даже 2.0). Попробую объяснить причины.

Во-первых, основной оптимизирующий компилятор *Microsoft C* отделен от оболочки *QuickC*, которую поддерживает свой более быстрый компилятор. В таком раскладе *QuickC* является прекрасным средством быстрого создания (то есть написания и отладки) программ, а окончательный оптимизационный лоск наводится основным компилятором *Microsoft C*. (Ограничение на среднюю модель памяти при работе с *QuickC* 1.0 снято в версии 2.0, но более подробно мы поговорим о ней в последующих выпусках серии *"C для PC"*.)

Во-вторых, я почти не сталкивался с проблемой аллокирования памяти при работе с *QuickC*, тогда как оболочка *Turbo C* жадно поглощает более 300 килобайт оперативной памяти на свои нужды, оставляя программисту какие-то крохи. Таким образом, при работе над сколь-нибудь серьезным проектом вам придется запускать компилятор *Turbo C* с командной

строки и искать "блох" в вашей программе с помощью отладчика *Turbo Debugger*® (примерно таким же образом раньше приходилось работать с прежней версией компилятора *Microsoft C* - 4.0 и его отладчиком *CodeView*).

В-третьих, я не считаю абсолютно правильным использование отдельного драйвера экранной графики *BGI* (*Borland Graphics Interface*) в *Turbo C*. Он, правда, обеспечивает доступ к несколько более широкому спектру видеоадаптеров, чем графическая библиотека *Microsoft C*, но, очевидно, в первую очередь представляет удобства не программисту, а фирме Borland, поставляющей *BGI* со всеми своими компиляторами.

В-четвертых, ошибки в самих *Microsoft C* и *QuickC* почти не портили мне нервов, тогда как одна из "блох" в *Turbo C*, связанная с обработкой многомерных массивов, стоила мне и моим коллегам многих часов труда и недоумения.

Наверное, этих аргументов достаточно.

Впрочем, мне не хотелось бы казаться догматиком - я отнюдь не рекламный агент фирмы Microsoft и могу перечислить ряд недостатков ее компиляторов. Но я намереваюсь это сделать в более развернутой форме в следующем выпуске, не нарушая сейчас общего пафоса, присущего этому тексту.

Вообще говоря, для любого пользователя компьютера хороша та программа, к которой он привык. Последующие версии только укрепляют уверенность человека в правильном выборе. Все это справедливо и для программистов, не исключая, разумеется, автора этих строк. Первым встретившимся мне *C*-компилятором оказался именно *Microsoft C*, поэтому я и буду рассказывать преимущественно о нем. В дальнейшем мне

бы хотелось поделиться своими соображениями и о других компиляторах, да и, кроме того, язык *C* есть нечто большее, чем любой конкретный компилятор вместе со всеми его библиотеками.

Microsoft C 5.0 и QuickC 1.0

Перед тем как начать наше знакомство с *Microsoft C* и *QuickC*, мы должны сделать одно существенное предположение, а именно: допустим, что вы стали владельцем комплекта дистрибутивных дискет *Microsoft C 5.0* или *QuickC*. Излагаемый ниже материал, впрочем, не потеряет своей ценности и в том случае, когда какой-то из этих продуктов уже установлен на жестком диске вашего компьютера.

Итак, начнем с установки компилятора.

SETUP.EXE - так называется программа, предназначенная для установки *Microsoft C 5.0* на ваш компьютер. (Если же вы обладаете дистрибутивными дискетами *QuickC* 1.0, то по поводу конкретных деталей его установки обратитесь к приложению А.) Эта программа, как и подобает, расположена на одноименном диске (номер 1) и очень проста в обращении. Вам достаточно только вызвать ее:

```
A> SETUP
```

и вся необходимая информация (за исключением некоторых деталей, которые мы, несомненно, упомянем) появится на экране. Вот как это будет выглядеть:

```
Microsoft (R) C Compiler Setup Program
(C) Copyright Microsoft Corp. 1986, 1987

setup <dest> <mem> [C40] [<\bin> <\incl> <\lib> <\src>]

   <dest>  Destination directory.
           Drive letter followed by colon.
           May include sub-directory specifications.

   <mem>   Memory model(s). S = small, M = medium,
           C = compact, and/or L = large.

   [C40]   C40 = C 4.0 argc/argv compatibility
           desired; no argument = No C 4.0 argc/argv
           compatibility.
```

```
[<\bin> <\incl> <\lib> <\src>]
       Optional subdir names created under <dest>.
       Args start with "\". Defaults = \bin,
       \include, and \lib.
       \src files are not loaded by default.

setup c:\new s
       /* small model, files loaded into c:\new\bin,
       c:\new\include, c:\new\lib.              */

setup c: s l \b5 \l5 \i5 \s5
       /* small and large model, files loaded into
       c:\b5, c:\l5, c:\i5, and startup sources in
       c:\s5.                                   */
```

Предположим, что мы решили назвать каталог, в котором будет располагаться все хозяйство *Microsoft C 5.0*, кратко и недвусмысленно - C:\MS5. Это название и явится параметром <dest> для программы SETUP.

Следующий параметр определяет выбор моделей памяти, необходимых вам для работы. Не вдаваясь в их подробное рассмотрение (дополнительную информацию о моделях памяти вы сможете найти в приложении B), заметим, что для нормальной работы достаточно ограничиться двумя моделями - *small* и *medium*. Большинство программ помещается в *small model*, предоставляющей по 64 килобайта и для текста программы, и для ее данных. Эта модель принимается по умолчанию при работе с основным драйвером *Microsoft C 5.0* - программой CL.EXE и с драйвером *QuickC* - программой QCL.EXE. В *medium model* текст может превышать 64 килобайта (эта модель вмещает программы с текстом, занимающим объем вплоть до 1 мегабайта),

однако объем данных опять ограничен 64 килобайтами. Эта модель используется при работе в среде *QuickC*, и это вполне достаточное обоснование для ее выбора.

Выбор моделей памяти приведет к появлению на твердом диске нескольких комбинированных библиотек (в количестве, пропорциональном числу моделей и числу пакетов математических операций, выбрать которые нам предстоит позднее). Создание этих комбинированных библиотек, сопряженное с затратами места на "винчестере", окупается экономией времени, затрачиваемого компоновщиком (так мы будем называть linker) на поиск нескольких нужных библиотек в том случае, если они не сведены воедино.

Допустим, что мы выбрали три модели памяти - две уже упомянутых плюс *large model*, в которую можно поместить любую программу (грубо говоря, с текстом до 1 мегабайта и данными того же объема). Соответственно, три следующих параметра SETUP - это буквы *s*, *m* и *l*.

Для совместимости с *Microsoft C 4.0*, а точнее, для возможности пользоваться OBJ-файлами, созданными с помощью этого компилятора и предназначенными для обработки командной строки аргументов с использованием шаблонов ДОС для имен файлов wild-card characters (* и ?), добавим к имеющимся аргументам аббревиатуру C40. Хотя, откровенно говоря, без этой совместимости можно вполне обойтись.

Четыре последних (необязательных) аргумента указывают имена подкаталогов, куда будут отправлены файлы *Microsoft C 5.0*. Как видите, каталог C:\MS5 остается пустым (никто, конечно, не мешает вам записать туда что-либо). Несмотря на эту пустоту, такое расположение файлов *Microsoft C 5.0* достаточно удобно для ориентации в дереве каталогов, например при

пользовании утилитой Питера Нортона *NCD - Norton Change Directory.*

Поскольку эти аргументы необязательны, мы их смело опустим. Укажем только, что без указания аргумента <\src> у нас не образуются дополнительные подкаталоги вида \src\model и мы не сможем получить исходные тексты процедур начальной загрузки программ (startup code). Ну и ладно, обойдемся пока без них.

Итак, в нашем случае вызов установочной программы выглядит следующим образом:

```
A> SETUP C:\MS5 S M L C40
```

Все - теперь нам остается только ответить на несколько вопросов и набраться терпения, ожидая конца процедуры установки *Microsoft C 5.0* на твердый диск.

Первые три вопроса задаются по поводу использования вами пакетов математических операций. Вы можете выбирать любые из трех вариантов: пакет для сопроцессора (*8087/80287 floating point math package*), пакет для его эмулятора (*Emulator floating point math package*), использующего, тем не менее, сопроцессор при его наличии, и пакет *Alternate floating point math package*, создающий в отсутствие сопроцессора самые быстрые программы в ущерб их точности (по сравнению с эмулятором).

Предположим, что мы ограничимся первыми двумя вариантами. Для этого достаточно только ответить *Y* на соответствующие вопросы программы SETUP.

Далее, она поинтересуется у нас, хотим ли мы включать графическую библиотеку в комбинированные библиотеки для выбранных нами моделей памяти. Это обойдется нам

примерно в 50 килобайт для каждой из библиотек, зато нам не придется указывать всякий раз библиотеку GRAPHICS.LIB при создании программ, использующих графику.

Последний вопрос, который задает программа SETUP, - хотим ли мы после построения комбинированных библиотек стереть с диска их компоненты, которые более нам не нужны? Я думаю, что хотим.

Больше установочная программа вам не понадобится, за исключением разве что случаев, когда указанных моделей памяти окажется недостаточно для вашей работы (тогда SETUP не будет создавать все заново, а только добавит к нашим библиотекам недостающую) или когда вы случайно отформатируете твердый диск на своем компьютере. А пока мы отложим в сторону дистрибутивные дискеты и посмотрим, что же делать дальше.

ЧТО ДАЛЬШЕ ?

После того как программа SETUP выполнит все необходимые действия, на экране появится надпись **Done!** Это означает, что *Microsoft C 5.0* установлен для работы на вашем компьютере и вы можете начинать программировать.

Однако имеет смысл предварительно ознакомиться со структурой компилятора, что может облегчить понимание принципов его работы и предостеречь от возможных ошибок.

Посмотрим на содержимое каталога C:\MS5\BIN после выполнения программы SETUP. Назначение некоторых из этих файлов кратко описано в приведенной ниже таблице:

CL.EXE	Драйвер *Microsoft C 5.0*
CL.ERR	Сообщения об ошибках CL.EXE
CV.HLP	Подсказка для CL.EXE
QC.EXE	*Microsoft QuickC*
QC.HLP	Подсказка для *QuickC*
QCL.EXE	Драйвер *QuickC*
QCL.HLP	Подсказка для QCL.EXE
LINK.EXE	Компоновщик (linker)
LIB.EXE	Программа управления библиотеками

QLIB.EXE	Утилита, создающая библиотеки типа *Quick library*
CV.EXE	Отладчик *Microsoft CodeView*
CV.HLP	Подсказка для CV.EXE
CVPACK.EXE	Упаковщик программ, содержащих символьную информацию для *CodeView*
MAKE.EXE	Утилита, поддерживающая разработку и видоизменение программ
ERROUT.EXE	Утилита, перенаправляющая вывод ошибок (stderr) с экрана в файл или на принтер
SETENV.EXE	Утилита, увеличивающая размер окружения ДОС
MOUSE.COM	Драйвер "мыши"

Мы рассмотрим более подробно многие из этих программ и утилит в последующих выпусках серии *"С для PC"*. А сейчас продолжим краткое ознакомление с результатами работы программы SETUP.

В каталоге C:\MS5\BIN образовалось два подкаталога - C:\MS5\BIN\PATCH и C:\MS5\BIN\SAMPLE. Первый из них, названный весьма откровенно "patch" (по-английски "заплата"), содержит файлы, необходимые для осуществления небольшого трюка, нейтрализующего ошибку операционной системы *PC-DOS 3.20*, и на них мы не станем обращать внимания. Во втором же находятся некоторые примеры текстов программ и batch-файл DEMOB.BAT, предназначенный для популярного изложения возможностей мощного символьного отладчика *CodeView* - одного из предметов гордости

фирмы Microsoft. Кроме того, здесь приведены части файлов AUTOEXEC.BAT (NEW-VARS:BAT) и CONFIG.SYS (NEW-CONF.SYS), обеспечивающие правильную установку переменных окружения и некоторых других параметров, необходимых для успешной работы компилятора.

Помимо указанных каталогов на вашем "винчестере" образовались каталоги C:\MS5\INCLUDE и C:\MS5\INCLUDE\SYS. В них содержатся так называемые header-файлы (или include-файлы, включаемые в текст программы директивой #include). От слова header ("заголовок") и образовано расширение этих файлов - ".H". В них записаны некоторые стандартные макросы, константы и прототипы функций библиотеки *Microsoft C 5.0*.

В каталоге C:\MS5\LIB располагаются библиотеки и несколько файлов с расширением .OBJ, необходимых для создания библиотек. Уточним назначение некоторых из этих файлов (обозначения: *E - Emulator floating point math package, 87 - 8087/80287 floating point math package, S - small model, M - medium model, L - large model*):

SLIBCE.LIB	*E, S*
MLIBCE.LIB	*E, M*
LLIBCE.LIB	*E, L*
SLIBC7.LIB	*87, S*
MLIBC7.LIB	*87, M*
LLIBC7.LIB	*87, L*
GRAPHICS.LIB	Графическая библиотека
GRAPHICS.QLB	Графическая библиотека типа *Quick library*

BINMODE.OBJ	Изменение моды чтения файлов по умолчанию (с текстовой на бинарную)
SETARGV.OBJ	OBJ-файл, содержащий процедуру расширения wild-card characters (* и ?) для имен файлов, передаваемых с командной строки
SVARSTCK.OBJ	OBJ-файл, включаемый в программу с изменяемым размером стека, *S*
MVARSTCK.OBJ	OBJ-файл, включаемый в программу с изменяемым размером стека, *M*
LVARSTCK.OBJ	OBJ-файл, включаемый в программу с изменяемым размером стека, *L*
MOUSE.LIB	Для примера - одна из библиотек, созданных с помощью программы LIB.EXE

В каталоге C:\MS5\MIXLANG находятся файлы F4COMPAT.BAT, LDBGMSG.OBJ и MDBGMSG.OBJ, предназначенные для создания на основе *C*-программ библиотек, совместимых с компилятором *Microsoft FORTRAN* 4.0.

Кроме всех вышеперечисленных каталогов программа SETUP создает пустой каталог C:\MS5\TMP, в котором будут записываться временные файлы, возникающие в процессе работы компилятора.

Удобнее, однако, использовать вместо настоящего твердого диска виртуальный, то есть тот, что создается, например, с помощью драйвера VDISK.SYS или RAMDRIVE.SYS. Виртуальный диск работает, что

называется, "со скоростью памяти", тем самым сокращая затраты на чтение и запись временных файлов и, следовательно, повышая общую скорость разработки ваших собственных программ. Разумеется, необходимо учитывать характеристики вашего "винчестера", а также объем оперативной памяти, остающийся после установки виртуального диска.

Фирма Microsoft рекомендует использовать виртуальный диск объемом, равным удвоенному размеру текста компилируемого файла. Но, поскольку этот диск устанавливается при включении компьютера, целесообразно выбрать его емкость один раз и надолго, а не перезагружать операционную систему при каждой компиляции.

Как правило, виртуальный диск емкостью 64 килобайт (принятой по умолчанию для указанных драйверов) умещает в себя все временные файлы; при этом на нем хватает места для записи нескольких мелких и часто используемых программ, а свободной оперативной памяти остается еще довольно много.

Если же ваш компьютер обладает расширенной памятью (extended memory), то подобной проблемы не возникает вообще - все временные файлы сбрасываются на виртуальный диск, создаваемый теми же драйверами с опцией */E* (или */A*) в области extended memory.

Каким образом компилятор узнает о том, куда именно нужно обращаться по поводу временных файлов? Этот вопрос мы рассмотрим в следующем разделе, посвященном окружению компилятора, или иначе - его среде.

ОКРУЖЕНИЕ

Окружение компилятора представляет собой элемент окружения ДОС (DOS environment), определяемый командами PATH и SET. Последняя из этих команд позволяет задать ряд переменных окружения, необходимых для правильного функционирования программ *Microsoft C 5.0*.

Эти переменные указаны в приведенной ниже таблице:

Переменная	Назначение
LIB	Указывает путь к библиотекам
INCLUDE	Указывает путь к header-файлам
TMP	Указывает путь к каталогу для временных файлов, создаваемых в процессе компиляции
CL	Задает опции драйверов CL.EXE и QCL.EXE, автоматически подключаемые к тем, что были вызваны с командной строки

Первые две переменные - LIB и INCLUDE - необходимы для работы как драйвера CL.EXE, так и *QuickC*. Без их правильного указания компилятор не сможет добраться до содержимого header-файлов, включаемых в текст ваших программ с помощью директивы #include, а компоновщик не сумеет отыскать стандартные библиотеки.

В нашем случае эти переменные следует определить в файле AUTOEXEC.BAT следующим образом:

```
SET LIB=C:\MS5\LIB
SET INCLUDE=C:\MS5\INCLUDE
```

Если же вы хотите добавить к этим двум каталогам еще какие-либо каталоги, в которых будут содержаться библиотеки и header-файлы, добавьте их названия в соответствующие команды SET через точку с запятой.

Третья переменная окружения - TMP - необходима только для работы драйвера CL.EXE, тогда как *QuickC* может функционировать и без нее. Как уже упоминалось, можно отвести для складирования временных файлов виртуальный диск (RAM disk), и тогда определение TMP необходимо записать как

```
SET TMP=D:\
```

если виртуальный диск обозначен буквой D.

Последняя переменная - CL, как следует из таблицы, необходима только для работы драйверов CL.EXE и QCL.EXE. Если, например, вы хотите, чтобы при работе всегда создавались максимально оптимизированные программы и вам лень постоянно набирать опцию */Ox*, либо она все время выскакивает за пределы 128 символов, отведенных для командной строки, вставьте в файл AUTOEXEC.BAT строку

```
SET CL=/Ox
```

В этой строке нельзя использовать опции, включающие знак равенства (как в случае /D<name>[=text], определяющем дополнительные макросы) и символы для множественных имен файлов(* и ?). Более подробное описание опций драйверов CL/QCL будет приведено в одном из следующих выпусков.

Нам осталось совсем немного - указать с помощью команды PATH путь к программам *Microsoft C 5.0*.

Поскольку все EXE-файлы, как мы видели, располагаются в каталоге C:\MS5\BIN, достаточно просто включить ее в PATH, уже имеющийся в файле AUTOEXEC.BAT.

```
PATH ...; C:\MS5\BIN;...
```

После этой модификации файла AUTOEXEC.BAT заглянем ненадолго в файл CONFIG.SYS, чтобы удостовериться в том, что параметр FILES имеет значение 20, а число буферов BUFFERS - не менее 10. На этом процесс установки окружения можно считать завершенным - при включении компьютера *Microsoft C 5.0* будет обладать информацией, достаточной для его нормальной работы.

Проверим теперь компилятор в деле. Для этого возьмем какую-нибудь программу из числа примеров, находящихся в каталоге C:\MS5\BIN\SAMPLE, например GRDEMO.C, и попробуем ее откомпилировать.

КОМПИЛЯЦИЯ GRDEMO.C

Я хотел бы начать эту часть с небольшого замечания. Если какая-либо процедура из описанных ниже приведет к ошибке *QuickC*, обратитесь к приложению С или последней главе выпуска, в которой приведены наиболее часто встречающиеся ошибки и рекомендованы способы их исправления.

Мы начнем наши первые опыты с *Microsoft C 5.0*, используя компилятор *QuickC*. Основное его предназначение - поддержка быстрого написания и отладки программ, которые в дальнейшем можно привести в более эффективную форму основным оптимизирующим компилятором с помощью драйвера CL.EXE.

QuickC использует для работы несколько более быстрый компилятор - QCL.EXE, а также включает в себя довольно удобный текстовый редактор и символьный отладчик, отдаленно напоминающий *CodeView*.

Помимо этого *QuickC* снабжен подробным подсказчиком, содержащим массу полезной информации как о самом компиляторе, так и о библиотечных функциях Microsoft.

Итак, перейдем в каталог C:\MS5\BIN\SAMPLE и вызовем *QuickC*, введя с командной строки ДОС команду QC. Если в нашем распоряжении менее 300 килобайт свободной оперативной памяти, *QuickC*, не сумев загрузиться, выдаст следующее сообщение:

```
Program too big to fit in memory
/* Программа слишком велика, чтобы уместиться
в памяти */.
```

В этом случае освободите оперативную память вашего компьютера от "лишних" резидентных программ и попробуйте повторить действия, перечисленные в предыдущем параграфе. После того как *QuickC* успешно загрузится в память, вы увидите на экране окно его редактора.

Теперь вы можете, пользуясь средствами редактора, ввести текст вашей программы или загрузить любой уже имеющийся на диске файл. Мы начнем с первого варианта и введем в редактор классический текст программы для начинающих программировать на языке *C* - HELLO.C:

```
#include <stdio.h>
main()
{
  printf("Hello, world!\n");
}
```

Теперь, нажав одновременно клавиши ALT и R, мы вызовем свешивающееся вниз с первой строки меню **Run**. Выбрав в этом меню опцию **Start** или **Continue**, мы можем отправить нашу программу на компиляцию. Выбору этих опций, как видно из меню, эквивалентны по своему действию так называемые shortcut keys - комбинации клавиш (в данном случае - SHIFT+F5 и F5 соответственно).

Итак, каким-либо из этих способов мы начали компилировать нашу программу. С экраном произойдут некоторые изменения: в появившемся большом белом окне побегут номера пройденных компилятором строк, потом в одной из строк в верху экрана возникнет надпись **Linking** и на экране появятся слова:

```
Hello, world!
Program returned (14). Press any key.
```

Как видите, все очень просто. Заметим только, что странный код возврата - 14 - в данном случае соответствует числу символов в строке для *printf*. Вообще, коды возврата в среде *QuickC* часто довольно трудно интерпретировать, поэтому обычно приходится просто игнорировать их.

Теперь мы перейдем к примеру, указанному в названии, - программе GRDEMO.C. Для этого вызовем меню **File**, нажав одновременно клавиши ALT и F, и выберем опцию **Open**. В появившемся окне будет приведен список файлов с расширением .C, находящихся в этом каталоге.

Нажав клавишу TAB, мы переместим курсор в это окно и выберем файл GRDEMO.C. Перед тем как поместить его в окно редактора, *QuickC* спросит, хотим ли мы сохранить файл UNTITLED.C ("неозаглавленный"). Таким именем *QuickC* назвал нашу программу HELLO.C. Это произошло оттого, что мы не сохранили текст программы на диске. Если сейчас мы захотим это сделать и нажмем на клавишу ENTER или Y, в очередном окошке нам будет предложено изменить название UNTITLED.C на нечто более осмысленное. Если же мы нажмем на клавишу N, *QuickC* забудет о нашем первом опыте. В обоих случаях дело кончится тем, что вместо краткого текста, приведенного выше, в редактор загрузится файл длиной 922 строки.

Это и есть программа демонстрации графических возможностей библиотеки Microsoft. Мы еще займемся ее подробным рассмотрением, когда будем изучать программирование графических функций, а сейчас

просто нажмем клавишу F5, начав компиляцию GRDEMO.C.

Эта процедура на сей раз займет немного больше времени, да и результат окажется не столь триумфальным - программа попросту не пройдет стадию **Linking**. В нижней части экрана появится окно, в котором будут перечислены ошибки (более 20) одного и того же типа:

```
error C2175 (порядковый номер ошибки of 21)
'имя функции' : unresolved external
```

В чем же дело? Оказывается, мы пропустили одну операцию, очень важную при работе с *QuickC*, - создание программного списка (Program List).

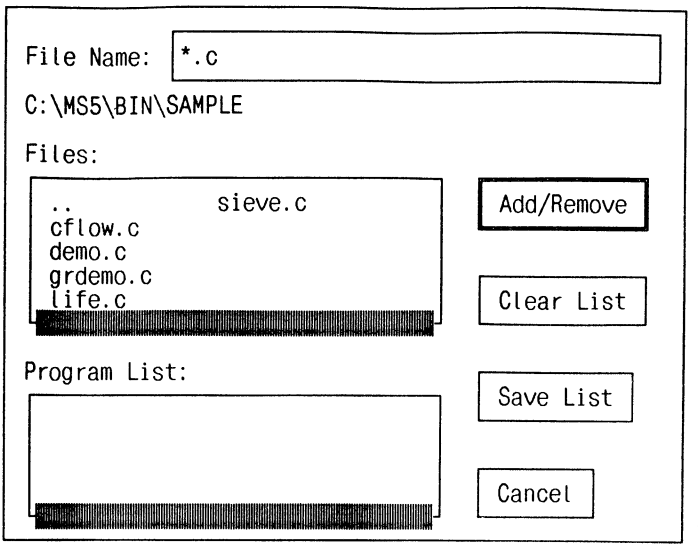
Во-первых, эта процедура необходима для создания программ, текст которых состоит более чем из одного файла. Однако вся программа GRDEMO состоит из одного-единственного файла GRDEMO.C. Остается "во-вторых".

А во-вторых, без создания Program List невозможно использовать функции, не входящие в так называемое "ядро" *QuickC* (*QuickC core library*).

К числу таких функций как раз и относятся функции графической библиотеки, до которых в отсутствие программного списка не добрался компоновщик.

Чтобы создать программный список, мы опять вызовем меню **File**, нажав одновременно клавиши ALT и F, и выберем опцию **Set Program List**. В следующем окошке введем GRDEMO, после чего *QuickC* переспросит нас, действительно ли мы намерены создать программный

список с именем GRDEMO.MAK. Ответив *Y*, мы с вами на этот раз обнаружим перед собой более сложную конструкцию окон, которую имеет смысл проиллюстрировать на рисунке:



Войдя с помощью нажатия клавиши TAB в окно **Files**, выберем файл GRDEMO.C. После нажатия клавиши ENTER он переместится в окно **Program List**. Если бы наша программа состояла из нескольких модулей, мы аналогичным образом добавили бы их в **Program List**.

Удаление лишних модулей из программного списка производится аналогичным образом но для этого курсор должен находиться в окне **Program List**, в которое он также перемещается при нажатии клавиши TAB. После этого необходимо сохранить программный список

(одновременным нажатием клавиш ALT и S - Save List). После этого на экране опять появится окно редактора.

Обратите внимание, что в нижней строке экрана в графе **Program List** теперь находится название GRDEMO.

Вот сейчас нажатие клавиши F5 должно увенчаться успехом. В результате всех наших манипуляций на экране появится небольшое меню, принадлежащее самой программе GRDEMO. Выбирая различные опции, вы сможете ознакомиться с основными возможностями библиотеки графических функций Microsoft.

Существует и другая возможность счастливо избежать вышеприведенных ошибок. Для этого перед загрузкой *QuickC* выполните одну операцию - создайте библиотеку типа *Quick library* из программы GRDEMO. Такая библиотека создается при исполнении команды

```
QLIB /s grdemo.c
```

После этого вызов *QuickC* необходимо осуществить следуюшим образом:

```
QC /l grdemo.qlb grdemo
```

и уже затем просто нажать F5.

Библиотеки типа *Quick library* удобны при работе с *QuickC*, установленным на дискеты. В этом случае, при загрузке *QuickC* с опцией */l*, *Quick library* загружается в оперативную память, тем самым существенно экономя время, затрачиваемое на обращение к дисководу, поиск необходимых файлов и так далее. При работе с *QuickC*,

установленным на твердый диск, лучше использовать программные списки, экономя оперативную память.

Покинем на некоторое время *QuickC*, для чего нажмем одновременно клавиши ALT и F, а затем клавишу X.

Если теперь посмотреть на содержимое каталога C:\MS5\BIN\SAMPLE, то обнаружится, что к имевшимся там до наших упражнений файлам добавились еще четыре - GRDEMO.MAK, GRDEMO.OBJ, GRDEMO.EXE и GRDEMO.MAP.

Исполняемый модуль GRDEMO.EXE, только что демонстрировавший нам разнообразные графические функции Microsoft, на поверку оказывается не таким уж исполнимым - при попытке его запуска с командной строки ДОС на экране появится уже знакомое нам сообщение

```
Program too big to fit in memory
```

Увы, этот файл работоспособен лишь в среде *QuickC*, из которой мы недавно вышли.

Как же нам получить программу, не требующую для своей работы этой среды?

Попробуем сделать это с командной строки с помощью драйверов CL.EXE и QCL.EXE. В этом случае после исполнения одной из команд

```
CL GRDEMO.C
```

или

```
QCL GRDEMO.C
```

вам нужно будет немного подождать, пока не закончатся процессы компиляции и компоновки, отображаемые на экране соответствующими надписями, а затем запустить программу GRDEMO. И мы опять сможем наблюдать круги и загогулины, скачущие по экрану.

Можно также использовать утилиту MAKE, позволяющую с помощью программного списка (здесь - GRDEMO.MAK) отслеживать только самые последние изменения в модулях программы и производить соответствующие модификации EXE-файла.

Чтобы посмотреть на эту утилиту в действии, сотрите файлы GRDEMO.OBJ и GRDEMO.EXE (иначе ничего не произойдет) и введите с командной строки

```
MAKE GRDEMO.MAK
```

Утилита MAKE работает в буквальном смысле как часы (главный критерий модификации для нее - дата и время сохранения файла). Увы, результатом и ее усилий будут все те же круги и спирали.

Теперь, когда мы убедились в работоспособности *Microsoft C 5.0* и ознакомились с работой программы GRDEMO, рассмотрим на ее же примере процесс отладки программ, осуществляемый с помощью *QuickC*.

Вернемся снова в *QuickC* с помощью команды

```
QC GRDEMO
```

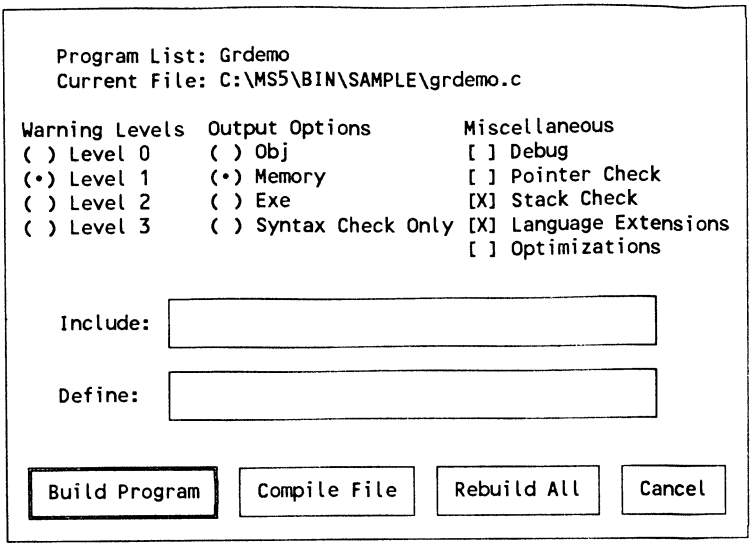
Поскольку программный список GRDEMO.MAK уже существует, *QuickC* спросит нас, хотим ли мы сразу установить его. Ответив *Y*, мы опять увидим на экране окно редактора со знакомым текстом программы GRDEMO.

До начала знакомства с отладчиком попробуем еще раз нажать клавишу F5. *QuickC*, недолго подумав, сообщит нам, что GRDEMO.EXE существует, однако после нажатия клавиши ENTER неожиданно выдаст сообщение об ошибке. Она заключается в том, что *QuickC* не в состоянии загрузить файл GRDEMO.EXE, способный исполняться с командной строки.

Итак, мы наблюдаем полное "взаимное отторжение" командной строки ДОС и среды *QuickC*.

Ничего страшного - нам не составит труда перекомпилировать программу заново, тем более что нам просто необходимо это сделать для работы с отладчиком.

Вызовем меню **Run** (ALT+R) и выберем опцию **Compile**. Перед нами появится очередное окно со множеством вложенных окошечек, круглых и квадратных скобок со значками внутри и столбцы слов рядом со скобками. Оно изображено на приводимом ниже рисунке:



Левый столбец скобок - **Warning Levels** (уровни предупреждений) - позволяет выбрать требуемый уровень строгости предупреждений о возможных несовершенствах компилируемой программы. Мы не будем в этот раз обращать на него внимания.

Следующий столбец - **Output Options** (опции вывода) - содержит указания на то, в каком виде будет представлен результат работы *QuickC*:

- выбор опции **Obj** или **Exe** приведет к образованию OBJ- или EXE-файла соответственно;

- выбор опции **Memory** позволит создать исполняемый модуль в знакомом по предыдущему

разделу EXE-виде, работающем только в среде *QuickC*;

- выбор опции **Syntax Check Only** дает *QuickC* указание ограничиться только проверкой синтаксических ошибок в тексте программы.

Если сейчас мы установим точку в круглых скобках против опции **Exe**, мы сможем получить файл GRDEMO.EXE, приемлемый для работы в ДОС. Впрочем, именно такой файл у нас и имеется в настоящий момент. Значит, и в этом столбце мы не станем ничего изменять, оставив указующую точку в скобках против опции **Memory**.

Последний столбец - **Miscellaneous** (который можно перевести как "разное") - представляет на выбор следующие опции.

Debug (отладка) - это как раз то, что нам нужно. Выбор этой опции (с помощью клавиш ALT+D) указывает *QuickC* на необходимость подготовки программы для отладки.

Следующие две опции - **Pointer Check** (проверка указателя) и **Stack Check** (проверка стека) - требуют от *QuickC* выполнения проверок правильного использования указателей и наличия места в стеке программы, достаточного, чтобы разместить в нем локальные переменные вызываемой функции.

Опция **Language Extensions** (расширения языка) позволяет использовать языковые элементы *Microsoft C*, отличные от стандарта ANSI (такие, например, как *fortran* или *huge*).

Последняя опция этого столбца - **Optimizations** (оптимизации) - удлиняет процесс компиляции, позволяя взамен получить более быструю программу. Однако иногда случается, что оптимизированная подобным

образом программа (состоящая, например, из нескольких вложенных циклов) приносит весьма странные результаты. Я бы не рекомендовал пользоваться этой опцией без особой необходимости.

Окно **Include** служит для указания дополнительных каталогов для поиска в них header-файлов. Окно **Define** может использоваться для дополнительного определения макросов, если в том возникнет нужда.

Итак, мы выберем опцию **Debug** и нажмем клавиши ALT+R, что соответствует операции **Rebuild All** (перекомпилировать заново все файлы, входящие в программный список), приведенной в нижней части окна **Compile**. Опять побегут номера компилируемых строк, пройдет стадия **Linking** (компоновка), и *QuickC* вернет нас к окну редактора. Обратите внимание, что после выполнения **Rebuild All** программа не исполняется, а ожидает наших дополнительных указаний.

Теперь можно начинать пошаговое выполнение программы GRDEMO. Нажмем клавишу F8. Если вы работаете на компьютере с цветным монитором, вы увидите, что текст GRDEMO.C в окне редактора переместится вверх и буквы в строке, содержащей функцию *main()*, позеленеют. Этим цветом (по умолчанию) в *QuickC* отмечается текущая строка исполняемой в отладчике программы.

Шаг за шагом, нажимая F8, мы с вами можем отслеживать все действия, которые производит программа GRDEMO. Впрочем, проходить (и неоднократно) все немыслимое количество строк в GRDEMO.C вовсе необязательно. Мы можем установить курсор в любой строке и заставить программу быстро дойти до этого места, нажав клавишу F7. Предположим, что мы выберем для этого одну из строк в начале, а именно:

```
switch (vc.adapter) {
```

После нажатия клавиши F7 буквы в этой строке, как и положено, станут зелеными.

Теперь давайте угадаем, куда дальше отправится наша программа, то есть какой именно вариант из предложенных она выберет.

Для этого мы попробуем определить значение переменной *vc.adapter*. Эта операция наиболее быстро выполняется в несколько приемов:

1. Пользуясь комбинацией клавиши CTRL и клавиш горизонтального движения курсора, установите курсор под буквой *v*.

2. Держа нажатыми одновременно клавиши CTRL и SHIFT, дважды нажмите клавишу движения курсора вправо. После этого *vc.adapter* выделится инверсным изображением (попутно вы уже научились выделять блок текста, который можно запомнить комбинацией клавиш CTRL+INS, а затем вывести в любом месте экрана другой комбинацией клавиш - SHIFT+INS).

3. Комбинацией клавиш ALT+D вызовите меню **Debug**. Его первая опция - **Add Watch** (добавить контролируемое выражение) - выделена красным цветом фона (активна).

4. Нажмите клавишу ENTER и посмотрите на вторую сверху строку экрана. Там вы увидите название искомой переменной *vc.adapter* и ее значение.

Удобством отладчика *QuickC* является возможность вывести в одной строке целые группы данных, например строку, массив или структуру. Поскольку переменная *vc*

представляет собой структуру, посмотрим на нее не отходя от оператора *switch*.

Для этого снова переместим курсор к букве v и снова вызовем меню **Debug**. Опять нажмем клавишу ENTER. На сей раз перед нами появится окно, содержащее название этой переменной. Еще раз нажмем клавишу ENTER. Окно редактора снова уменьшится, и под строкой с *vc.adapter* появится строка, содержащая значение переменной *vc*. Числа, разделенные запятой, соответствуют значениям элементов структуры videoconfig, описанной в header-файле `<graph.h>` (мы еще коснемся более подробно этой структуры в одном из выпусков серии *"C для PC"*, посвященных основам графического программирования).

Заглянем в *graph.h* и посмотрим на эту структуру. Для этого снова изберем самый быстрый способ:

1. Установите курсор в любую позицию строки, содержащей директиву

   ```
   #include <graph.h>
   ```

   Для этого нажмите клавиши CTRL+HOME (в начале текста) и перейдите к искомой строке, уже находящейся в пределах окна редактора.

2. Комбинацией клавиш ALT+V вызовите меню **View**. Нажмите клавишу I (соответствующую опции **Include**).

3. В тексте появившегося перед вами файла *graph.h* найдите структуру *videoconfig* (почти в самом начале).

Как и следовало ожидать, элемент *adapter* (третий с конца в структуре) имеет одно и то же значение

независимо от способа ознакомления с ним. Полистав немного файл *graph.h*, мы вскоре набредем на таблицу макросов адаптеров:

```
/* videoconfig adapter values */
/* these manifest constants can be used to test  */
/* adapter values for a particular adapter using */
/* the bitwise-AND operator (&)                  */

#define _MDPA 0x0001 /* Monochrome Display Adapter */
#define _CGA  0x0002 /* Color Graphics Adapter     */
#define _EGA  0x0004 /* Enhanced Graphics Adapter  */
#define _MCGA 0x0008 /* MultiColor Graphics Array  */
#define _VGA  0x0010 /* Video Graphics Array       */
```

Определив из этой таблицы по значению переменной *vc.adapter* тип вашего адаптера, вы сможете легко предугадать следующий шаг программы GRDEMO, остановившейся на операторе switch.

Вернемся к файлу GRDEMO.C. Для этого необходимо просто нажать клавишу **F2** (эквивалентную выбору опции **Open Last File** в меню **File**). Нажав клавишу F8, мы можем убедиться, что программа GRDEMO отправится именно к тому адаптеру, который находится внутри вашего компьютера.

Теперь рассмотрим еще один способ "форсированного" исполнения программы. Для этого нажав пару раз клавишу PgDn отыщите в тексте GRDEMO.C следующую строку:

```
setturtle(vmode);
```

Установите курсор на этой строке и нажмите клавишу F9, эквивалентную опции **Toggle Breakpoint** (установить/удалить точку приостановки выполнения программы в отладчике в меню **Debug**). Эта строка "покраснеет".

Если вам не нравятся цвета отладчика *QuickC*, вы можете сами установить их по вашему желанию. Для этого когда-нибудь на досуге вызовите меню **View** и выберите опцию **Options**. Я думаю, что для вас не составит труда разобраться во всем этом буйстве красок самостоятельно.

Пока же продолжим знакомство с отладчиком. Установив **Breakpoint**, мы можем нажать клавишу F5 и подождать, пока GRDEMO остановится на отмеченной строке. Для этого нам придется еще разок взглянуть на меню программы GRDEMO. Выбрав из него, скажем, опцию **Circles** и нажав любую клавишу, мы вернемся к этой строке программы. Отметка **Breakpoint** при этом не снимается, и, если мы еще раз нажмем клавишу F5, а затем выберем из меню GRDEMO любую опцию, кроме **Quit**, программа вновь вернется на старое место. Такой возможностью удобно пользоваться при отладке программ, в которых есть большие циклы. Кроме того, можно устанавливать **Breakpoint** сразу в нескольких местах, отслеживая соответствующие изменения Watch-переменных. Чтобы не забыть при необходимости снять все отметки **Breakpoint**, в меню **Debug** существует опция **Clear All Breakpoints**.

А как быть в ситуации, когда программа, откомпилированная с опцией **Debug**, запущена клавишей F5 без установки **Breakpoint**, а вам необходимо остановить ее, чтобы продолжить отладку? Воспользуйтесь стандартной комбинацией CTRL+BREAK и вы вернетесь в редактор *QuickC* к строке программы,

соответствующей ее состоянию в момент прерывания исполнения.

Давайте попробуем так сделать, убрав **Breakpoint**. Если вы нажмете клавиши CTRL+BREAK в тот момент, когда на экране изображено меню GRDEMO, то окажетесь в редакторе на строке, содержащей оператор

```
continue;
```

Чтобы сориентироваться в тексте программы, вызовем меню **Calls**, нажав клавиши ALT+C. Это меню, в отличие от всех предшествующих, не содержит никаких команд. Список его опций - это названия вложенных функций, которые необходимо вызвать, чтобы попасть в данную точку программы. В нашем случае таких функций две - *main()* и *menu()*. Они расположены (сверху вниз) в порядке, обратном последовательности их вызова (вложенности).

Итак, мы находимся внутри функции *menu()*. Посмотрим, как мы попали сюда, выбрав в меню **Calls** опцию *main()*.

Замечательно: мы опять оказались на той строке, на которой в свое время устанавливали **Breakpoint**. В этом нет ничего шарлатанского: - *QuickC* при подобной операции возвращает нас в строку, соответствующую точке завершения вызванной функции - *menu()*.

Все - теперь вы знакомы с основными функциями и возможностями отладчика *QuickC*. Надеюсь, что это знакомство не было слишком утомительным. Во всяком случае оно поможет вам сберечь драгоценное время при дальнейшем создании программ с помощью *QuickC*.

И в заключение- еще несколько мелких полезных советов на эту же тему:

1. Если ваш программный список состоит из нескольких файлов, для переходов между ними пользуйтесь опцией **Source** из меню **View**. Это гораздо быстрее, нежели использование опции **Open** из меню **File**.

2. При использовании длинных и невразумительных названий функций и констант используйте описанные здесь "прыжки" в header-файлы, содержащие эти названия и запоминание/восстановление выделенных фрагментов текста.

3. Для копирования строк используйте комбинацию клавиш CTRL+Y (удаление строки с запоминанием) с последующим восстановлением ее с помощью комбинации клавиш SHIFT+INS.

4. Неверные исправления в пределах одной строки, пока курсор все еще находится на ней, восстанавливаются командой **Undo** из меню **Edit**

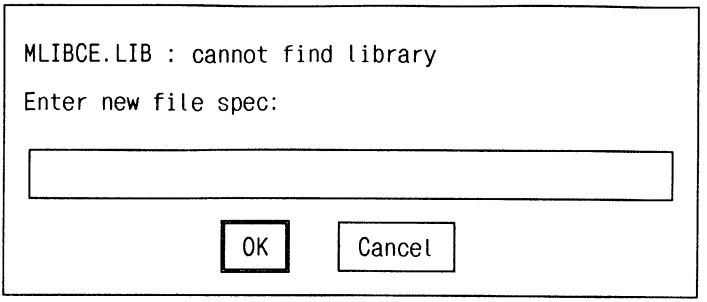
5. В этом же меню есть полезная опция **Read Only**, защищающая отлаживаемую программу от случайных исправлений, возникающих при лишних нажатиях клавиш.

6. Для перелистывания списка ошибок в нижнем окне *QuickC* пользуйтесь комбинациями SHIFT+F3 (следующая ошибка) и SHIFT+F4 (предыдущая ошибка).

7. Для уточнения синтаксиса прототипа некоторой библиотечной функции или оператора *C* установите курсор в позицию в пределах названия этой функции (оператора) в тексте вашей программы и нажмите клавиши SHIFT+F1. В верхней части экрана появится окно, содержащее необходимую краткую информацию.

НЕКОТОРЫЕ ТИПИЧНЫЕ ОШИБКИ ИЗ-ЗА НЕТОЧНОЙ УСТАНОВКИ Microsoft C и QuickC

■ После компиляции GRDEMO.C в *QuickC* перед вами возникает окно, содержащее сообщение о невозможности найти библиотеку MLIBCE.LIB:



Приводит к образованию файла LINK.ERR следующего содержания:

```
Microsoft (R) Overlay Linker Version 3.65
Copyright (C) Microsoft Corp 1983-1988.
All rights reserved.

LINK : fatal error L1023: terminated by user
```

Причина	Способ исправления
Medium model не установлена	Воспользуйтесь программой SETUP с опцией *M*
Не установлена переменная LIB окружения ДОС	Установите переменные окружения по образцу, приведенному в главе "Окружение"
Выход за пределы окружения ДОС	Вставьте в файл CONFIG.SYS строку `SHELL=COMMAND.COM /E:1000 /P`

■ При компиляции GRDEMO.C в нижней части экрана появляется окно, содержащее информацию об ошибке:

```
fatal error C1015: (1 of 1)
cannot open include file 'graph.h'
```

Причина	Способ исправления
Скорее всего произошел выход за пределы окружения ДОС	Вставьте в файл CONFIG.SYS строку `SHELL=COMMAND.COM /E:1000 /P`

■ После компиляции GRDEMO.C в момент начала стадии **Linking** в центре экрана на очень непродолжительное время появляется надпись, которую совершенно невозможно прочесть. После этого образуется пустой файл LINK.ERR.

Причина	Способ исправления
Вызов LINK.EXE, входящего в состав ДОС	Поменяйте в команде PATH в файле AUTOEXEC.BAT местами C:\DOS; и C:\MS5\BIN;
APPEND (ДОС)	Исключите из списка аргументов APPEND каталог C:\MS5\BIN

ПРИЛОЖЕНИЕ А

В этом приложении описана процедура установки *QuickC* на твердый диск либо на систему с гибкими дисками. Недостающая информация приведена в основном тексте выпуска.

Запуск программы SETUP на системе с жестким диском

Как только вы будете готовы установить вашу систему на жесткий диск, вставьте рабочую копию дистрибутивного диска Libraries#1 в дисковод A. Затем введите следующую командную строку в форме:

```
SETUP H C:\dest {S|M|L} {EM|87} [GR]
[\bin][\incl][\lib]
```

Далее описаны аргументы, которые входят в командную строку программы SETUP для *QuickC*

Аргумент	Значение
H	Установка *QuickC* на жестком диске.
C:\dest	Введите *C*, чтобы установить систему на дисковод C, за ним непосредственно следует имя целевого каталога (*dest*) для установки программного обеспечения. SETUP

	предполагает,что все остальные имена, заданные в командной строке, - это имена подкаталогов в данном каталоге.
S, M, C, L	Одна или более из данных букв, разделенных пробелами, сообщает программе SETUP, какая модель памяти будет использоваться. Введите *M*, поскольку для *QuickC* средняя модель памяти является стандартной. Если вы будете разрабатывать программы вне программной среды *QuickC*, например с помощью драйвера QCL.EXE, наберите *S*, поскольку малая модель памяти стандартна для команды QCL.
EM,87	Введите любой из двух или оба данных аргумента, чтобы сообщить программе SETUP, каким образом будут обрабатываться в ваших программах операции с плавающей точкой. Чтобы сообщить пакету *QuickC*, что в первую очередь будут использоваться процедуры эмуляции операций с плавающей точкой, (а не сопроцессор 8087 или 80287), введите опцию *EM*. Если вы разрабатываете программы, использующие научные, математические или финансовые вычисления с вещественными числами и всегда будете работать с сопроцессором, введите *87*. В этом

случае будет построена дополнительная библиотека, которая потребует дополнительной дисковой памяти, но разработка и выполнение программ с данной библиотекой будет производиться гораздо быстрее.

GR — Если вы будете использовать графические функции пакета *QuickC*, введите параметр *GR*. Эти функции будут добавлены в библиотеку, которую строит программа SETUP. Данная опция делает библиотеку приблизительно на 50K больше. Если данная опция не будет задана, графические функции не будут включены в библиотеку, и, когда ваша программа будет использовать графические функции, компилятор *QuickC* должен будет найти библиотеку GRAPHICS.LIB.

\bin — Введите имя подкаталога каталога *dest*, в который будут сброшены выполняемые файлы компилятора, в частности компилятор, компоновщик и утилиты. Если вы опускаете этот аргумент или введете вместо него знак вопроса (?), программа SETUP по умолчанию использует подкаталог \BIN.

\incl — Введите имя подкаталога каталога *dest*, в котором будут установлены header-файлы. Если вы

	опускаете данный аргумент или введете вместо него знак вопроса (?), программа SETUP использует по умолчанию подкаталог \INCLUDE.
\lib	Введите имя подкаталога каталога *dest*, в котором вы хотите установить библиотечные файлы. Если вы опустите данный аргумент или введете вместо него знак вопроса (?), программа SETUP использует по умолчанию подкаталог LIB.

Запуск программы SETUP на системе с гибкими дисками

Перед тем как запустить программу SETUP на системе с гибкими дисками, вам следует определить, сколько вам понадобится чистых отформатированных дискет для того, чтобы разместить все библиотеки, которые будут построены в процессе установки системы. Вам потребуется по одной чистой отформатированной дискете для каждой модели памяти, которую вы будете использовать, плюс одна рабочая дискета; если вы будете пользоваться только одной стандартной для программной среды *QuickC* моделью памяти (средняя модель - *medium model*), вам понадобятся всего две дискеты. Программа SETUP разместит каждую построенную библиотеку на отдельном гибком диске. После того как вы отформатируете требуемое количество дискет, вставьте в дисковод A вашу рабочую копию дистрибутивного диска Libraries#1. Затем введите командную строку в следующей форме:

```
SETUP F B: {S|M|C|L} {EM|87} [GR]
```

Аргумент	Значение
F	Установка *QuickC* на дискеты.
B:	Введите *B:*, чтобы установить библиотеки на дисковод B.

Прочие аргументы описаны выше.

Установка переменных окружения на системе с гибкими дисками

Эти переменные следует определить в файле AUTOEXEC.BAT следующим образом:

```
SET PATH=A:;
SET LIB=A:;
SET INCLUDE=A:\INCLUDE;
```

Данные команды дают *QuickC* указание искать исполняемые и библиотечные файлы в корневом каталоге дискеты в дисководе A, а header-файлы - в каталоге \INCLUDE на дискете в дисководе A.

ПРИЛОЖЕНИЕ В

"Ближняя" и "дальняя" адресация

Термины "near" и "far" (ближний и дальний) обозначают, каким образом осуществляется доступ к данным в сегментной архитектуре семейства микропроцессоров 80n86.

Операционная система *DOS* загружает код и данные, распределенные вашей программой в сегменты (физическую память). Каждый сегмент имеет длину 64K. Поскольку для программного кода и данных всегда выделяются разные сегменты, минимальное число сегментов, выделенных для программы, это два. Поскольку эти два сегмента требуются каждой программе, они называются стандартными (используются по умолчанию).

Малая модель памяти использует только два этих сегмента. Остальные модели памяти, описанные в данном приложении, используют либо более одного сегмента кода, либо более одного сегмента данных, либо и то, и другое.

В семействе микропроцессоров 80n86 адресация памяти состоит из двух частей:

1. 16-битовое число, представляющее собой базовый адрес сегмента памяти.
2. 16-битовое число, которое задает смещение в пределах данного сегмента.

Архитектура микропроцессора 8086 такова, что доступ к данным в стандартном сегменте кода или данных

осуществляется посредством 16-битового значения смещения. Это возможно, поскольку адреса стандартных сегментов всегда известны. Данное 16-разрядное смещение и называется "ближним" адресом; к нему можно обратиться с помощью near-указателя. Так как для доступа к "ближней" памяти требуется только 16-разрядная арифметика, near-ссылки на код или данные всегда более эффективны.

Когда данные или код лежат вне стандартных сегментов, адрес должен использовать как значение сегмента, так и значение смещения. Такие адреса называются "дальними" адресами; в *C*-программе к таким адресам обращаться можно с помощью far-указателей. Доступ к данным, которые можно назвать "дальними", стоит дороже (в смысле размера и скорости программ), зато это средство позволяет вам адресовать всю память, а не только 64 килобайта.

Использование стандартных моделей памяти QuickC

Библиотеки, построенные с помощью программы SETUP, поддерживают четыре стандартные модели памяти. Использование стандартных моделей памяти является самым простым способом управлять доступом к данным вашей программы.

Когда вы используете стандартные модели памяти, компилятор поддерживает библиотеки, соответствующие выбранной вами модели памяти. Каждая модель памяти имеет свою собственную библиотеку.

Основным преимуществом использования стандартных моделей памяти для ваших программ является простота. В стандартных моделях памяти управление памятью задается посредством опций компилятора. Поскольку

стандартные модели не требуют использования расширенных ключевых слов, они являются лучшим средством написания программ, переносимых на другие системы (не применяющие сегментную архитектуру).

Очень важно помнить о двух общих особенностях всех четырех моделей:

1. Ни один исходный модуль не может сгенерировать 64K или больше кода.
2. Ни один элемент данных не может превышать 64K.

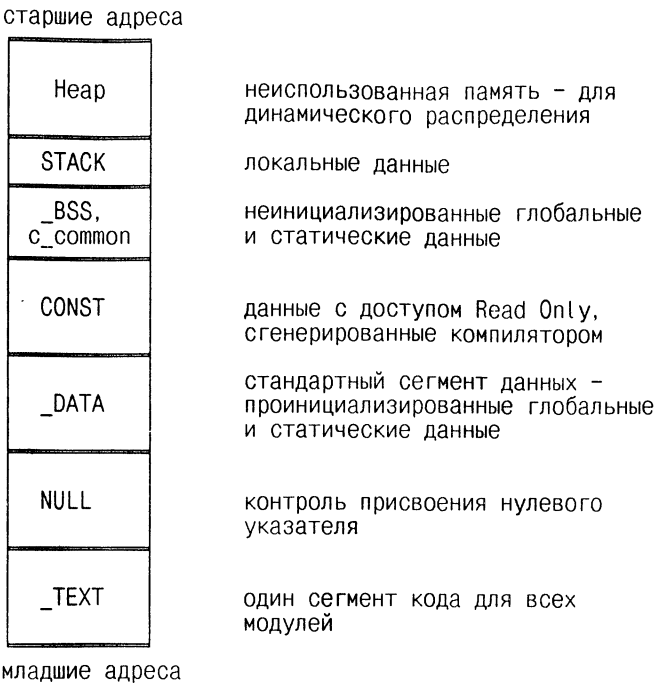
Создание программ для малой модели памяти (small model)

Опция */AS*

Опция малой модели памяти требует, чтобы компилятор создал программу, занимающую два стандартных сегмента - один для кода и один для данных. Программы малой модели памяти обычно небольшие по размеру либо имеют ограниченные цели. Поскольку размеры кода или данных для программ ограничены 64K, общий размер программы для малой модели памяти не может превышать 128K. Большинство программ легко умещаются в данную модель.

По умолчанию как к коду, так и к данным программ малой модели памяти доступ осуществляется через "ближние" адреса. Вы можете отменить такой подход посредством использования ключевого слова "far". Если вы явно не определите модель памяти, команда CL/QCL автоматически создает программы для малой модели памяти. Опция */AS* обеспечивается автоматически, вам нет необходимости задавать ее явно.

На рисунке показано, как распределяется память в малой модели.

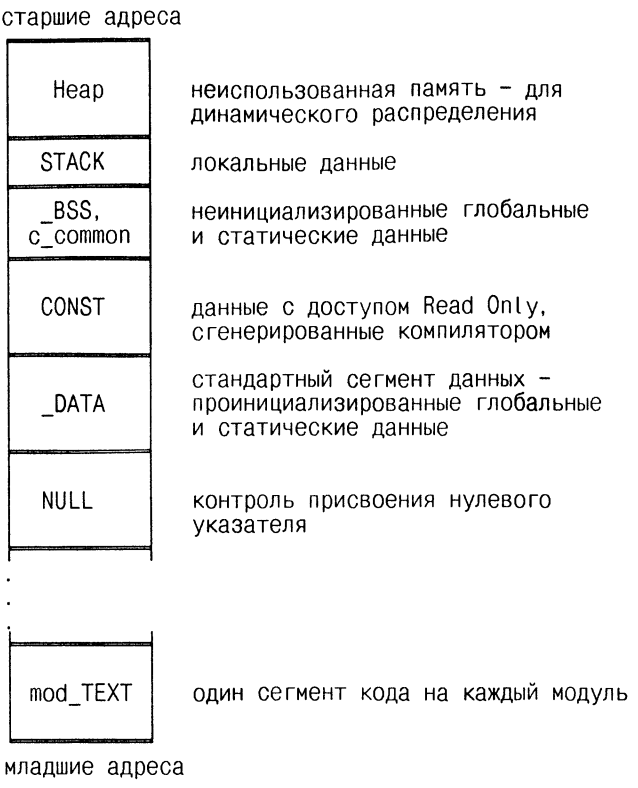


Создание программ для средней модели памяти (medium model)

Опция */AM*

Опция средней модели памяти распределяет один сегмент для данных и несколько сегментов для программного кода. Каждому исходному модулю отводится свой собственный сегмент кода.

На рисунке показано, как распределяется память для средней модели.



C-программы средней модели памяти обычно имеют большое количество операторов (занимающих больше 64K), но относительно небольшое количество данных (менее 64K). Программный код может занимать большое количество памяти - столько программных сегментов, сколько требуется, в то время как общее количество данных не может быть более 64K. В программах средней

модели памяти обеспечивается хорошее сочетание между скоростью работы и размером программ, поскольку большинство программ чаще обращаются к данным, а не к коду.

Поскольку программы, скомпилированные в среде *QuickC*, всегда используют среднюю модель памяти, данную опцию вам следует задавать в том случае, если вы применяете команду CL/QCL, чтобы скомпилировать модуль для использования в среде *QuickC*.

Создание программ для компактной модели памяти (compact model)

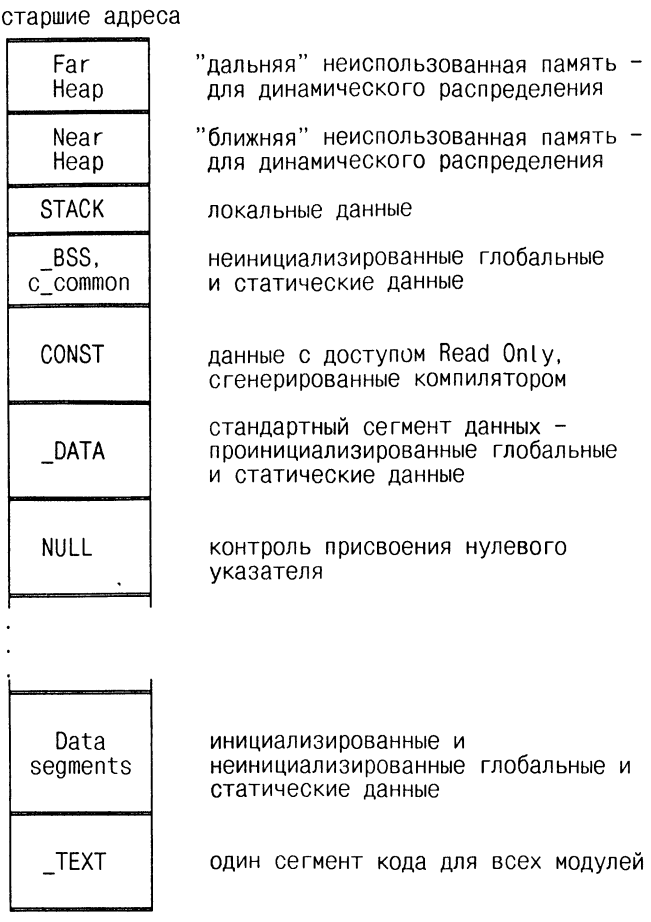
Опция */AC*

При задании компактной модели памяти компилятор создает несколько сегментов для данных, но всего один сегмент для программного кода.

Программы компактной модели памяти обычно имеют бо́льшое количество данных, но относительно небольшое число программных операторов. Данные могут занимать любое количество памяти и столько сегментов, сколько требуется.

По умолчанию доступ к элементам кода в программах компактной модели осуществляется посредством "ближних" адресов, а к элементам данных - посредством "дальних" адресов. Вы можете отменить это с помощью ключевого слова "near" для данных и ключевого слова "far" для программного кода.

На рисунке показано, как распределяется память для компактной модели памяти.


старшие адреса
Far Heap
"дальняя" неиспользованная память - для динамического распределения
Near Heap
"ближняя" неиспользованная память - для динамического распределения
STACK
локальные данные
_BSS, c_common
неинициализированные глобальные и статические данные
CONST
данные с доступом Read Only, сгенерированные компилятором
_DATA
стандартный сегмент данных - проинициализированные глобальные и статические данные
NULL
контроль присвоения нулевого указателя
Data segments
инициализированные и неинициализированные глобальные и статические данные
_TEXT
один сегмент кода для всех модулей
младшие адреса

Создание программ для большой модели памяти (large model)

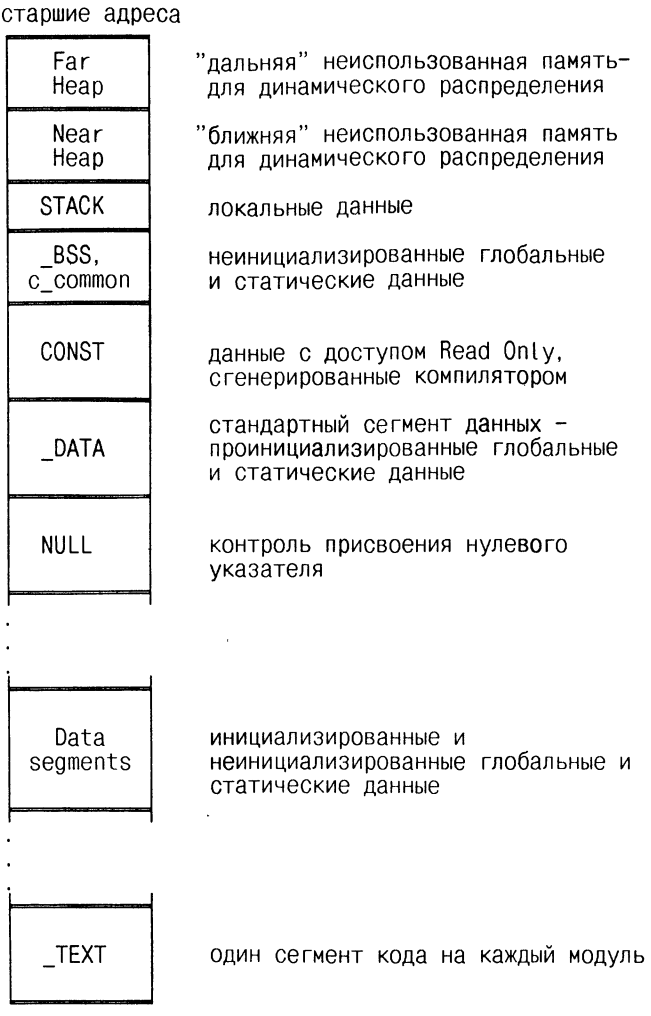
Опция */AL*

При заданной опции большой модели памяти компилятор создает несколько сегментов (столько, сколько требуется) как для кода, так и для данных.

Программы для большой модели памяти обычно имеют очень большой размер программного кода и используют большое количество данных.

По умолчанию в программах большой модели памяти доступ к коду или данным осуществляется посредством "дальних" адресов. Отменить данный стандарт вы можете с помощью использования ключевого слова "near".

На рисунке показано распределение памяти большой модели.


старшие адреса
Far Heap
"дальняя" неиспользованная память- для динамического распределения
Near Heap
"ближняя" неиспользованная память для динамического распределения
STACK
локальные данные
_BSS, c_common
неинициализированные глобальные и статические данные
CONST
данные с доступом Read Only, сгенерированные компилятором
_DATA
стандартный сегмент данных - проинициализированные глобальные и статические данные
NULL
контроль присвоения нулевого указателя
Data segments
инициализированные и неинициализированные глобальные и статические данные
_TEXT
один сегмент кода на каждый модуль
младшие адреса

ПРИЛОЖЕНИЕ С

Справочник сообщений об ошибках

В выпусках серии *"С для PC"* мы будем приводить информацию об ошибках, выдаваемых в процессе работы программ *Microsoft C 5.0*. Ввиду большого объема справочника сообщений об ошибках в каждом выпуске будет описана одна из групп таких сообщений.

Сообщения о фатальных ошибках

Сообщение о фатальной ошибке указывает на наличие некоторой проблемы, которая запрещает компилятору продолжить выполнение его работы. Данный тип сообщения имеет следующий формат:

```
имя файла(строка): fatal error C1xxx: сообщение
```

После того как компилятор выдаст сообщение о фатальной ошибке, он завершит выполнение без создания объектного файла и какой-либо проверки на последующие ошибки.

Следующие сообщения идентифицируют фатальные ошибки. Компилятор не может исправить фатальную ошибку; он прекращает работу после вывода сообщения об ошибке.

Номер	Сообщение о фатальной ошибке
C1000	*"Неизвестная фатальная ошибка. Свяжитесь с Техническим сервисом фирмы Microsoft".* Компилятор обнаружил неизвестную ошибку. Пожалуйста, сообщите фирме Microsoft условия возникновения данной ошибки посредством специальной формы "Product Assistance Request".
C1001	*"Внутренняя ошибка компилятора. Свяжитесь с Техническим сервисом фирмы Microsoft".* Компилятор обнаружил внутреннее несоответствие. Пожалуйста, сообщите условия возникновения данной ошибки с помощью бланка "Product Assistance Request".
C1002	*"Выход за пределы динамической области".* Компилятор вышел за пределы динамической области памяти. Данная ситуация обычно означает, что ваша программа имеет слишком много символических имен и/или комплексных выражений. Чтобы разрешить данную проблему, разделите файл на несколько меньших исходных файлов либо разбейте выражения на меньшие подвыражения.
C1003	*"Счетчик ошибок превысил n; компиляция остановлена".* Ошибок в программе слишком много либо они слишком серьезны, чтобы возможно было продолжение; компилятор должен прервать выполнение.

C1004 *"Неожиданный конец файла (EOF)".* Данное сообщение появляется, если у вас не достаточно памяти на стандартном дисковом устройстве, чтобы компилятор создал требуемые временные файлы. Требуемое пространство примерно в два раза больше размера исходного файла. Данное сообщение может быть также сгенерировано, если комментарий не имеет закрывающего ограничителя (*/) либо если директиве #if не хватает закрывающей директивы #endif.

C1005 *"Строка слишком велика для буфера".* Строка в промежуточном файле компилятора переполняет буфер.

C1006 *"Ошибка записи в промежуточный файл компилятора".* Компилятор не может создать промежуточные файлы, используемые в процессе компиляции. К данной ошибке обычно приводят следующие ситуации:

1. Слишком мало файлов в строке "files = number" файла CONFIG.SYS (компилятор требует чтобы число number было не меньше 15).
2. Не хватает памяти на устройстве, содержащем промежуточные файлы компилятора.

C1007 *"Нераспознанный флаг 'string' в 'option'";* string в опции командной строки option не является корректной опцией.

C1009 *"Ограничения компилятора, возможно рекурсивно определенное макроопределение".* Расширение макрокоманды превышает

размеры доступной памяти. Проверьте, не было ли рекурсивно-определенных макрокоманд либо не слишком ли велик расширяемый текст.

C1010 *"Ограничения компилятора: макрорасширение слишком большое"*. Расширение макрокоманды превышает доступную память.

C1012 *"Неверное вложение скобок - пропущенный 'character'"*. Несоответствие скобок в директиве препроцессора; 'character'- здесь либо левая, либо правая скобка.

C1013 *"Невозможно открыть исходный файл 'filename'"*. Данный файл 'filename' либо не существует, либо не может быть открыт, либо не найден. Удостоверьтесь, что параметры установки среды корректны и что для файла задано корректное имя пути.

C1014 *"Слишком много включаемых файлов"*. Вложение директив #include превышает 10-уровневый предел.

C1015 QCL *"Невозможно открыть включаемый файл 'filename'"*. Данный файл либо не существует, либо не может быть открыт, либо не найден. Удостоверьтесь, что переменные окружения среды заданы корректно и что вы определили корректное имя пути для данного файла.

C1016 *"Директиве #if[n]def требуется идентификатор"*. С директивами #ifdef и #ifndef обязательно употребляется идентификатор.

C1017 *"Неверное выражение целой константы".* Выражение в директиве #if должно вычисляться в константу.

C1018 *"Неожиданная директива '#elif'".* Появление директивы #elif разрешено только внутри директив #if, #ifdef или #ifdef.

C1019 *"Неожиданная директива '#else'".* Появление директивы #else возможно только внутри директив #if, #ifdef или #ifndef.

C1020 *"Неожиданная директива '#endif'".* Директива #endif появилась без соответствующих директив #if, #ifdif или #ifndef.

C1021 *"Неверная команда препроцессора 'string'".* Символы, следующие за знаком (#), формируют неверную директиву препроцессора.

C1022 *"Ожидается директива '#endif'".* Директивы #if, #ifdef или #ifndef не заканчиваются директивой #endif.

C1026 *"Переполнение стека. Пожалуйста, упростите вашу программу."* Ваша программа не может быть далее обработана, поскольку память, требуемая для "разбора" программы, переполняет стек компилятора.

Чтобы разрешить данную проблему, упростите вашу программу.

C1027 QCL — *"Ограничения компилятора: вложение структур/объединений"*. Определения структур и объединений вложены более 10 раз.

C1028 QCL — *"Сегмент segment занимает более 64K"*. В данном сегменте размещены более 64 "дальних" данных. Один модуль может

C1027 CL — иметь не более 64K "дальних" данных. Чтобы разрешить данную проблему, либо разбейте данные на отдельные модули, либо сократите общий объем используемых данных, либо откомпилируйте вашу программу с помощью оптимизирующего компилятора *Microsoft C 5.0* (/Gt).

C1032 — *"Невозможно открыть файл, содержащий объектный листинг 'filename'"*. Имеет силу одно из следующих утверждений, касающихся имени файла или имени пути:

1. Данное имя неверно.
2. Файл с данным именем не может быть открыт из-за нехватки памяти.
3. Уже существует файл с данным именем и атрибутом Read Only.

C1033 — *"Невозможно открыть выходной файл на языке ассемблер 'filename'"*. Одно из условий рассмотренных в описании ошибки с кодом C1032, привело к невозможности открыть данный файл.

C1034 *"Невозможно открыть исходный файл 'filename'"*. Одно из условий, рассмотренных в описании ошибки с кодом C1032, привело к невозможности открыть данный файл.

C1035 *"Выражение является слишком сложным. Пожалуйста, упростите"*. Компилятор не смог сгенерировать код для сложного выражения. Чтобы разрешить данную проблему, разбейте выражение на более простые подвыражения и повторите компиляцию.

C1036 *"Невозможно открыть файл, содержащий исходный листинг 'filename'"*. Одно из условий, рассмотренных в описании ошибки с кодом C1032, привело к невозможности открыть данный файл.

C1037 *"Невозможно открыть объектный файл 'filename'"*. Одно из условий, рассмотренных в описании ошибки с кодом C1032, привело к невозможности открыть данный файл.

C1039 *"Невосстанавливаемое переполнение динамической области в третьем проходе компилятора"*. В третьем оптимизирующем проходе компилятор переполнил динамическую область и прекратил работу. Попытайтесь повторить компиляцию с включенной опцией Optimization (в среде программирования *QuickC*), либо с опцией /Od (в командной строке CL/QCL), либо попытайтесь отделить функцию, содержащую строку, вызвавшую ошибку.

C1040 *"Неожиданный EOF в исходном файле 'filename'"*. В процессе создания листинга исходного файла либо исходного/объектного файла компилятор обнаружил неожиданный конец файла. Данная ошибка произошла, вероятно, если исходный файл был отредактирован в процессе компиляции.

C1041 *"Невозможно открыть промежуточный файл компилятора - больше нет файлов"*. Компилятор не может создать промежуточный файл, используемый в процессе компиляции, поскольку больше нет логических номеров файлов. Данная ошибка может быть исправлена путем изменения строки "files = number" в файле CONFIG.SYS, чтобы задать большее число одновременно открытых файлов (рекомендуется назначить число 20).

C1042 *"Невозможно открыть промежуточный файл компилятора - нет такого файла или каталога"*. Компилятор не может создать промежуточные файлы, используемые в процессе компиляции, поскольку в переменной окружения среды TMP задан неправильный путь.

C1043 *"Невозможно открыть промежуточный файл компилятора"*. Компилятор не может создать промежуточные файлы, используемые в процессе компиляции. Точная причина неизвестна.

C1044 *"Нехватка дисковой памяти для промежуточного файла компилятора".* Из-за недостатка памяти компилятор не может создать промежуточный файл, используемый в процессе компиляции. Для исправления данной ситуации освободите место на диске и повторите компиляцию.

C1045 *"Переполнение при операции с плавающей точкой".* Компилятор получил ошибку при присваивании арифметических констант элементам с плавающей точкой, как в следующем примере:

```
float fp_val = 1.0e100;
```

В данном примере константа двойной точности 1.0e100 превышает максимально допустимое значение для данных с плавающей точкой.

C1047 *"Слишком много появлений опции 'string'".* Данная опция упоминается слишком много раз. Строка 'string' содержит опцию, вызвавшую ошибку.

C1048 *"Неизвестная опция 'character' в 'optionstring'".* Символ 'character' является некорректной буквой для опции 'optionstring'.

C1049 *"Неверный числовой аргумент 'string'".* Вместо string ожидался числовой аргумент.

C1050 *"Сегмент кода 'segmentname' слишком большой".* В процессе компиляции сегмент кода вырос за пределы 36 байтов от 64K. В данном случае используется 36-байтовый заполнитель, поскольку сбои на некоторых платах микропроцессоров 80286 могут вызвать непредсказуемое поведение программ, если среди прочих условий размер кодового сегмента находится в пределах 36 байтов от 64K.

C1052 *"Слишком много директив #if/#ifdef's".* В программе превышено максимальное число уровней вложения директив #if/#ifdef.

C1053 *"Размещение данных DGROUP превышает 64K".* В стандартном сегменте данных были размещены более 64K переменных. Для программ компактной, средней и большой модели памяти выполняйте компиляцию с помощью команды CL/QCL, используя опцию /Gt для размещения элементов данных в отдельных сегментах.

C1054 *"Ограничения компилятора: слишком глубокая вложенность инициализаторов".* Были превышены ограничения компилятора на вложенность инициализаторов. Предел - от 10 до 15 уровней, в зависимости от комбинации инициализируемых типов. Чтобы решить данную проблему, для сокращения уровней вложенности упростите тип инициализируемых данных либо после описания присваивайте первоначальное значение в отдельных операторах.

C1056 *"Ограничения компилятора: переполнение в процессе макрорасширения"*. Компилятор переполнил внутренний буфер при расширении макрокоманды.

C1057 *"Неожиданный EOF в макрорасширении; (пропущена ')?)"*. Компилятор обнаружил конец исходного файла в процессе сборки аргументов макровызова. Обычно это является результатом опущенной закрывающей правой скобки ')' в макровызове, как и в следующем примере:

```
#define print(a) printf(string is(,#a))
main()
{
print(the quick brown fox;
}
```

C1059 *"Превышены пределы "ближней" динамической области"*. При размещении элементов данных в "ближней" динамической области (стандартный сегмент данных) компилятор вышел за допустимые пределы.

C1060 *"Превышены пределы "дальней" динамической области"*. При размещении элементов данных в "дальней" динамической области компилятор вышел за допустимые пределы памяти. Обычно данная ошибка происходит во встроенных в память программах по причине того, что таблица имен содержит слишком много имен.

Чтобы исправить данное положение, попробуйте выполнить компиляцию с выключенной опцией Debug либо попытайтесь подключить меньше включаемых файлов. Если такой способ не спасает ситуацию, выполните компиляцию программы посредством команды CL/QCL.

C1061 QCL — *"Ограничения компилятора: слишком глубокое вложение блоков"*. Вложенность блоков в данной программе превышает возможности компилятора. Для исправления данного положения перепишите программу так, чтобы вложенность блоков была меньшей.

C1063 QCL — *"Ограничения компилятора - переполнение стека компилятора"*. Ваша программа слишком сложна, поэтому произошло переполнение стека компилятора. Упростите вашу программу и повторите компиляцию.

ОГЛАВЛЕНИЕ

ПЛАНИРУЕМАЯ ТЕМАТИКА ВЫПУСКОВ СЕРИИ "С для PC"

Выпуск 0. *Обзор компиляторов Microsoft C 5.0 и QuickC 1.0.*

Выпуск 1. *Связь С-программ с операционной системой.*

Выпуск 2. *Время и звук.*

Выпуск 3. *Стандартный ввод-вывод и использование прерываний.*

Выпуск 4. *Графика на экране I.*

Выпуск 5. *Графика на экране II.*

Выпуск 6. *Текст и графика на принтере.*

Выпуск 7. *Клавиатура.*

Выпуск 8. *Плоттер и диджитайзер.*

Выпуск 9. *Мышь.*

Уважаемый читатель, если у Вас появилось желание приобрести выпуски этой серии, Вы можете сэкономить свое время, оформив предварительный заказ.

Для этого пришлите в наш Центр письмо, вложив в него почтовую открытку с заполненным адресом. На обороте открытки напишите название серии, номер выпуска и количество экземпляров.

Наш адрес: 115409 Москва, ул.Москворечье, 31, корп. 2, Центр МИФИ СП Диалог

ОБУЧЕНИЕ ПРОВОДИТСЯ ПО СЛЕДУЮЩИМ КУРСАМ:

MS-DOS

Подготовка пользователей к практической работе на персональных компьютерах IBM PC включает в себя: изучение операционных систем MS-DOS 3.30 и 4.0, сервисных пакетов, наиболее популярных пакетов текстовых процессоров, электронных таблиц, баз данных и других. Вы также сможете познакомиться с новейшими разработками программных продуктов ведущих западных фирм.

Microsoft Word

Word - один из лучших текстовых процессоров на сегодняшний день. Он обеспечивает возможность просмотра макетов страниц и эффективную работу с графикой. Полезен всем, кто занимается обработкой текстовой информации. Эффективно используется в редакционно-издательской деятельности. Предоставляет широкие возможности для коллективной работы над документами, в том числе для пользователей, работающих в сети.

Microsoft Works

Интегрированный пакет, включающий базу данных, электронную таблицу, текстовый процессор и средства коммуникации, обеспечивает гибкое совместное использование всех этих средств. 70% пользователей ПЭВМ в Западной Европе используют его в деловой, управленческой, экономической и других областях деятельности. Освоение пакета не требует специальных знаний языков программирования или операционных систем.

Microsoft Excel

Этот новый пакет (электронная таблица с деловой графикой и базой данных) в 1989 г. вошел в число лучших программных продуктов данного класса. Имеет рекордный объем таблицы. Позволяет быстро и эффективно решать экономические, математические; технологические, логические и многие другие задачи, которые поддаются вычислениям. Имеются большие возможности по графическому представлению информации в виде диаграмм, графиков и др.

Microsoft Windows

Уникальная графическая среда Windows с системой раскрывающихся меню, диалоговыми окнами и пиктограммами создает отличные условия для одновременной работы с графикой и текстом. Работа в этой среде дает возможность уже сегодня осваивать операционную систему будущего - OS/2. Под управлением Windows работает большое число прикладных программ различных фирм.

QuickC QuickBASIC MS-Pascal

Системы программирования фирмы Microsoft объединяют в единой интегрированной среде текстовый редактор, эффективный компилятор и мощный отладчик. Все системы обеспечивают эффективную оптимизацию программ и удобное графическое отображение информации.

Локальная сеть ПК

Если Вы еще не работали с локальными сетями, то Вы откроете для себя новый мир вычислительных сетей, а если работали, то сможете по достоинству оценить преимущества сети G/NET фирмы Gateway с одной из лучших сетевых операционных систем Advanced Netware 286 фирмы NOVELL.

AutoCAD

Профессиональная универсальная система автоматизированного проектирования фирмы Autodesk. Используется в архитектуре и строительстве, электронике, химической и электронной промышленностях, а также во многих других областях современной науки и техники.

Каждому слушателю выдается учебное пособие по изучаемому курсу. Пособия, выпускаемые Центром МИФИ СП Диалог, рецензируются фирмой Microsoft

Занятия проводятся в течение двух недель с отрывом от производства. Оплата по наличному и безналичному расчету. Зачисление на курсы слушателей, направляемых организацией, проводится по гарантийному письму. Иногородним гарантируется проживание. Телефон учебного центра: 324 83 56

По вопросам покупки программных продуктов фирмы Microsoft обращаться в коммерческий отдел Центра по тел.: 324 71 66, факс: 324 30 55

Адрес Центра МИФИ СП ДИАЛОГ:
115409 Москва, ул.Москворечье, 31, корп. 2

СП ДИАЛОГ
предлагает программные продукты корпорации MICROSOFT
(за конвертируемую валюту)

Наименование	Версия	Система
Языки программирования		
Quick C-Compiler	2.00	MS-DOS
Quick Pascal	1.00	MS-DOS
Quick Basic Compiler	4.50	MS-DOS
Basic Compiler	6.00	MS-DOS & OS/2
C Compiler	6.00	MS-DOS & OS/2
Cobol	3.00	MS-DOS & OS/2
Fortran	5.00	MS-DOS & OS/2
Macro Assembler	5.10	MS-DOS & OS/2
Pascal	4.00	MS-DOS & OS/2
Прикладные системы		
Chart	3.00	MS-DOS
Excel	2.10	MS-DOS
Learning MS-DOS	2.00	MS-DOS
Project	4.00	MS-DOS
Windows Draw	1.05	MS-DOS
Word Exchange	1.00	MS-DOS
Multiplan	4.00	MS-DOS & OS/2
Word	5.00	MS-DOS & OS/2
Системное программное обеспечение		
Windows 286	2.11	MS-DOS
Windows 386	2.11	MS-DOS
Windows Toolkit	2.10	MS-DOS
Аппаратура с программным обеспечением		
Mouse/EasyCad2 (Bus)	2.00	MS-DOS & OS/2
Mouse/EasyCad2 (Serial & PS/2)	2.00	MS-DOS & OS/2
Mouse/Paintbrush (Bus)	1.00	MS-DOS & OS/2
Mouse/Paintbrush (Serial & PS/2)	1.00	MS-DOS & OS/2

Кроме того, фирма Microsoft предлагает набор продуктов для операционной системы MACINTOSH.

Научное издание

Григорьев Андрей
Компиляторы Microsoft C 5.0 и QuickC 1.0

Серия "С для PC. Программирование на языке С для персональных компьютеров", выпуск 0

Редактор *Ю.П.Красильников*
Оформление обложки: *О.А.Кузьминова*
Технические редакторы:
О.А.Красильникова, Л.В.Прохорова
Корректор *В.С.Кустов*

ИБ № 41515

Подписано в печать 17.07.90. Т-11854. Формат 60x88/16. Печ. л. 5.
Усл.-печ. л. 4,9. Тираж 30 000 экз. Заказ № 375

Оригинал-макет подготовлен с помощью текстового редактора Microsoft Word 5.0 и отпечатан на лазерном принтере HP LaserJet series II в Центре совместного советско-американского предприятия "Диалог" на базе Московского инженерно-физического института
115409 Москва, ул. Москворечье, 31, корп. 2

Ордена Трудового Красного Знамени издательство "Наука"
Главная редакция физико-математической литературы
117071 Москва В-71, Ленинский проспект, 15

Московская типография № 11 Госкомпечати СССР
113105 Москва, Нагатинская ул., 1

ДЛЯ ЗАМЕТОК

В серии "С для PC" планируется выпустить:

Выпуск 0.	Компиляторы Microsoft C 5.0 и QuickC 1.0.
Выпуск 1.	Связь С-программ с операционной системой.
Выпуск 2.	Время и звук.
Выпуск 3.	Стандартный ввод-вывод и использование прерываний.
Выпуск 4.	Графика на экране I.
Выпуск 5.	Графика на экране II.
Выпуск 6.	Текст и графика на принтере.
Выпуск 7.	Клавиатура.
Выпуск 8.	Плоттер и диджитайзер.
Выпуск 9.	Мышь.

Предварительные заказы на выпуски серии Вы можете направить по адресу: 115409 Москва, ул. Москворечье, дом 31, корп. 2, Центр МИФИ СП "Диалог".