情報基礎・第 5 回

2026 年 5 月 7 日

配布資料の内容

5.1 プログラミング言語とは

いろいろな用途にコンピュータが使われています。複雑な計算を行ったり、携帯電話を制御したり、ビデオゲームを楽しませてくれたりします。しかし、単にコンピュータという物 (ハードウエア) があっただけでは、それは何の役にも立ちません。コンピュータが役に立つのは、そのコンピュータがどのように働くべきかを指示したプログラム (ソフトウエア) が用意されているからです。コンピュータの行うべき仕事を事細く記述したものを「コンピュータプログラム」、あるいは単に「プログラム」と呼びます。このとき、どのようなプログラムを書けばどのようにコンピュータが働くかについての約束事が必要になります。プログラムの書き方とその解釈のされ方があらかじめ決っていなければ、コンピュータに意図した通りの働きをさせることはできません。その「約束事」を提供してくれるものを**プログラミング言語**と呼びます。日本語が、日本語を話す人の間での意思疎通を可能にしてくれるのと同様に、1 つのプログラミング言語は人間の意図をコンピュータに伝えることを可能にしてくれます。

現在広く利用されているコンピュータは、その中に「メモリ」と呼ばれる記憶装置を備えており、ここにいろいろな情報を保存することができるようになっています。見掛けの上では、メモリに蓄えられる情報は 0 と 1 の二種類の数だけからなる長い長い数列のように見えます。人間が扱う普通の文章や、数値などの情報は、適当な規則にしたがって、0 と 1 からなる数列として表現されてメモリ中に記憶されます。コンピュータ自身の働きを指定するためのプログラムもやはりこのメモリ中に置かれます。0 と 1 の数個から数十個の並びを 1 つの単位として、コンピュータが行うべき作業を指示します。

どのような並びでどのような作業を行うかについての約束事は、そのコンピュータによってあらかじめ決められています。コンピュータはこの約束事にしたがって順に指示されたとおりの作業を行っていきます。この指示の単位となる 0 と 1 の並びを**機械語命令**と呼びます。1 つ 1 つの機械語命令は、非常に単純な作業しか行わせることはできませんが、この単純な作業を何千、何万と組み合わせることにより、コンピュータに非常に複雑な作業を行わせることができます。機械語命令をたくさん組み合わせて、コンピュータの行うべき一連の作業を指定したものを**機械語プログラム**と呼びます。

メモ

コンピュータに一連の作業をさせるには、この機械語プログラムというものを用意することが必要ですが、1つ1つの機械語命令がどのような作業に対応するかについての約束事は一般に非常に複雑であり、また、1つ1つの機械語命令は非常に単純な作業しか行わせることができないため、ある程度のまとまった作業をコンピュータに行わせるために必要な機械語プログラムを人間が直接作るのは相当に困難です。そこで、もっと人間が理解しやすい書き方でコンピュータの行うべき仕事を記述し、それを機械語プログラムに機械的に変換することでコンピュータに仕事をさせるという方法がよく用いられます。この、より人間が理解しやすい書き方も、一定の約束事 (どう書けば、どう働くか) に基づいて書かれますが、ちょうど、人間の日常の言語に英語や日本語、中国語の違いがあるように、この「より人間が理解しやすい書き方」にも、その用途に応じていろいろなものがあります。機械語に対して、より人間が理解しやすいプログラミング言語を総称して、一般に「高級 (プログラミング) 言語」と呼びます。ここでの「高級」という語は「優れている」という意味ではありませんので誤解しないでください。

高級プログラミング言語はコンピュータが直接理解することはできませんから、何らかの方法でコンピュータの理解できる機械語へ変換してやらなければなりません。これには大きく分て2つの方法があります。その1つは、高級言語で書かれたプログラムを少しずつ読み取りながら、そこで指示されている作業を、機械語でコンピューターに指示し、実行させて行く方法です。あたかもコンピュータに対して人間の話している言葉をその場で通訳しているかのように働きますので、このような仕組みは**インタプリタ** (通訳) と呼ばれます。通常、このインタプリタ自身も1つのコンピュータプログラムとして実現されます。もう1つの方法は、コンピュータがそのプログラムの実行を始める前に、高級言語で書かれたプログラムをあらかじめ機械語にすべて翻訳しておく方法です。一旦翻訳が済んでしまえば、翻訳済みの機械語プログラムは何度でも利用することができます。この翻訳作業のことを**コンパイル**と呼び、翻訳を行ってくれる仕組みを**コンパイラ**と呼びます。また、コンパイルする前の高級言語で書かれたプログラムを**ソースプログラム**と呼び、コンパイルして得られた機械語プログラムのことを**オブジェクトプログラム**と呼びます。コンパイラ自身もやはり1つのコンピュータプログラムとして実現されます。

メモ

5.2 プログラミング言語 C

C は高級プログラミング言語の1つです。コンピュータの機械語を知らなくても、コンピュータに意図した働きをさせることができます。C は通常コンパイラによる方法で利用されます。非常に単純なプログラムを例として、プログラムの作成と、コンパイル、コンパイルして出来上がったオブジェクトプログラムの実行を行ってみましょう。

C プログラムの作成　コンピュータの記憶装置に格納されたプログラムやデータなどの個々のまとまりを「ファイル」と呼びます。ソースプログラムもオブジェクトプログラムもコンピュータの中では、1 つのファイルとして存在することになります。

C のソースプログラムは、アルファベットや数字、記号などの文字が並んでできあがったものですから、Microsoft Word などの文書作成ソフトで文書を編集するのと同じようにして作ることができます。ただし、文書作成ソフトの場合は、文字の大きさ、太さ、飾りなどを含めた文書の見掛けが重要ですが、C のソースプログラムとしては、単にそこにどんな文字がどの順に並んでいるかだけが重要になります。このような「文字の並び」としてだけの情報を持ったファイルは、一般に「(プレーン) テキストファイル」と呼ばれます。C のソースプログラムは単に、このテキストファイルの 1 つにしか過ぎません。

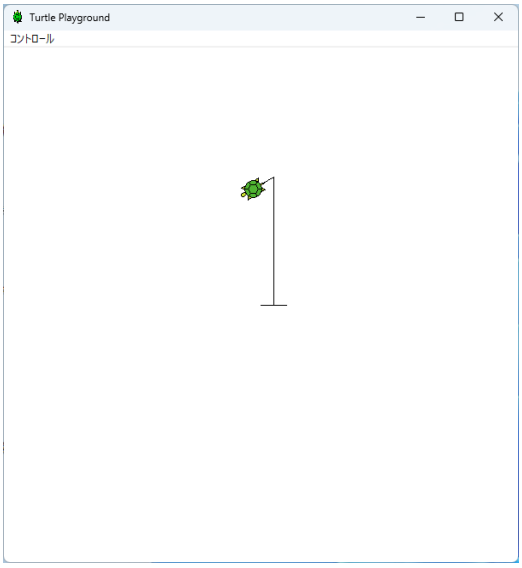
Windows や macOS などの PC (パソコン) 環境では、それぞれのファイルに名前を付けることができて、その名前でファイルを区別することができますが、C のソースプログラムが書いてあるファイル名は、通常、末尾を「.c」[1]とします。ここでは、myfirst.c という名前のファイルに、次のようなソースプログラムを書き込んでみましょう。

myfirst.c

```
#include <turtle.h>

int main()
{
    tForward(30);
    tBackward(15);
    tTurn(90);
    tForward(150);
    tTurn(120);
    tForward(28);
}
```

この C プログラムをコンパイルして、PC で実行すると、右ののようのなウィンドウが現れ、その中のカメが動いて、数字の 1 の形を描いてくれます。



Windows 環境でも macOS 環境でも使用できる「Visual Studio Code」や、Windows 環境の「サクラエディタ」[2]や「メモ帳」、macOS 環境 の「テキストエディット」といった**テキストエディタ**と総称されるソフトウェア (これもプログラム) を用いてテキストファイルを編集することができますので、これを用いてソースプログラムを書きます。C のソースプログラムとしては、どのような

[1] このように、ファイルの種別を区別するために、ファイル名の末尾につけられた「.」で始まる文字列を、Windows 環境では「拡張子」と呼びます。

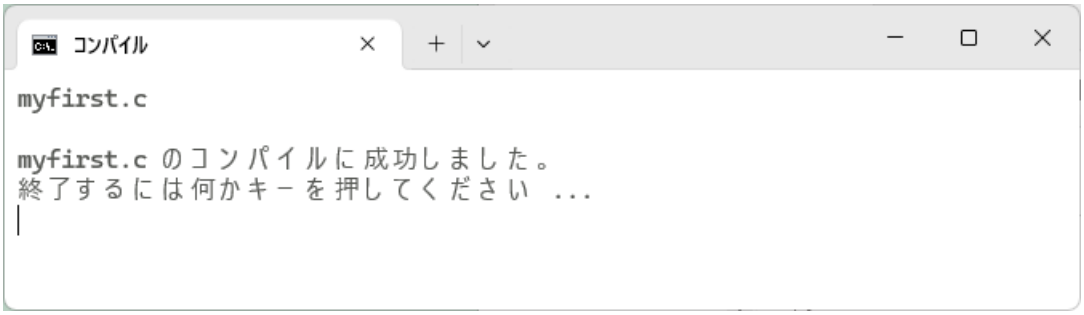
[2] Visual Studio Code やサクラエディタは無料のアプリケーションソフトウェアですが、自分でインストールする必要があります。

文字がどのように並んでいるかだけが意味を持ちますから、どんなテキストエディタを使って書いたとしても、できあがったテキストファイル中の文字の並びが同じである限り、そのプログラムの働きには影響はありません。

メモ

C プログラムのコンパイル　ソースプログラムをファイル (`myfirst.c`) に保存したら、そのソースプログラムをコンパイラを使ってコンパイルし、オブジェクトプログラムに変換します。コンパイラ自身も 1 つのオブジェクトプログラムですから、アプリケーションプログラムの一つとして、それを起動することになります。この科目では、コンパイラを簡単に使うために、デスクトップにショートカットを用意して、それを用いて C プログラムのコンパイルを行います。Windows PC の場合、このショートカットは、デスクトップ上に右のようなアイコンで表示されているはずです。

先ほどテキストエディタで作った `myfirst.c` という C プログラムをコンパイルするためには、この `myfirst.c` というファイルを、このアイコンへドラッグ & ドロップします。こうすると、ターミナルウィンドウが現れ、正常にコンパイルできた場合には次のように表示されるます。

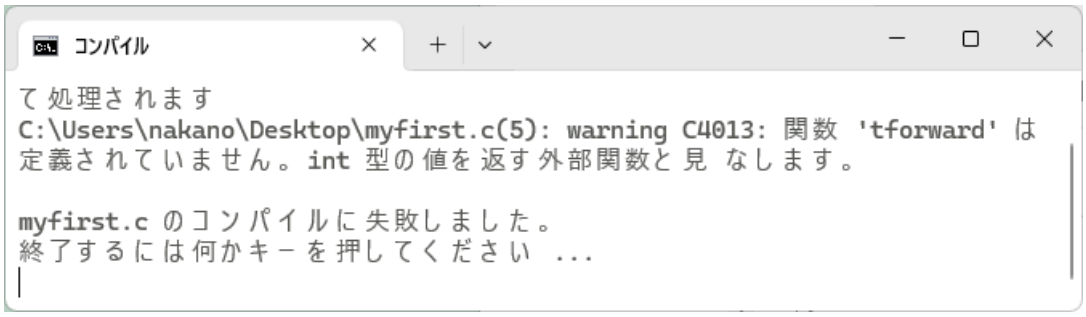


このターミナルウィンドウは、Enter キーを押せば消えてなくなります。正常にコンパイルできていれば、ソースファイル `myfirst.c` のあったフォルダ (ディレクトリ) に `myfirst.exe` という名前のオブジェクトプログラムが作成されます[3]。Windows 環境の設定次第で「`.c`」や「`.exe`」などの拡張子は表示されないので注意してください。コンパイルしてできたオブジェクトプログラム、右のようなアイコンで表示されます。

もし、ソースプログラムに何らかの間違いがあって、正常にコンパイルができなかった場合、その間違いの内容に関するエラーメッセージが表示されます。この場合、オブジェクトプログラムは作成

[3]macOS の場合は、myfirst.app という名前のディレクトリが作られ、そこにオブジェクトプログラムが置かれます。

されません。次は、ソースプログラム myfirst.c 中の「tForward(30);」の部分を「tforward();」と書き間違えてコンパイルに失敗した場合の例です[4]。



また「tForward(30);」の「;」を書き忘れると次のようになります。

いずれにせよ、正常にコンパイルができない場合は、テキストエディタで myfirst.c の内容を修正し、もう一度、コンパイルを行います。

メモ

オブジェクトプログラムの実行　コンパイルが成功したら、いよいよできあがったオブジェクトプログラムの実行です。オブジェクトプログラムは、そのアイコンをダブルクリックすることで、実行することができます。

ソースプログラムが正常にコンパイルできた場合でも、できあがったオブジェクトプログラムが期待した通りに動作しない場合もあります。このような場合もやはり、テキストエディタでソースプログラム myfirst.c の内容を修正し、もう一度コンパイルし直すことになります。一方、正常に動作するオブジェクトプログラムが完成してしまえば、そのオブジェクトプログラムは何度でも実行することができますので、実行の度にコンパイルしてやる必要はありません。

[4] これは Windows での例で、macOS の場合のエラーメッセージは異なります。

5.3 演習問題

1. 例として挙げた `myfirst.c` というプログラムをテキストエディタを使って作成し、コンパイル、実行してみなさい。

2. 次のような内容の C プログラムを作成、コンパイル、実行してみなさい。プログラムの内容を理解する必要はありません。ただし、ソースプログラムのファイル名は `fakelogo.c` としなさい。

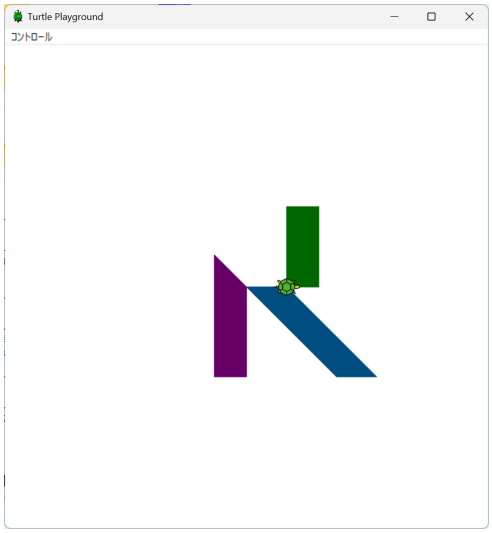
fakelogo.c

```
#include <turtle.h>

int main()
{
    int i;

    tSetColor(0.4, 0.0, 0.4);
    for (i = 0; i < 4; i++) {
        tTurn((3-i/2)*45);
        tForward(8*(7-(i/2)*(2+(i/3)*3)+(i%2)*12));
    }
    tFill();
    tTurn(-90);
    tSetColor(0.0, 0.4, 0.0);
    tForward(50);
    for (i = 0; i < 4; i++) {
        tForward(40+60*(i%2));
        tTurn(90);
    }
    tFill();
    tSetColor(0.0, 0.3, 0.5);
    for (i = 0; i < 4; i++) {
        tTurn(-45-(i%2)*90);
        tForward(158-(i%2)*108);
    }
    tFill();
}
```

このプログラムをコンパイルして実行すると、次のような図が描かれるはずです。



情報基礎・第 5 回・終わり