This guide is for students who are not computer science majors or are beginners in computer science. The goal of this guide is to teach you how to use the GDB debugging tool for codes written in C language.

GDB (GNU Debugger) is a powerful tool for debugging. You can use it from the command line to find and fix issues with programs, like crashes or logic errors. Using GDB helps programmers understand how their code works in more detail.

Before you start using GDB, install both GDB and the program you want to debug. We assume that you already know the basics of C language for this guide.

Be careful when you're debugging: you might run into system errors or crashes. Save your important data to avoid losing it.

Preparation Work

1. **Install GDB**
    a. On Linux systems, employ package managers for GDB installation. For instance, on Ubuntu, execute the following command:

```
sudo apt install gdb
```

    b. Under the Windows system, install MinGW or Cygwin to emulate a Linux environment.

    For MinGW installation, follow these steps:

        1. Navigate to the MinGW official website and download the installation package, like mingw-get-setup.exe.
        2. Run the downloaded installation package. In the GUI, select to install GCC and GDB.
        3. Upon completion of installation, append the bin directory of MinGW to the environment variable Path. For example, if the directory is C:\MinGW\bin, add ;C:\MinGW\bin to Path.
        4. Open a command-line window. Enter gcc -v to verify successful GCC configuration. Input gdb -v to confirm GDB setup.

    Now, you can use the gcc and gdb commands directly in the Windows command line.

    Next, write and compile C language code using gcc, then debug with gdb. This allows you to experience the complete C language development and debugging process under the Windows environment.

Should you encounter any issues during configuration, troubleshoot based on the error messages received. Mastering the setup of these tools is fundamental to developing in the C language.

2. **Compile the Program and Include Debugging Information**
While compiling a C language program, add the -g parameter to the gcc/g++ compiler as illustrated below:

```
gcc -g hello.c -o hello
```

Here, '-g' means to generate and include debugging information. When compiling an executable file, this option incorporates debugging details such as line numbers and function names from the C language source code.

Without using the '-g' parameter, the compiler produces only machine code without any debugging details, making the debugging process extremely challenging.

Therefore, always add the '-g' parameter when compiling C language programs. When debugging with GDB, use executable files that contain debugging information. This practice enables the display of code line numbers and function names.

Use the '-o' parameter to designate the name of the output executable file. For instance, compile the source code 'hello.c' into an executable file named 'hello'.

In summary, to compile a C program into a debuggable executable, use the following command format:

```
gcc -g source.c -o executable
```

Generate the executable file with included debugging information so that it can be directly utilized by the GDB debugging tool.

3. **Key GDB Commands**
- 'list': Display code
- 'break': Set breakpoints
- 'delete': Remove breakpoints
- 'next': Step over code within functions
- 'continue': Resume execution
- 'print': Output variable values
- 'backtrace': Show function call stack information
- 'step': Debug step by step
- 'finish': Step out of the current function
- 'info breakpoints': Display all breakpoints

- ‘watch’: Monitor expression values
- ‘display’: Show expression values at each step
- ‘quit’: Exit GDB

Note: Use the help command to view assistance for all commands.

Debugging Practice Example

We will use a Hello World program with a bug:

```
#include <stdio.h>

int main() {
  int i = 0;
  while (i < 5) {
    printf("Hello World!");
    i--; // The original intention was to write i++, but it was mistakenly written as i--.
  }

  return 0;
}
```

This program intends to print "Hello World" five times, but due to a code error, it results in an infinite loop.

Now, let’s debug this program using GDB.

1. **Compile the Program**
Compile with the ‘-g’ parameter:

```
gcc -g hello.c -o hello
```

2. **Launch GDB**

```
gdb ./hello
```

3. **Set breakpoints**
Place a breakpoint at the first line of the main function:

```
break main
```

4. **Run the program**

```
run
```

The program will pause at the breakpoint.

5. **Step Through the Code**
Use the ‘step’ command to execute step-by-step:

```
step
```

Now we notice that the value of ‘i’ does indeed decrease, confirming that ‘i--' caused the error.

6. **Resume Execution**
Employ the ‘continue’ command to resume running, verifying that an infinite loop indeed occurs.

```
continue
```

7. **Modify Code**
Correct i-- to i++, recompile and debug, resolving the issue.

The above demonstrates how to use GDB’s basic features to assist in debugging C language code. GDB significantly enhances debugging efficiency and is an indispensable tool for C language programmers.