実践報告

土曜講座におけるC言語学習プログラム

― オリジナルテキストの作成と運用 ―

C Language Learning Program in the Saturday Course:

Creation and Management of Original Text

石部忠之　　国際学院中学校高等学校

国際学院中学校高等学校において、希望生徒を対象とした土曜講座（Versatile Saturday）を担当することとなり、プログラミング言語として長い歴史をもつ、C言語講座を通年（10回程度）で実施するにあたり、オリジナルテキストを作成した。受講生徒は任意で市販のテキストを購入することも可能であるが、環境構築、プログラム構文の説明、サンプルプログラム、課題を網羅することにより、オリジナルテキストのみでの受講で問題なく進めることができた。

キーワード:構造化プログラム言語、テキストエディタ、文字コード

１. はじめに

本校における土曜講座は年間 10 回程度開講可能であるため、90 分×2 コマ×10 回のボリュームで環境構築、基本構文の説明、簡単なアプリケーションの作成といった内容を網羅するテキストを準備した。対象となる生徒はプログラム言語の初学者を想定して作成した。また、基本的には各自が自前のノートパソコンを持参して講座を進めることとし、インストールするアプリケーションはすべてフリーソフトで行うこととした。

また、原則として Windows に対する環境構築を前提としたテキストとなっているが、Mac 対応の設定ファイルは別途用意した。Windows と Mac ではインストールするアプリケーションや内部で処理する文字コードが異なるなど、配慮が必要となる部分は多いため、注意が必要である。

２. 環境構築

2-1 コンパイラのインストールと設定

MacOS には、標準コンパイラである clang が入っている場合はインストール不要だが、Windows の場合、別途 C コンパイラをインストールする必要がある。Windows で動作するフリーの C コンパイラはいくつかあるが、今回は Minimalist GNU for Windows プロジェクトの MinGW をインストールすることとした。実際のインストール手順を示した文章は次の図１のとおりである。

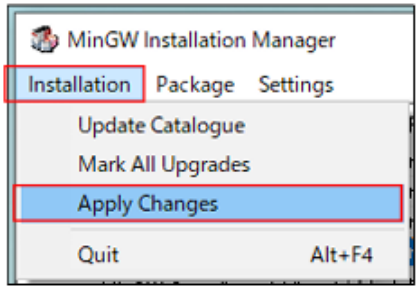
MinGW をインストール

ブラウザ（Edge，Chrome など）から mingw-get 検索

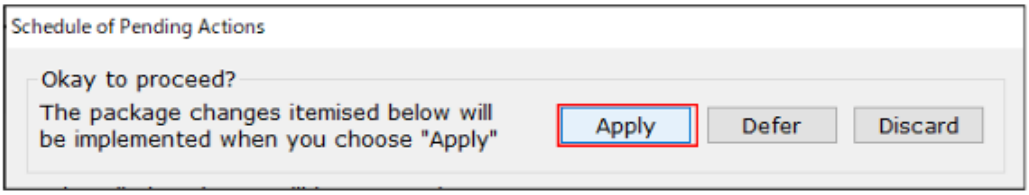
OSDN の「mingw-get-setup.exe をダウンロード」をクリック

（ダウンロードした）「mingw-get-setup.exe」をダブルクリック
⇒「Install」⇒「Continue」

（MinGW Installation Manager 左メニューの Basic Setup を選択）
☑mingw32-base-bin
☑mingw32-gcc-g++-bin
にチェックを入れて
（上のメニューの）「Installation」⇒「Apply Changes」をクリック



「Apply」をクリック



（終わったら）「Close」をクリックして、Installation Manager を終了

図 1　C コンパイラのインストール手順

コンパイラをインストールしただけでは正常に動作しないため、Path を通しておく必要がある。そこで、環境変数の Path に「C:¥MinGW¥bin」を追加する。

2-2 テキストエディタのインストールと設定

次に、テキストエディタをインストールしていく。Windows 標準のメモ帳でも構わないが、近年はフリーでありながら高機能なエディタがいくつか存在するので、それを利用することにした。中でも VisualStudioCode（VSCode）はフリーで高機能、カスタマイズがしやすく、プロも利用しているテキストエディタであるので、こちらを使うことにした。当然、Windows 用だけでなく、Mac や Linux 用も用意されているので、様々な環境に対応している。

以下に、その手順を図 2 に示す。

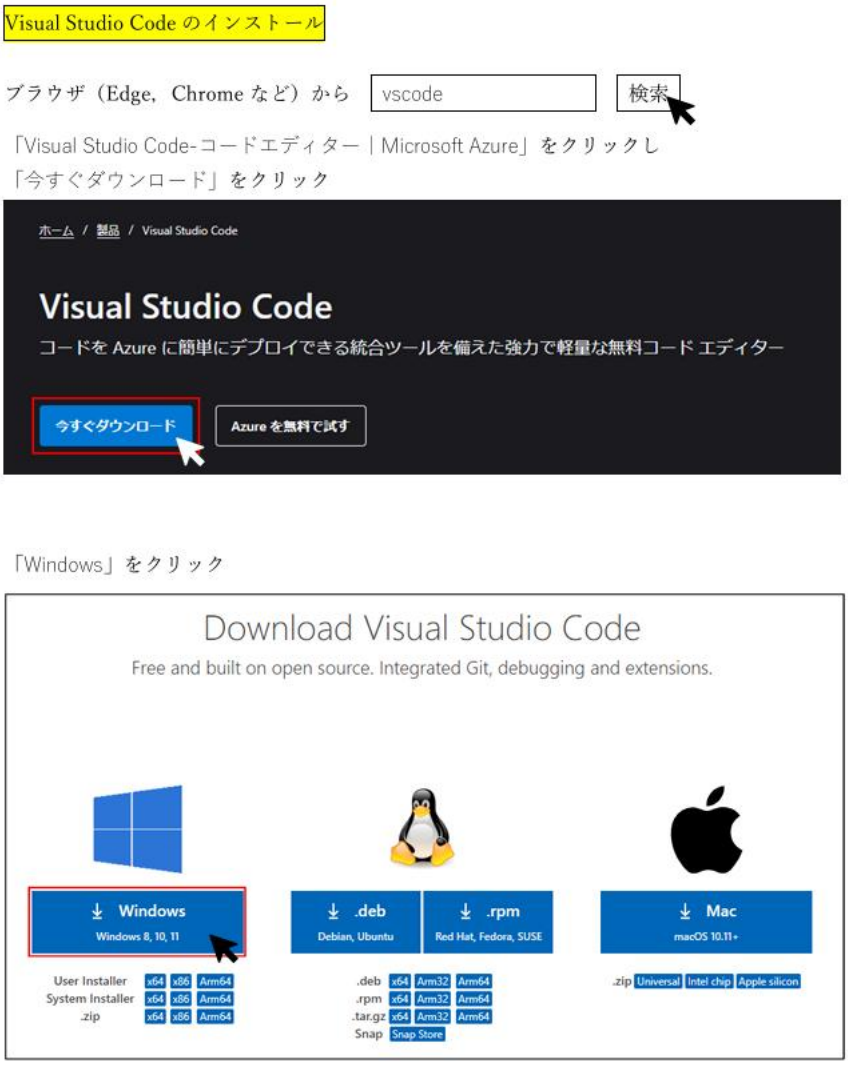


図 2　VSCode のインストール手順

なお、先ほど述べたように VSCode は高機能ではあるが、設定ファイルの作成がやや難解であるため、以下の json ファイルを作成しておく必要がある。図 3 は tasks.json、図 4 は settings.json の設定例である。

```
{
    "version": "2.0.0",
    "tasks": [
      {
        "label": "Clang",
        "type": "process",
        "command": "C:¥¥MinGW¥¥bin¥¥gcc.exe",
        "args": [
          "${file}",
          "-o",
          "${fileDirname}¥¥${fileBasenameNoExtension}.exe"
        ],
        "problemMatcher":[],
        "group": {
          "kind": "build",
          "isDefault": true
        }
      }
    ]
  }
```

図 3　tasks.json の設定例

なお、command の部分には、コンパイラの Path を、args の部分には、コンパイル実行時に指定する引数を設定しておくことで、コンパイル時の作業が軽減される。

```
{
    "workbench.colorTheme": "Default High Contrast",
    "editor.fontSize": 24,
    "editor.formatOnPaste": true,
    "editor.formatOnSave": true,
    "editor.formatOnType": true,
    "terminal.integrated.fontSize": 24,
    "terminal.integrated.defaultProfile.windows": "Command Prompt",
    "terminal.integrated.profiles.windows": {
        "Command Prompt": {
            "path": [
                "${env:windir}¥¥Sysnative¥¥cmd.exe",
                "${env:windir}¥¥System32¥¥cmd.exe"
            ],
            "args": [
                "/k",
                "chcp",
                "65001",
                "&",
                "cd",
                "¥¥c"
            ],
            "icon": "terminal-cmd"
        }
    }
}
```

図 4　settings.json の設定例

ここで、文字コードを UTF-8（chcp␣65001）に、作業ディレクトリを C ドライブの直下に作成した C というフォルダに設定している。なお、日本版 Windows では、Shift-JIS が標準の文字コードであるが、世界的な流れとしては UTF-8 などのマルチバイト文字が主流となっており、メモ帳のデフォルトも UTF-8 に変更になっている。また、「¥」記号は環境によって「\」（バックスラッシュ）で表示される場合がある。

３．Ｃ言語の特徴

1970 年代に開発された言語であるＣ言語は、あまりプログラミングの初学者向けではない部分もあるが、今回の講習で用いることとした理由は以下の通りである。

- 構造化プログラミング言語として、小規模なプログラムを組むのに適している。（逆に大規模なプログラム開発は、オブジェクト指向型プログラミング言語を学習して進めていくのがよい）
- ポインタ変数を扱うことができるため、内部構造を理解しながら学習を進めることができる。（ただし、適切に扱わなければ暴走するプログラムができてしまうため注意が必要である）
- Ｃ＋＋やＪａｖａなどのオブジェクト指向プログラミング言語は、Ｃ言語をお手本としており、多くのプログラミング言語に影響を与えている。

また、フリーフォーマットであることから、できるだけ読みやすい記法を心掛けるべきであるが、その点も初学者にはやや難しい面もある。そこで、VSCode の自動整形機能を活用することで、その点についてもクリアできると考えた。例えば、図 5 や図 6 のように入力した場合でも、保存時に自動的に図 7 のように見やすく、字下げ（インデント）処理や原則 1 行 1 文の形式に整えてくれる。日頃から見やすいプログラムに触れることで、字下げなどの必要性なども自然と身につくことが期待できる。

```
#include <stdio.h>
int␣main(){printf("Hello world!¥n");return␣0;}
```

図 5　1 行に 2 文以上入れたプログラム

```
#include <stdio.h>
int␣main()
{
printf("Hello world!¥n");
return␣0;
}
```

図 6　字下げ（インデント）をしていないプログラム

3-1 構成内容と図解説明

コンパイラやエディタをインストールして、設定作業を完了した際に最初に実行するのは「Hello world!」（図 7）である。

```c
#include <stdio.h>
int␣main()
{
    printf("Hello world!¥n");
    return␣0;
}
```

図 7　hello.c のソースコード

なお、tasks.json で設定した通り、hello.c をコンパイルするとファイル名は自動的に hello.exe となる。もし、実行ファイル名を変更したければ、-o オプションのあとに、作成したい実行ファイル名を設定すればよい。

今回のプログラムが正しく実行されれば、コマンドプロンプトの画面に「Hello world!」と表示されるはずである。ちなみに最後の¥n は改行コードを意味しており、動作環境を問わず、表示を整えるためにも¥n をつけることを習慣化しておくべきである。参考まで、改行コードを入れなかった場合のソースコード（図 8）、動作例 Windows（Command Prompt）（図 9）と Linux（図 10）を示しておく。

```c
#include <stdio.h>
int␣main()
{
    printf("Hello world!");
    return␣0;
}
```

図 8　改行コードなしの Hello world! ソースコード

```
E:¥c>gcc hello.c
E:¥c>a.exe
Hello world!
E:¥c>
```

図 9　Windows（Command Prompt）での実行例

```
$ gcc hello.c
$ ./a.out
Hello world!$
```

図 10　Linux での実行例

「Hello world!」が無事に表示されれば、環境設定などは問題ないので、次に、分岐（if 文、switch 文）や繰り返し（while 文、for 文）などの学習を進めていく。その際に、フローチャ

ートと併せて示していくことで、動きが正しく理解できるように配慮している。（図 11）

```c
#include <stdio.h>
int main()
{
    int a, b;
    printf("a = ");
    scanf("%d", &a);
    printf("b = ");
    scanf("%d", &b);
    if (a >= b) {
        printf("a は b 以上¥n");
    }
    return 0;
}
```

解説（else ブロックは省略可）

```
if (条件式) {
    命令 1;
    命令 2;
} else {
    命令 3;
    命令 4;
}
```

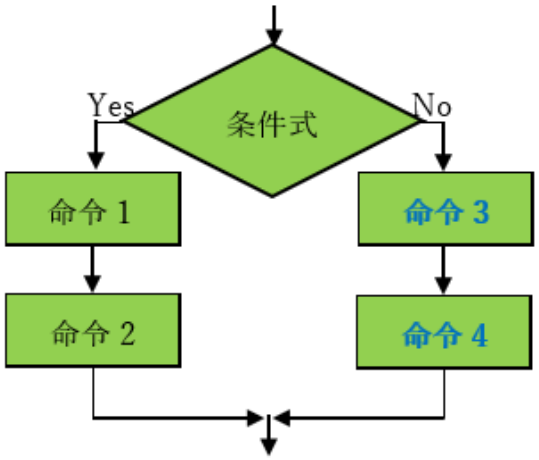


図 11　if 文のプログラム例と解説、フローチャート

基本的に、サンプルプログラムを 1 つ示したあとに、問題を用意し、1 問 1 答形式で理解を深めながら進めるように配慮した。（図 12）

問題） 2 つの整数を入力すると、大きい数を表示するプログラムをつくりなさい。
（利用変数は a，b，max の 3 つ）

```
C:¥c>.¥program.exe↵
a = 5↵
b = 8↵
大きい数は 8 です
```

図 12　if 文の問題と実行例

４. 終わりに

今回の講座では、環境構築から構文説明、サンプルプログラムの提示、問題と解答例を示すといった構成でオリジナルテキストを作成し、講座を進めた。やや、難しい部分もあったようだが、生徒は概ね理解をしながら講習に参加することができていたと思う。

著者の利益相反：開示すべき利益相反はない