

目次

1. GCCとは	1
2. 作業フォルダの作成.....	1
3. 作業の概略	2
4. エディタを用いたプログラムの作成	3
4. 1 NoEditor の起動(①)	3
4. 2 プログラムの作成(②)	3
4. 3 プログラムの保存(③)	4
5. GCCを用いたコンパイル・リンク	5
5. 1 コマンドプロンプトの起動(④).....	5
5. 2 カレントフォルダの移動(⑤)とファイルの表示	6
5. 3 gcc を用いてコンパイル(⑥).....	7
5. 4 コンパイル・リンク時のエラー(⑦)	7
6. 実行 (⑧)	8

1. GCCとは

GCC(ジーシーシー)は、GNU の GNU Compiler Collection (グニューコンパイラコレクション) が開発したコンパイラ群のことです。フリーソフトウェアとして公開されており、Linux や FreeBSD の開発に不可欠のものとなっています。また、現在では多くの環境で標準的な C コンパイラとして採用されています。

このテキストでは、**GCC**の C 言語用コンパイラである"**gcc**"のプログラムを用いて、作成したプログラムをコンパイル、リンクし、実行ファイルを作成する手順について説明します。

Tips: 参考

GNU コンパイラコレクション:

<https://ja.wikipedia.org/wiki/GNUコンパイラコレクション> (Wikipedia, 参照日 2023 年 9 月 20 日)

2. 作業フォルダの作成

プログラミングによって、**ソースプログラム**と**実行ファイル**が作成され、頻繁に書き換えや消去が行われます。他の文書ファイルなどと混在して不注意でファイルを消さないためにも、予めソースプログラムや実行ファイルを保存する**作業フォルダ**を作成しておきます。

Windows では、ユーザごとに C ドライブのドキュメント (Documents) フォルダが作成されます。このフォルダのデスクトップ上にプログラミング用の作業フォルダを作成します。

エクスプローラーでドキュメント (Documents) フォルダを開きます。次に左上にある**[新規作成]**メニューを開き、**[フォルダー]**を選びます(図 1)。フォルダの名前はわかりやすいものが良いので、ここでは、"**GC**"とつけることにします(図 2)。この授業では常にこのフォルダ内で作業を行うことにすれば、他のファイルと混同することがありません。

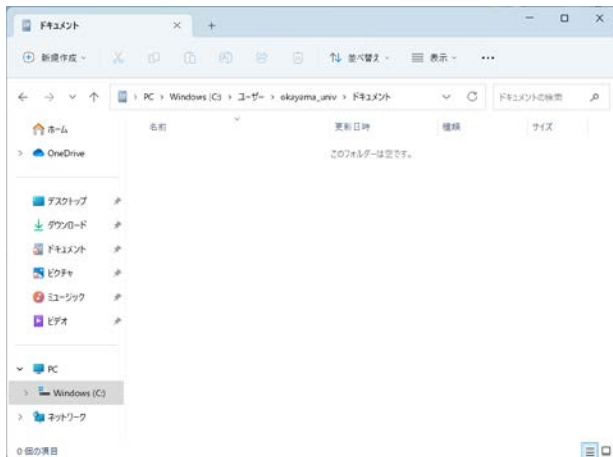


図 1 Documents へのフォルダの作成

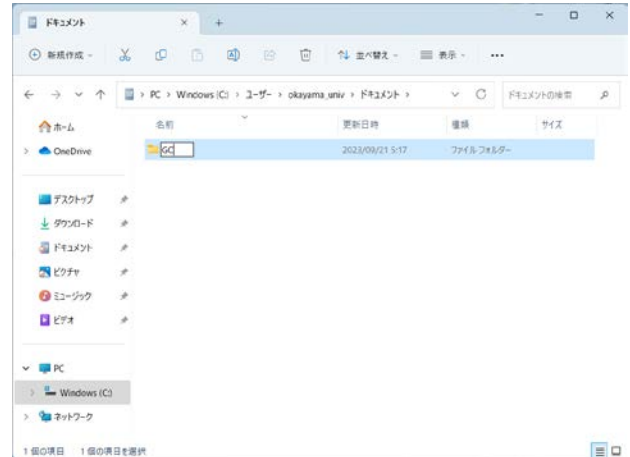


図 2 GC フォルダの作成

3. 作業の概略

プログラムの作成は、大きく分けて

- ・ エディタを用いたプログラムの作成 (第 4 節)
- ・ コマンドプロンプトでの gcc によるコンパイル (第 5 節)
- ・ プログラムの実行 (第 6 節)

に分けられます。これらの作業を行っているときのデスクトップの様子が図 3 です。右側のウィンドウで NoEditor を用いたプログラムの入力、左側のウィンドウでコマンドプロンプトによる作業をそれぞれ行っています。以降の 4 節～ 6 節では、これらの作業について説明します。

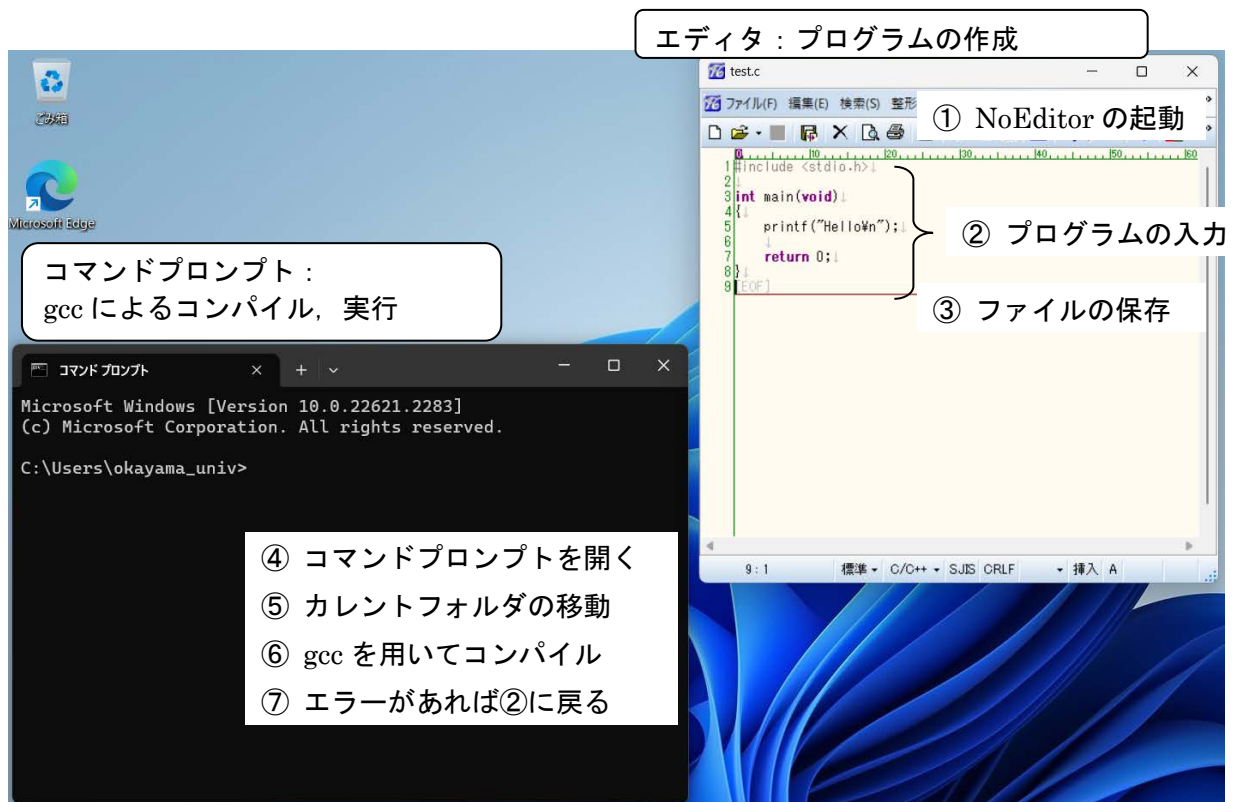


図 3 プログラム作成，コンパイル，実行時のデスクトップの様子

4. エディタを用いたプログラムの作成

プログラムを入力するソースファイルの形式は、テキスト形式です。従って Windows に付属のメモ帳(Notepad)でも入力できます。ここでは、便利なフリーソフトウェア NoEditor を用いたプログラムの入力方法を説明します.. NoEditor は以下の URL からダウンロードできます。

<https://www.vector.co.jp/soft/winnt/writing/se300436.html> (Vector)

4. 1 NoEditor の起動 (①)

インストールした NoEditor は、下記の 2 種類のいずれかの方法で起動するのが簡単です。

- ・[スタートメニュー]-[すべてのアプリ]-[Yokka]-[NoEditor]を選んでクリック。(図 4)
- ・ウィンドウズキーを押し、"noeditor"と入力した後、Enter キー。(図 5)

すると NoEditor の画面が表示され、新たなソースファイルを作成できる状態になります。

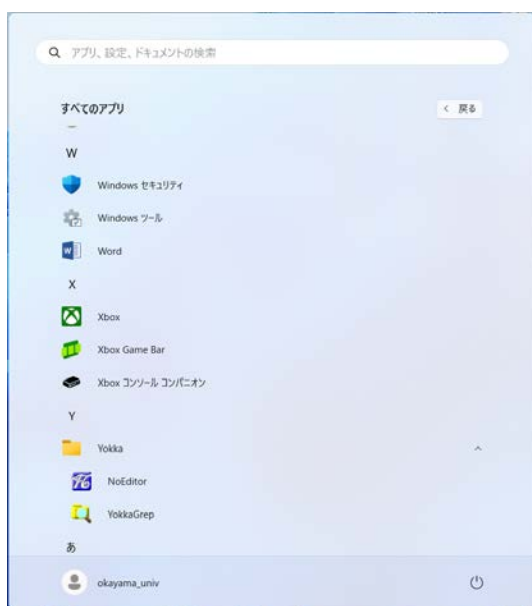


図 4 すべてのアプリから NoEditor を起動

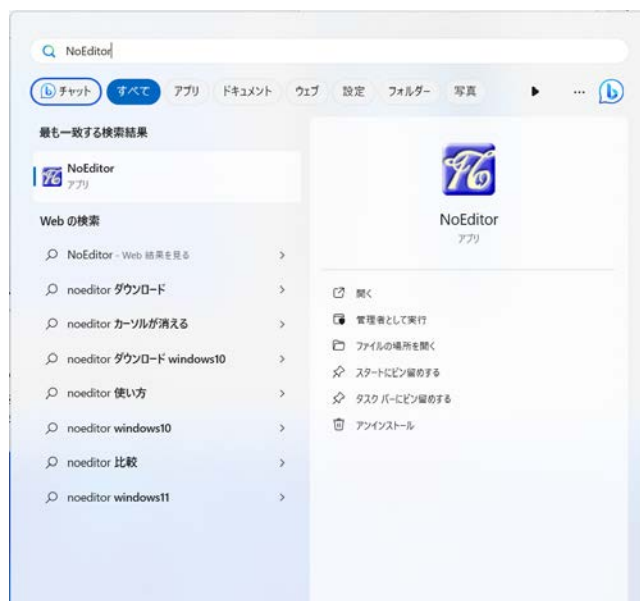


図 5 NoEditor を検索して起動

4. 2 プログラムの作成 (②)

プログラムを入力していきます。最初から入力画面の左上に[EOF]があります。これは"End of file"の略で、入力された内容の末尾を意味します。文字を入力すると[EOF]は自動的に後ろへ移動します。

サンプルファイルとして、図 6 のプログラムを入力してください。プログラムは全て半角で入力してください。全角文字を入力すると、コンパイル時にエラーが表示されます。(例外として、コメント部分(`/* */`)で囲まれる部分(テキスト p.25)と、書式付入出力関数(テキスト p.112)のダブルコーテーション(" ")で囲まれる範囲には、全角文字での入力が許されています) また、`printf`や`return`の左側の空白(インデント)の入力には TAB キーを使います。さらに、NoEditor では、入力内容に応じて自動的にインデントが行われます。{ と } の位置を同じ列にするとプログラムの構造が分かりやすいものとなります。

```
#include <stdio.h>

int main(void)
{
    printf("Hello\n");

    return 0;
}
[EOF]
```

図 6 サンプルプログラム

4. 3 プログラムの保存 (③)

作成したプログラムを保存します。保存は[ファイル(F)]-[名前をつけて保存(A)]を選びます。保存場所は、2. で作成したドキュメント (Documents) フォルダの GC フォルダになるようにしてください。

(図 7)

ファイル名の命名には、次の点を守ってください。

- ・ ファイル名はできるだけ半角文字で入力してください。全角文字でもコンパイルは可能ですが、作業時の入力の手間が増えます。
- ・ ファイルの名前の末尾に、".c"を半角文字で必ずつけてください。これは作成したファイルが C 言語のファイルであることを示す拡張子の規則であり、これ以外の拡張子をつけると、誤動作の原因となります。(ファイル名の例 : test.c)

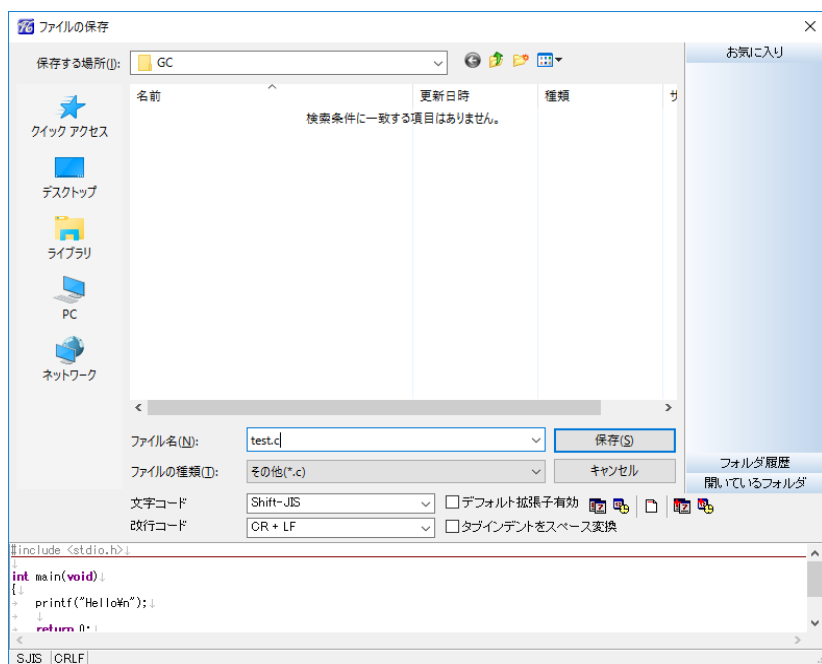


図 7 NoEditor でのファイルの保存

正しく保存されると、NoEditor のプログラム入力画面の各文字や記号の色が変わります。これは入力したファイルを NoEditor が C 言語のファイルと認識したためで、C 言語の予約語や関数、波括弧 ({})などに色が付きます。これによってプログラムの構造がわかりやすくなるだけでなく、入力ミスなどを見つける手がかりにもなります。既に作成した C 言語のソースファイルを開いたときは、初めから該当する文字・記号に色が付きます。

コンパイル・リンク時にエラー表示が出た場合は、再度 NoEditor でプログラムを修正します。このため NoEditor のウィンドウはプログラムが完成するまでは開いたままの方がよいでしょう。

Tips: 他メディアへの保存 (バックアップ)

作業フォルダ内には、ソースファイル(拡張子:".c")と実行ファイル(拡張子:".exe")が存在します。大切なのはソースファイル(".c")の方です。ドライブの内容が消える場合もありますので、重要なソースファイルは USB メモリ等にバックアップ (コピー) しておくとい良いでしょう。

5. GCCを用いたコンパイル・リンク

プログラムが入力できたら、次にプログラムをコンパイルします。コンパイルは、ソースプログラムに書かれた人間に分かりやすい命令文を、マシン語に翻訳する作業で、この作業を行わないと、コンピュータは命令文を理解することができません。さらにリンクは、作成したプログラムをあらかじめ用意されたライブラリ関数と結びつけ、実行可能なプログラムを作成する作業です。これらの作業が問題なく終了して初めて、コンピュータで実行可能なプログラムが完成します。

GCCでは、gcc プログラムでコンパイルとリンクを一度に行えます。

5.1 コマンドプロンプトの起動 (④)

gcc はコマンドプロンプト(コマンドウィンドウ、コンソール画面、MS-DOS ウィンドウ、DOS 窓とも呼ばれる)で直接プログラム名を入力して起動させるプログラムです。コマンドプロンプトは、MS-DOS (Microsoft Disk Operating System)コマンドを入力するためのエントリーポイントを提供する Windows の機能です。特徴は、Windows の GUI (Graphical User Interface)を使用せずにコンピュータでタスクを実行できる点です。

gcc を使用するためには、まずコマンドプロンプトを起動します。コマンドプロンプトは、ウィンドウズキーを押した後、"cmd"と入力して Enter キーを押すことによって起動できます。(図8)

コマンドプロンプトを起動すると、図9のような画面が開きます。">"の右側で点滅する白い四角をカーソルと呼び、キーボードから入力した文字がカーソルの位置に表示されます。文字列を入力して Enter キーを押すと、指定するプログラムが実行されます。(Windows より前のコンピュータは、全てコマンドプロンプトで制御していました。また Linux でも基本的に同様の方法で制御します。)

">"の左側の文字は、現在作業を行っているフォルダ(カレントフォルダ)の位置を示しています。例えば図9では、

C:\Users\okayama_univ>

と表示されています。これは、カレントドライブがCドライブで、カレントフォルダが"Users\okayama_univ"であることを示しています。("\はフォルダ名の区切りを意味し、Users フォルダの下にある okayama_univ フォルダがカレントフォルダになっていることを意味しています。)(バックスラッシュ「\」は、円マーク「¥」で表されることも多いです)

コマンドプロンプトのウィンドウは、右上の×をクリックするか、カーソル位置に"exit"と入力して Enter キーを押せば、閉じます。



図8 コマンドプロンプトの起動方法

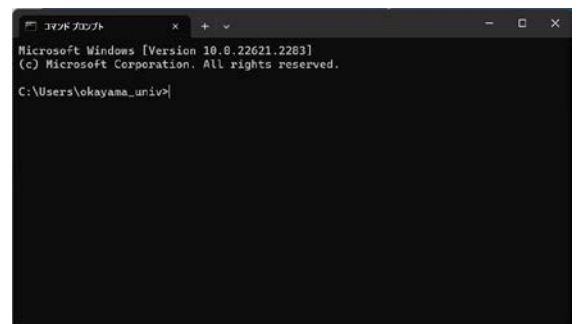


図9 コマンドプロンプト

Tips: 参考

これだけは覚えておきたい Windows のコマンドプロンプトの使い方 :

<http://www.atmarkit.co.jp/ait/articles/1502/13/news147.html> (参照日 2023 年 9 月 20 日)

5. 2 カレントフォルダの移動 (⑤) とファイルの表示

コマンドプロンプトを起動した時のカレントフォルダは、ログインしたユーザの個人設定を保存する C ドライブの "Users\okayama_univ" となっています。これを 2. で作成した "Documents" の "GC" に移動します。

移動には、コマンドプロンプトの、ドライブを変更する命令 (z:) と、カレントフォルダを移動する命令 (cd) を用います。コマンドプロンプトでは、大文字 / 小文字の区別はありません。

- ・ **z:** ドライブを変更する命令
 ドライブ名の後コロン(:)を入力して Enter キーを押すと、そのドライブに移動する。
 (c: と入力すると C ドライブに移動)
- ・ **cd** カレントフォルダを移動する命令
 cd 現在のカレントフォルダの場所を表示
 cd .. カレントフォルダを一つ上のフォルダに移動
 cd \ 現在のドライブのもっとも上位のフォルダに移動

これらのコマンドを用いて、C ドライブの "Documents" の "GC" に移動する方法を、下記に示します。

方法 1 :	>cd Documents	← "Documents" に移動
	>cd GC	← "Documents" の下の GC に移動

方法 2 :	>cd Documents\GC	← "Documents" の下の "GC" に移動
--------	------------------	----------------------------

方法 3 :	GC フォルダを他のドライブに設定した場合 (Z ドライブの場合を示す)	
	>z:	← カレントドライブを Z ドライブに移動
	>cd Documents\GC	← "Documents" の下の "GC" に移動

Tips: フォルダ名にスペースを含む場合

フォルダ名 / ファイル名をダブルコーテーション (" ") で囲むと、指示通りに移動できます。

Tips: 文字の補完

cd do まで入力し、TAB キーを押すと残りの文字が補完されて、cd Documents と簡単に入力できます。

Tips: コマンドプロンプトを開く (第三の方法)

エクスプローラーで GC フォルダを開き、Shift+右クリックで出てくるメニューから "コマンドウィンドウをここで開く" を選ぶと、カレントフォルダでウィンドウを開くことが可能です。ただしフォルダ操作を理解するためにも、上記手法 1 または手法 2 に慣れるようにしてください。

カレントフォルダ内のフォルダやサブフォルダの情報を表示するのが、dir というコマンドです。

- ・ **dir** ファイル・フォルダの作成年月日、容量などを表示する命令

dir コマンドを用いて、カレントフォルダに 4. で作成したプログラムが正しく保存されていることを確認してください。 (図 10)

5. 3 gcc を用いてコンパイル (⑥)

最も簡単なコンパイル方法は、

gcc ファイル名

です。しかしこのままでは作成されるプログラム名が"a.exe"になってしまいますので、下記のように入力するようにしてください。

・プログラムをコンパイル・リンクするコマンド

gcc **ファイル名** -o **実行ファイル名** -lm -ansi -pedantic -Wall

-o 実行ファイル名を指定するオプション。その後のファイル名が実行ファイル名となる。実行ファイルの拡張子は、必ず".exe"とする。

-lm 数学関数(sin, sqrt 等)を用いる場合に必要なオプション。数学関数を使っていたら、それに必要なライブラリがリンクされる。

-ansi -pedantic 厳格な ANSI 仕様でのコンパイル。(本講義必須)

-Wall 文法チェックで疑わしい箇所を全て表示する。(推奨)

例 : gcc **test.c** -o **test.exe** -lm -ansi -pedantic -Wall

ソースプログラムが **test.c**, 実行ファイル名が **test.exe**

Tips: gcc のオプション

gcc のオプションについては、

gcc --help

と入力することによって見るすることができます。必要に応じて参考になさってください。

Tips: コマンドプロンプトの履歴機能

コマンドプロンプトでは同じ様な入力をする場合、矢印キーの"↑"を押すと前に入力したコマンド内容呼び出せます。この履歴はいくらでも過去にさかのぼることができ、また呼び出したものを部分的に修正して使うことも可能です。

5. 4 コンパイル・リンク時のエラー (⑦)

エラーがなければ、実行ファイル(test.exe)がカレントフォルダに作成されます。dir コマンドを用いて、実行ファイルの作成日時が最新であることを確認してください。(図 10)

入力ミスやプログラムの文法的なエラーがある場合は、gcc がコンパイル・リンク時にチェックしてエラーを表示し、コンパイル・リンクの作業を中止します。エラーが表示される場合は、エラーの内容を参考に、プログラムを修正してください。

エラーの例を 3 つあげておきます。ソースファイル名(test.c)の後の数字がエラーと考えられるプログラムの先頭からの行数を示しています。その行かその前にエラーがある場合が多いです。

例 1 : 文末にセミコロン(;)が無い場合

```
Z:\My Documents\GC>gcc test.c -o test.exe -lm -ansi -pedantic -Wall
test.c: In function 'main':
test.c:7:2: error: expected ';' before 'return'
    return 0;
    ^
test.c:8:1: warning: control reaches end of non-void function [-Wreturn-type]
}
```



```
1 #include <stdio.h>
2
3 int main(void)
4 {
5     printf("Hello\n")
6
7     return 0;
8 }
```

例 2 : ダブルコーテーション(")が不足する場合

```
Z:\My Documents\GC>gcc test.c -o test.exe -lm -ansi -pedantic -Wall
test.c: In function 'main':
test.c:5: error: missing terminating " character
test.c:7: error: parse error before "return"
```

```
1 #include <stdio.h>
2
3 int main(void)
4 {
5     printf("Hello\n");
6
7     return 0;
8 }
```

例 3 : 関数名を間違えている場合

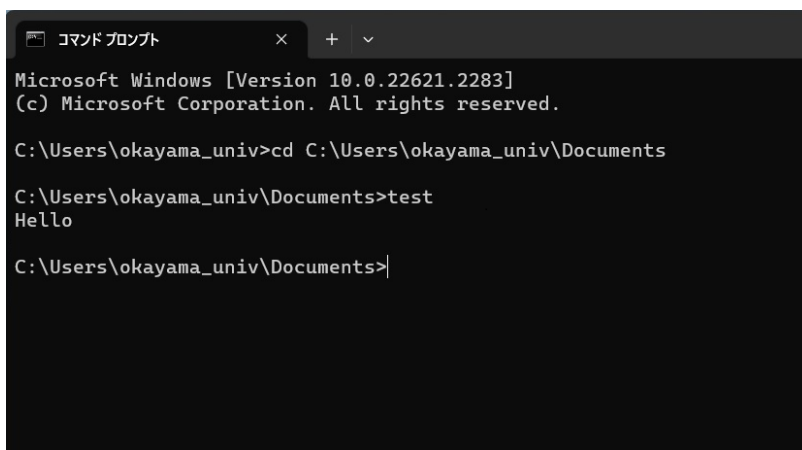
```
Z:\My Documents\GC>gcc test.c -o test.exe -lm -ansi -pedantic -Wall
test.c: In function 'main':
test.c:5:2: warning: implicit declaration of function 'print' [-Wimplicit-
function-declaration]
    print ("Hello\n");
    ^
/cygdrive/c/Users/OKAYAM~1/AppData/Local/Temp/ccP5Ecqd.o:test.c:(.text+0x15):
undefined reference to `print'
/cygdrive/c/Users/OKAYAM~1/AppData/Local/Temp/ccP5Ecqd.o:test.c:(.text+0x15):
relocation truncated to fit: R_X86_64_PC32 against undefined symbol `print'
collect2: error: ld returned 1 exit status
```

```
1 #include <stdio.h>
2
3 int main(void)
4 {
5     print("Hello\n");
6
7     return 0;
8 }
```

6. 実行 (⑧)

正しくプログラムがコンパイルできれば, コマンドウィンドウで**実行ファイル名**(拡張子. exe は省略可)を入力して **Enter** キーを押して下さい。プログラムが実行されます。(図 10)

プログラムが予定通り実行されない場合は, 再度プログラムを修正し, コンパイルしなおしてください。(②へ戻る)



```
コマンドプロンプト
Microsoft Windows [Version 10.0.22621.2283]
(c) Microsoft Corporation. All rights reserved.

C:\Users\okayama_univ>cd C:\Users\okayama_univ\Documents

C:\Users\okayama_univ\Documents>test
Hello

C:\Users\okayama_univ\Documents>|
```

図 10 プログラムの実行

Tips: プログラムが終了しない or 動き続けて止まらない

実行したプログラムにバグがあって終了できなくなってしまうたら, たいていの場合は **Ctrl + C** で強制終了できます。C プログラムが無限ループに陥った場合も, この操作でプログラムを終了させることができます。

Tips: それでも止まらない (ハングアップした) 場合の対処

ハングアップしたときにコンピュータの電源を切ることは, ハードディスクなどに物理的損傷を与えたり Windows システムファイルを壊したりする場合があります。下記の操作で Windows タスクマネージャを起動して, 該当プログラムを強制終了してください。

Ctrl + Shift + ESC (Ctrl キー, Shift キー, ESC キーを同時に押す)