Dev-C++ 3.99

Dev-C++ is a full-featured integrated development environment (IDE), which is able to create Windows or dos-based C/C++ programs using the Mingw compiler system (included with this package), or the Cygwin compiler.

This program is a quality free software. However, if you like it and want to support us for further development, please register it by paying a small fee of $20. Please see About Dev-C++ in this help file.

Dev-C++ has many tools, like a Setup Creator, automatically DLL generation, Project Templates, Icon Library and much more.

Bloodshed Software: http://www.bloodshed.net/

About Dev-C++ and registering

Free Software :

Bloodshed Dev-C++ is free software.
When we speak of free software, we are referring to freedom, not price.
However Dev-C++ is free (and this time we mean price :-)

Paying for Dev-C++ :

This program is a quality free software.
However, if you like it and want to support us for further development, please register it by paying a small fee of $20. Please read on.

Why paying for Dev-C++ when it is said that it is free ?

My goal in creating Bloodshed Software was to provide to the web community and programmers, quality software which was free.

I don't want to make money with this (and I could! Just look at the success of some software), and I don't want, and will never, sell my software.
It will always stay free. However, it seems many (well, some :) people are willing to register the program (or get the CD) as a donation for the work I've put in it. This is probably because Bloodshed is not a company, but rather a service I can offer to people.
I am very happy to get some money each month, because it really helps me in my student life.

So here is the deal: software is still free.
If you like a program, and use it often, then you can register it or order the CD-ROM for a very small fee (compared to the big software company prices).

You are not obliged to pay, and you can use any software when you want, as long as you want, without paying anything.

This is just a contract from a pleased user to a pleased programmer.
Registered users of Bloodshed software will get the complete source code of the Bloodshed program they choose (Exception: complete source code is available for free download for Bloodshed products covered by the GNU
General Public License, such as Bloodshed Dev-C++).
The prices are also very cheap because for example Dev-C++ is only $20.

The fee can be paid in three ways:

1. By credit card on a high secure server at http://www.bloodshed.net/devcpp.html
2. By check (only French francs, sorry). In this case please contact me at webmaster@bloodshed.net
3. By international mail. Please also contact me.

This doesn't applied to all software. For the moment, it only works with Dev-C++ and may

be applied to new software.

System requirements

These are the minimum requirements of Dev-C++:

Microsoft Windows 95, 98, NT 4 or 2000
8 MB RAM with a big swapfile
100 Mhz Intel compatible CPU
30 MB free disk space

These are the recommended requirements of Dev-C++:

Microsoft Windows 98, NT or 2000
24 MB RAM
233 Mhz Intel compatible CPU
45 MB free disk space

FAQ

1. When I compile my dos program and execute it, Dev-C++ minimizes and then restore in a second but nothing appears ?

When creating a console application, be sure to uncheck "Do not create a console" in Project Options (when working with source files only uncheck "Create for win32" in Compiler Options).

2. When executing my dos program, it closes automatically. How I can change this ?

You can use an input function at the end of you source, like the following example :

```
#include <stdlib.h>

int main()
{
   system("PAUSE");
   return 0;
}
```

3. After linking, i get the error "C:\DEV-C++\LIB\\libmingw32.a(main.o) (.text+0x8e): undefined reference to `WinMain@16'

You probably haven't declared any main() function in your program. Otherwise, try recompiling a second time.

4. When I launch Dev-C++ i get the message saying "WININET.DLL not found" ?

If you are missing WININET.DLL on your Windows system, you can download it at:
http://www.rocketdownload.com/supfiles.htm

5. When I compile a file, I get a message saying "could not find <filename> "

Check in Compiler options if the directories settings are correct. With a default setup, you should have :

C:\DEV-C++\Bin\
c:\DEV-C++\Include\
c:\DEV-C++\Include\
c:\DEV-C++\Lib\

6. The EXE files created are huge. What can i do to reduce the size ?

If you want to reduce your exe file size from 330 Ko to 12 Ko for example, go to compiler options. Then click on the Linker page and uncheck "Generate debug information". This will

remove debugging information (if you want to debug, uncheck it). You can also click on Optimization page and check "Best optimization".

7. Under Windows NT, every time i launch Dev-C++ i get the message "Failed to set data for"

The is because you are not in Administrator mode, and Dev-C++ tries to write to the registry. To get rid of the error message, log on as the Administrator, or uncheck the file association options in Environment options, Misc. Sheet.

8. when I try to compile I get: ld: cannot open crt2. o: No such file or directory. What can i do ?

Go to Compiler options, and check if the Lib directory is correctly set to:
C:\Dev-C++\Lib\
(for a default installation).

9. How can i use the OpenGL library and others ?

All the libraries that comes with Mingw reside in the Lib directory. They are all named in the following way: lib*.a
To link a library with your project, just add in Project options, Further option files :
-lopengl32
This is for including the libopengl32.a library. To add any other library, just follow the same syntax:
Type –l (L in lowercase) plus the base name of the library (filename without "lib" and the ".a" extension).

10. When i compile a file that contains references to Windows filename (like <\ mydir\myfile.h>), i get a "unrecognized escape sequence" message ?

The Mingw compiler understands paths in the Unit style (/mydir/myfile.h). Try replacing the \ in the filename by /

11. Is there any GUI library or packages available for Dev-C++ ?

You can download extra packages for Dev-C++ at http://www.bloodshed.net/dev/

Managing a project

I invite you to read the Tutorial for creating projects, which includes step-by-step procedures for creating and managing projects.

You will learn there how to create, add, remove and rename files from your project, as well as modifying the options and resource file. You can also create a Makefile for your project, for use with the GNU Make utility, as well as with the Borland Make tool.

By right-clicking on the Project Manager or by clicking the Project menu, the project and units options will be displayed.

Compiling, running and debugging

If you want to create the executable of your program and test it, this section will help you on how to do this.

Compiling :

When your sources are ready to be compiled into an executable, click on the Execute menu, then on Compile. The compilation will start and if it is successful (no errors in your program), then an executable with the name of your project and .exe extension will be created in your project's directory. If you got errors after compile, the errors will be shown in the Compiler Output, so just double-click on them to go to the right line in your code (this is not available for Windows NT users, instead errors are shown in a dos console).

Running :

If your project has been successfully compiled, you can execute it by clicking on the Execute menu, then on Compile. Dev-C++ will minimize (by default) and your application will be executed.

Debugging :

If your project has been successfully compiled, you can debug it by clicking on the Execute menu, then on Debug. A window will explain you how to work with the GNU Debugger (please also read the GDB help file in Help\Gdb.hlp), and click on Continue. GDB will be opened and your executable file will be passed to it.

You can use the Cygnus Insight visual (GUI) debugger with Dev-C++. For more information, please go to http://www.bloodshed.net/dev/

Compiler options

This section is about to explain you all the options available in Dev-C++ when running the compiler. The Compiler options dialog is available by clicking on the Options menu, then on "Compiler options".
Most of the parameters description have been taken from the GCC help file.

Directories page :

You can add multiple directories to be search for include files during compilation by checking the appropriate check box, and by typing your directory in the edit field (separate pathname with a semicolon "**;**").

Others compiler-specific parameters can be added by checking the check box "Add the followings commands…", and by typing your commands in the edit field (separate commands by spaces). These commands can be found at Mingw32 web site.

You can also change/add the different directories needed by Dev-C++ (separate folders with a semicolon, but only one Bin directory can be specified).

C/C++ compiler page :

When compiling C programs, you can set the following options :

- Support all ANSI C standard programs (compiler parameter : -ansi)**:**

This turns off certain features of GNU C that are incompatible with ANSI C, such as the asm, inline and typeof keywords, and predefined macros such as unix and vax that identify the type of system you are using. It also enables the undesirable and rarely used ANSI trigraph feature, and it disables recognition of C++ style // comments.

The alternate keywords __asm__, __extension__, __inline__ and __typeof__ continue to work despite -ansi. You would not want to use them in an ANSI C program, of course, but it is useful to put them in header files that might be included in compilations done with -ansi. Alternate predefined macros such as __unix__ and __vax__ are also available, with or without -ansi.

The -ansi option does not cause non-ANSI programs to be rejected gratuitously. For that, -pedantic is required in addition to -ansi. See Warning Options.

The macro __STRICT_ANSI__ is predefined when the -ansi option is used. Some header files may notice this macro and refrain from declaring certain functions or defining certain macros that the ANSI standard doesn't call for; this is to avoid interfering with any programs that might use these names for other things.

The functions alloca, abort, exit, and _exit are not builtin functions when -ansi is used.

- Try to support some aspects of the traditional C preprocessor (compiler parameter : -traditional)

All extern declarations take effect globally even if they are written inside of a function definition. This includes implicit declarations of functions.

The newer keywords typeof, inline, signed, const and volatile are not recognized. (You can still use the alternative keywords such as __typeof__, __inline__, and so on.)

Comparisons between pointers and integers are always allowed.
Integer types unsigned short and unsigned char promote to unsigned int.
Out-of-range floating point literals are not an error.

Certain constructs which ANSI regards as a single invalid preprocessing number, such as 0xe-0xd, are treated as expressions instead.

String "constants" are not necessarily constant; they are stored in writable space, and identical looking constants are allocated separately. (This is the same as the effect of –fwritable-strings.)

Ordinarily, GNU C follows ANSI C: automatic variables not declared volatile may be clobbered.

The character escape sequences \x and \a evaluate as the literal characters x and a respectively. Without –traditional, \x is a prefix for the hexadecimal representation of a character, and \a produces a bell.

In C++ programs, assignment to this is permitted with –traditional. (The option –fthis-is-variable also has this effect.)

You may wish to use –fno-builtin as well as –traditional if your program uses names that are normally GNU C builtin functions for other purposes of its own.

You cannot use –traditional if you include any header files that rely on ANSI C features. Some vendors are starting to ship systems with ANSI C header files and you cannot use –traditional on such systems to compile files that include any system headers.

- Show all warnings (compiler parameter : -w)**:**

This will tell the compiler to show warnings all the warnings.

For C++ programs, you can set the following options:

- Turn off all access checking (compiler parameter : -fno-accesschecking) **:**

Turn off all access checking. This switch is mainly useful for working around bugs in the access control code.

- Accept $ in identifiers (compiler parameter : -fdollars-in-identifier) **:**

Accept $ in identifiers. You can also explicitly prohibit use of $ with the option -fno-dollars-in-identifiers. (GNU C allows $ by default on most target systems, but there are a few exceptions.) Traditional C allowed the character $ to form part of identifiers. However, ANSI C and C++ forbid $ in identifiers.

- Use heuristics to compile faster (compiler parameter : -fsave-memoized) **:**

Use heuristics to compile faster. These heuristics are not enabled by default, since they are only effective for certain input files. Other input files compile more slowly.

The first time the compiler must build a call to a member function (or reference to a data member), it must (1) determine whether the class implements member functions of that name; (2) resolve which member function to call (which involves figuring out what sorts of type conversions need to be made); and (3) check the visibility of the member function to the caller. All of this adds up to slower compilation. Normally, the second time a call is made to that member function (or reference to that data member), it must go through the same lengthy process again. This means that code like this:

cout << "This " << p << " has " << n << " legs.\n";

makes six passes through all three steps. By using a software cache, a "hit" significantly reduces this cost. Unfortunately, using the cache introduces another layer of mechanisms which must be implemented, and so incurs its own overhead. -fmemoize-lookups enables the software cache.

Because access privileges (visibility) to members and member functions may differ from one function context to the next, G++ may need to flush the cache. With the -fmemoize-lookups flag, the cache is flushed after every function that is compiled. The -fsave-memoized flag enables the same software cache, but when the compiler determines that the context of the last function compiled would yield the same access privileges of the next function to compile, it preserves the cache. This is most helpful when defining many member functions for the same class: with the exception of member functions which are friends of other classes, each member function has exactly the same access privileges as every other, and the cache need not be flushed.

The code that implements these flags has rotted; you should probably avoid using them.

Code generation/Optimization page :

Code generation :

- Enable exception handling (compiler parameter: -fexceptions) **:**

Enable exception handling, and generate extra code needed to propagate exceptions. If you do not specify this option, GNU CC enables it by default for languages like C++ that normally require exception handling, and disabled for languages like C that do not normally require it. However, when compiling C code that needs to interoperate properly with exception handlers written in C++, you may need to enable this option. You may also wish to disable this option is you are compiling older C++ programs that don't use exception handling.

- Use the same size as double as for float (compiler parameter: -fshortdouble)

- Put extra information in generated assembly code (compiler parameter: -fverbose-asm) **:**

Put extra commentary information in the generated assembly code to make it more readable. This option is generally only of use to those who actually need to read the generated assembly code (perhaps while debugging the compiler itself).

-fno-verbose-asm, the default, causes the extra information to be omitted and is useful when comparing two assembler files.

Optimization :

- Optimize (compiler parameter: -O) **:**

Optimize. Optimizing compilation takes somewhat more time, and a lot more memory for a large function.

Without -O, the compiler's goal is to reduce the cost of compilation and to make debugging produce the expected results. Statements are independent: if you stop the program with a breakpoint between statements, you can then assign a new value to any variable or change the program counter to any other statement in the function and get exactly the results you would expect from the source code.

Without -O, the compiler only allocates variables declared register in registers. The resulting compiled code is a little worse than produced by PCC without -O.

With -O, the compiler tries to reduce code size and execution time.

When you specify -O, the compiler turns on -fthread-jumps and -fdefer-pop on all machines. The compiler turns on -fdelayed-branch on machines that have delay slots, and -fomit-frame-pointer on machines that can support debugging even without a frame pointer. On some machines the compiler also turns on other flags.

- Best optimization (compiler parameter: -O3) **:**

Optimize even more. GNU CC performs nearly all supported optimizations that do not involve a space-speed tradeoff. As compared to -O, this option increases both compilation time and the performance of the generated code.
-O3 turns on all optional optimizations like for loop unrolling and function inlining. It also turns on the -fforce-mem option on all machines and frame pointer elimination on machines where doing so does not interfere with debugging.
and also turns on the inline-functions option.

- Perform a number of minor optimizations (compiler parameter: -fexpensive-optimizations) **:**

Perform a number of minor optimizations that are relatively expensive.

Linker page :

- Link an objective C program (compiler parameter: -lobjc)**:**

You need this special case of the -l option in order to link an Objective C program.

- Generate debugging information (compiler parameter: -ggdb) **:**

Produce debugging information for use by GDB. This means to use the most expressive format available (DWARF 2, stabs, or the native format if neither of those are supported), including GDB extensions if at all possible.

- Do not use standard system files or libraries

Do not use the standard system startup files or libraries when linking. No startup files and only the libraries you specify will be passed to the linker.

One of the standard libraries bypassed by -nostdlib and -nodefaultlibs is libgcc.a, a library of internal subroutines that GNU CC uses to overcome shortcomings of particular machines, or special needs for some languages. In most cases, you need libgcc.a even when you want to avoid other standard libraries. In other words, when you specify -nostdlib or -nodefaultlibs you should usually specify -lgcc as well. This ensures that you have no unresolved references to internal GNU CC library subroutines. (For example, __main, used to ensure C++ constructors will be called; see collect2.)

- Compile for Win32 – no console (compiler parameter : -mwindows) **:**

This is used when working on source files (for projects, use "Do not create a console" in Project Options). It will the linker to create a windows program with no dos console.

Environment options

The environment options are described for the following sheets :

Preferences :

Default directory : You can select a directory that will be automatically use when creating, opening and saving files.

Create backup file : Check this if you want Dev-C++ to create a backup of the sources when saving.

Auto-save desktop and editor's position : If you want to use the same size of the editor and main window every time.

Save toolbars availability : Dev-C++ will restore the toolbars depending on latest session.

Auto-Arrange windows : Tile windows often.

Do not show project manager : Check this if you don't want to work with the Project Manager.

Make compile result window stay on top : After compiling, if this option is checked the compile window result will remain stay on top.

Minimize during execution : Dev-C++ will minimize when executing your program.

Show exit code after running : Tells Dev-C++ to show the exit code of your program after executing it.

Give the following parameters when executing a compiled project : Use this to call your program with parameters, in case it needs some.

Editor :

Editor font, size and background color : You can set these options to work with what you are used to.

Show lines number : This shows line numbers in the left the editor.

Auto-Indent : Use Dev-C++ tab settings (intelligent tabs).

Tab indent : Select Tab size. You can only use this feature if "Auto-Ident" is unchecked.

Show hint when scrolling : This will show a hint with the current line number when scrolling.

Do not show scrollbars : Check it if you don't want scrollbars in the editor.

Syntax colors :

You can modify the different syntax color of the C and C++ grammar, by first clicking on the type you would like to change. Then, select a color by clicking, as well as a text attributes (optional).
You will be available to view your change in the editor in this sheet.

Code completion :

Code completion is for speeding up your coding time. By pressing Ctrl+Space in the editor, a dialog will open containing a list of your usual coding text. Then you will be able to select a code and after pressing Enter it will be automatically inserted in the editor. You can edit them by modifying the text field on the sheet.

Misc. :

Assign .dev files with Dev-C++ : This is set by default. It allows you to open Dev-C++ project files by double-clicking in the Windows Explorer.

Assign .c and .cpp files with Dev-C++ : This is set by default. It allows you to open .c and .cpp files by double-clicking in the Windows Explorer.

Buttons : By clicking on these buttons you will be able to reset the default options of Dev-C++ and remove files associations.

Default code when creating new source files: Type in the editor the code you would like to have when creating new source files.

Contacting the author

Suggestions, remarks are always welcome, but please try to look at the {button FAQ,KL("FAQ (common problems)",1,`',`')} or the Bloodshed forum (http://forum.bloodshed.net/) before asking for a problem (90 % of problems are described there, and you can also find the up-to-date FAQ at http://www.bloodshed.net/faq.html).

You can contact Colin Laplace at webmaster@bloodshed.net

Mingw homepage : http://www.mingw.org/

Mumit Khan's homepage : http://www.xraylith.wisc.edu/~khan/software/gnu-win32/

Mingw discussion list : http://www.egroups.com/mingw32/

Creating a project

What is it?
Bloodshed Dev-C++ uses **project files** to keep source code together.
A project is a file with the extension **.dev**.

Why using projects?
You know the disadvantages of the C++ language. You have created a program with many files, lets say *15 *.cpp* files, and sometimes it is annoying to edit one-by-one source file and then recompile everything by command line, including the linking of all the resulting object files. About the navigation, you must every time find the file on your hard-disk.
With projects, you can *easily navigate through files* that's part of your program, and you can easily compile/assemble/link the entire program with one mouse-click !
You can compare it with Borland C++ Builder, that also have such feature.
Just open one single file, and you can edit or compile the whole program!

How to create it?
You can create a new project with the following steps:

1. Click on the **"File"** menu
1. Click the menu item **"New Project"**
3. A dialog will appear, and you can choose what kind of program you want to create.
4. You will get a prompt. You can enter a name for you project.
   The name doesn't matter, just enter any name you like.
   It can be **"My First Project"** or **"TApplication1"** or **"$!^#&$@^#^%!*&^#!(#"**.
5. In this step, you can save your project to disk.

6. Guess what...*your done!* You only have to save the file by using the File menu and choose **"Save All"**.

Mailing list

The purpose of this mailing list is to ask and answer questions about Dev-C++ and C++ programming. Please note that you may receive a few messages from the list everyday. You can subscribe to the Dev-C++ mailing list by sending a mail to:

majordomo@bloodshed.net

Leave the subject field empty, and type the following command in the body (not the subject) :

```
subscribe dev-cpp
```

You will then receive all the information for sending messages to the group of user.

If you wish to unsubscribe, send an email to majordomo@bloodshed.net with the following command in the body of the message :

```
unsubscribe dev-cpp myemail@mydomain.com
```

Help files

A tutorial for making projects as well as the complete Standard Template Library guide are available by clicking on the Help menu in Dev-C++.

These files are a good way to learn more about Dev-C++ and the C++ programming language.
Help files for the Mingw32 compiler are available at Mingw or Mumit Khan's site:

http://www.mingw.org/

http://www.xraylith.wisc.edu/~khan/software/gnu-win32/

Dev-C++ license agreements

Bloodshed Dev-C++ is distributed under the GNU General Public License. Be sure to read it before using Dev-C++.

GNU GENERAL PUBLIC LICENSE
Version 2, June 1991

Copyright (C) 1989, 1991 Free Software Foundation, Inc.
675 Mass Ave, Cambridge, MA 02139, USA
Everyone is permitted to copy and distribute verbatim copies
of this license document, but changing it is not allowed.

Preamble

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change free software--to make sure the software is free for all its users. This General Public License applies to most of the Free Software Foundation's software and to any other program whose authors commit to using it. (Some other Free Software Foundation software is covered by the GNU Library General Public License instead.) You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs; and that you know you can do these things.

To protect your rights, we need to make restrictions that forbid anyone to deny you these rights or to ask you to surrender the rights. These restrictions translate to certain responsibilities for you if you distribute copies of the software, or if you modify it.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must give the recipients all the rights that you have. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

We protect your rights with two steps: (1) copyright the software, and (2) offer you this license which gives you legal permission to copy, distribute and/or modify the software.

Also, for each author's protection and ours, we want to make certain that everyone understands that there is no warranty for this free

software. If the software is modified by someone else and passed on, we want its recipients to know that what they have is not the original, so that any problems introduced by others will not reflect on the original authors' reputations.

Finally, any free program is threatened constantly by software patents. We wish to avoid the danger that redistributors of a free program will individually obtain patent licenses, in effect making the program proprietary. To prevent this, we have made it clear that any patent must be licensed for everyone's free use or not licensed at all.

The precise terms and conditions for copying, distribution and modification follow.

GNU GENERAL PUBLIC LICENSE
TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

0. This License applies to any program or other work which contains a notice placed by the copyright holder saying it may be distributed under the terms of this General Public License. The "Program", below, refers to any such program or work, and a "work based on the Program" means either the Program or any derivative work under copyright law: that is to say, a work containing the Program or a portion of it, either verbatim or with modifications and/or translated into another language. (Hereinafter, translation is included without limitation in the term "modification".) Each licensee is addressed as "you".

Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running the Program is not restricted, and the output from the Program is covered only if its contents constitute a work based on the Program (independent of having been made by running the Program). Whether that is true depends on what the Program does.

1. You may copy and distribute verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and give any other recipients of the Program a copy of this License along with the Program.

You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

2. You may modify your copy or copies of the Program or any portion of it, thus forming a work based on the Program, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:

a) You must cause the modified files to carry prominent notices stating that you changed the files and the date of any change.

b) You must cause any work that you distribute or publish, that in whole or in part contains or is derived from the Program or any part thereof, to be licensed as a whole at no charge to all third parties under the terms of this License.

c) If the modified program normally reads commands interactively when run, you must cause it, when started running for such interactive use in the most ordinary way, to print or display an announcement including an appropriate copyright notice and a notice that there is no warranty (or else, saying that you provide a warranty) and that users may redistribute the program under these conditions, and telling the user how to view a copy of this License. (Exception: if the Program itself is interactive but does not normally print such an announcement, your work based on the Program is not required to print an announcement.)

These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Program, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Program, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Program.

In addition, mere aggregation of another work not based on the Program with the Program (or with a work based on the Program) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

3. You may copy and distribute the Program (or a work based on it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you also do one of the following:

a) Accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

b) Accompany it with a written offer, valid for at least three years, to give any third party, for a charge no more than your cost of physically performing source distribution, a complete machine-readable copy of the corresponding source code, to be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

c) Accompany it with the information you received as to the offer to distribute corresponding source code. (This alternative is

allowed only for noncommercial distribution and only if you received the program in object code or executable form with such an offer, in accord with Subsection b above.)

The source code for a work means the preferred form of the work for making modifications to it. For an executable work, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the executable. However, as a special exception, the source code distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.

If distribution of executable or object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place counts as distribution of the source code, even though third parties are not compelled to copy the source along with the object code.

4. You may not copy, modify, sublicense, or distribute the Program except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense or distribute the Program is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

5. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Program or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Program (or any work based on the Program), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Program or works based on it.

6. Each time you redistribute the Program (or any work based on the Program), the recipient automatically receives a license from the original licensor to copy, distribute or modify the Program subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties to this License.

7. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you

may not distribute the Program at all. For example, if a patent license would not permit royalty-free redistribution of the Program by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Program.

If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply and the section as a whole is intended to apply in other circumstances.

It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system, which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

8. If the distribution and/or use of the Program is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Program under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.

9. The Free Software Foundation may publish revised and/or new versions of the General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Program specifies a version number of this License which applies to it and "any later version", you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of this License, you may choose any version ever published by the Free Software Foundation.

10. If you wish to incorporate parts of the Program into other free programs whose distribution conditions are different, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

NO WARRANTY

11. BECAUSE THE PROGRAM IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

12. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

END OF TERMS AND CONDITIONS

Appendix: How to Apply These Terms to Your New Programs

If you develop a new program, and you want it to be of the greatest possible use to the public, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest to attach them to the start of each source file to most effectively convey the exclusion of warranty; and each file should have at least the "copyright" line and a pointer to where the full notice is found.

> <one line to give the program's name and a brief idea of what it does.>
> Copyright (C) 19yy <name of author>
>
> This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version.
>
> This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.
>
> You should have received a copy of the GNU General Public License along with this program; if not, write to the Free Software Foundation, Inc., 675 Mass Ave, Cambridge, MA 02139, USA.

Also add information on how to contact you by electronic and paper mail.

If the program is interactive, make it output a short notice like this
when it starts in an interactive mode:

> Gnomovision version 69, Copyright (C) 19yy name of author
> Gnomovision comes with ABSOLUTELY NO WARRANTY; for details type `show w'.
> This is free software, and you are welcome to redistribute it
> under certain conditions; type `show c' for details.

The hypothetical commands `show w' and `show c' should show the appropriate
parts of the General Public License. Of course, the commands you use may
be called something other than `show w' and `show c'; they could even be
mouse-clicks or menu items--whatever suits your program.

You should also get your employer (if you work as a programmer) or your
school, if any, to sign a "copyright disclaimer" for the program, if
necessary. Here is a sample; alter the names:

> Yoyodyne, Inc., hereby disclaims all copyright interest in the program
> `Gnomovision' (which makes passes at compilers) written by James Hacker.
>
> <signature of Ty Coon>, 1 April 1989
> Ty Coon, President of Vice

This General Public License does not permit incorporating your program into
proprietary programs. If your program is a subroutine library, you may
consider it more useful to permit linking proprietary applications with the
library. If this is what you want to do, use the GNU Library General
Public License instead of this License.

Authors

Dev-C++ IDE development :

- Colin Laplace : Main development
- Hongli Lai : IDE updates, splash screen and icons

Mingw32 compiler system development :

- Mumit Khan, J.J. Var Der Heidjen, Colin Hendrix and GNU coders.

VCL component authors (used in Dev-C++ IDE):

- SynEdit : powerful editor, with syntax highlighting, exporting,...
  Authors : Bruno Mikkelsen, Colin Laplace, David H. Muir, Michael Beck, Michael Hieke, Primoz Gabrijelcic, Stefan van As, Woo Young Bum

- Marscap : this is a "beautifuller" for the border of the window.
  Author: Chen Ken

- ToolBar97: great components for using powerful toolbars.
  Author: Jose Sebastian Battig

- TBackup : backup/compression
  Author: Alexander Halser

Contributing to the development

As both Dev-C++ and Mingw32 are under the {button GNU General Public License,KL("Dev-C++ license agreements",1,`',`')} ,you can download the source code that was used to built these software. You can then participate in improving these tools, or simply look at the code to learn from it.

Dev-C++ source code:

You can download it at the Dev page : http://www.bloodshed.net/dev/ This page contains all the newest information about the Dev-C++ development.

Mingw32 source code:

You can get it at Mumit Khan's site:
http://www.xraylith.wisc.edu/~khan/software/gnu-win32

Using your own Mingw and Cygwin compiler system

THERE IS NO NEED TO READ BELOW IF YOU GOT THE COMPLETE DEV-C++ PACKAGE WITH THE MINGW COMPILER SYSTEMS INCLUDED.

*** Configuring for Mingw compiler:**

Run Dev-C++, goto Compiler Options (in Options menu). Remove all C, C++ and Lib directories, and replace them by a .
Modify the Bin directory name to the one used in your Mingw installation (like C:\gcc-2.95.2\bin\). Click OK to save changes.

Open your autoexec.bat file (right-click on it in the windows explorer and click Edit). Add the following lines at the end of it (change the directories according to your Mingw distribution, the following assume you are using GCC 2.95.2 MSVCRT):

```
SET PATH=C:\gcc-2.95.2\bin;%PATH%
SET C_INCLUDE_PATH=C:\gcc-2.95.2\i386-mingw32msvc\include
SET CPP_INCLUDE_PATH=C:\gcc-2.95.2\include\g++-3;C:\gcc-2.95.2\i386-mingw32msvc\
include
SET GCC_EXEC_PREFIX=C:\gcc-2.95.2\lib\gcc-lib
SET LIBRARY_PATH=C:\gcc-2.95.2\lib\
```

Save your autoexec.bat file and reboot your system.

*** Configuring for Cygwin compiler:**

Run Dev-C++, goto Compiler Options (in Options menu). Remove all C, C++ and Lib directories, and replace them by a .
Modify the Bin directory name to the one used in your Cygwin installation (like C:\cygnus\cygwin-b20\H-i586-cygwin32\bin). Click OK to save changes.

Open your autoexec.bat file (right-click on it in the windows explorer and click Edit). Add the following lines at the end of it (change the directories according to your Cygwin distribution, the following assume you are using CYGWIN B20 full package):

```
SET PATH=C:\cygnus\cygwin-b20\H-i586-cygwin32\bin;%PATH%
SET C_INCLUDE_PATH=C:\cygnus\cygwin-b20\include
SET CPP_INCLUDE_PATH=C:\cygnus\cygwin-b20\include\g++;C:\cygnus\cygwin-b20\
include
SET GCC_EXEC_PREFIX=C:\cygnus\cygwin-b20\H-i586-cygwin32\lib\gcc-lib
SET LIBRARY_PATH=C:\cygnus\cygwin-b20\H-i586-cygwin32\i586-cygwin32\lib
```

Save your autoexec.bat file and reboot your system.

Making and using Templates

Dev-C++ can create and manage template files. Those templates contains information for creating automatically specific projects of your own. Dev-C++ contains 5 default projects, but it lets you design your own too with templates.

Making templates :

To create a new template, click on the File menu and then on "New template file".
This will bring you the Template Builder, which creates and registers template files in Dev-C++.

Template Information : Enter the name, description and category of your template. You can also select an icon, which will be used by the executable generated after compiling a template project.

Editor information : Here you can set the cursor position that will be set when creating a template file. For example, if your templates contains the following default code:

```
int main()
{

}
```

you should put the following cursor position :
Column = 4 (like for a TAB)
Row = 3 (points to the third line, where the text is empty).

Project information : This is for setting the default options your template will generate for a project. They are the same as in Project Options.

Code : Type in the text field the default C/C++ codes that will be used when creating a project from your template. Default code can be for example:

```
#include <stdio.h>
int main()
{

}
```

When you are ready to save and add your file to the Template list, click on the Save button, and type a filename in the dialog box that follows.

Using templates :

Creating a project from your template is the same as for creating usual projects. Click on the File menu in Dev-C++, then on "New Project…". On the following dialog, click on the "Custom Templates" tab sheet, this will bring you the list of available templates to create a project from. Select a template and press the OK button (or double-click on the selected

icon), and a new project will be created using the default options and code as you wrote in your template.

Deleting templates :

In the New Project dialog, select the template you are willing to delete and press the DEL (delete) button on your keyboard.