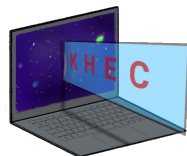
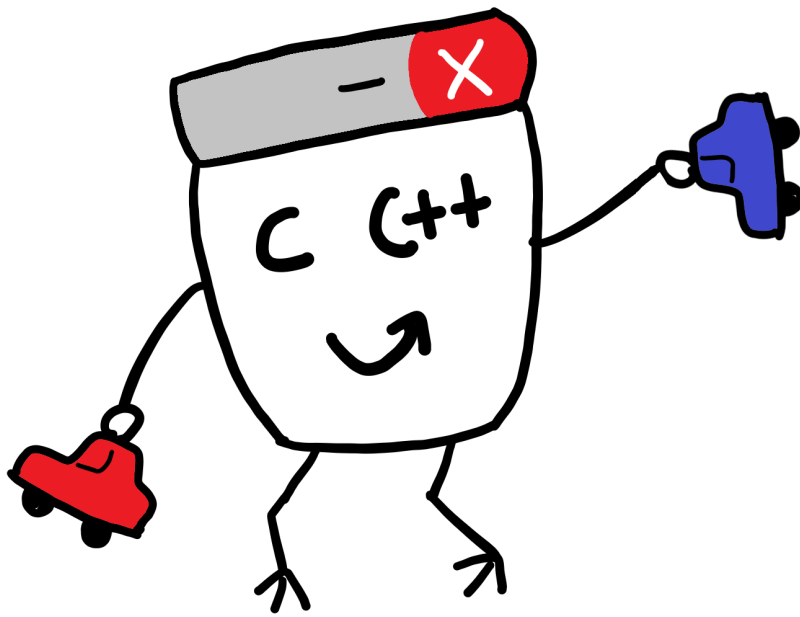


Cから学ぶC++



<目次>

A、準備 4

1、前書き 4

1-1、目的 4

1-2、数値的な目標 4

1-3、資料の特徴 4

1-4、C/C++とは 4

1-5、C/C++の構造 4

2、開発環境 4

2-1、環境 4

2-2、VisualStudio2019の用意 5

2-3、プログラムの作成 5

2-4、ソースコードの追加 6

2-5、ソースコードを記述し実行 7

2-6、ヘッダファイルの追加 7

2-7、プロジェクト管理 7

B、C言語(基本) 8

1、初めてのC言語 8

1-1、printf関数(基本) 8

1-2、printf関数(応用) 9

1-3、ヘッダファイル 10

1-4、関数 10

| | |
|------------|----|
| 1-5、記号 | 11 |
| 2、演算・変数 | 11 |
| 2-1、演算(基本) | 11 |
| 2-2、変数(基本) | 12 |
| 5、配列 | 13 |
| 5-1、文字列と配列 | 13 |
| 5-2、多重配列 | 14 |
| 6、関数 | 15 |
| 6-1、関数の作成 | 15 |

A、準備

1、前書き

1-1、目的

CやC++の学習の補助。

少なくともArduinoの初級編でSerial.print()を理解した人を対象としています。

1-2、数値的な目標

和田先輩のC++の知識の4割以上の理解。

1-3、資料の特徴

重要な部分は赤シートで隠せます。

専門用語は最小限です。

1-4、C/C++とは

Arduino言語のベースとなっている言語で、Arduino言語とは異なりパソコン単体で動く。

C++はCの上位互換。Cで使える記述の多くはC++で使える。

C/C++は古くからある言語だが、無駄の少ない記述ができるのでメモリの容量が小さいマイコン等に多用される。また、処理が速い。

1-5、C/C++の構造

ソースコード: 人が記述するプログラム

↓

コンパイラ: ソースコードをコンパイルし、コンピュータが理解出来るマシン語に翻訳します。

↓

マシン語: OSやCPUで動作する0と1の羅列で記述された命令。人は理解出来ません。

2、開発環境

2-1、環境

OS: Windows 10

統合開発環境: VisualStudio2019 Community Edition(導入方法は異なりますがVisual Studio Codeも可。)

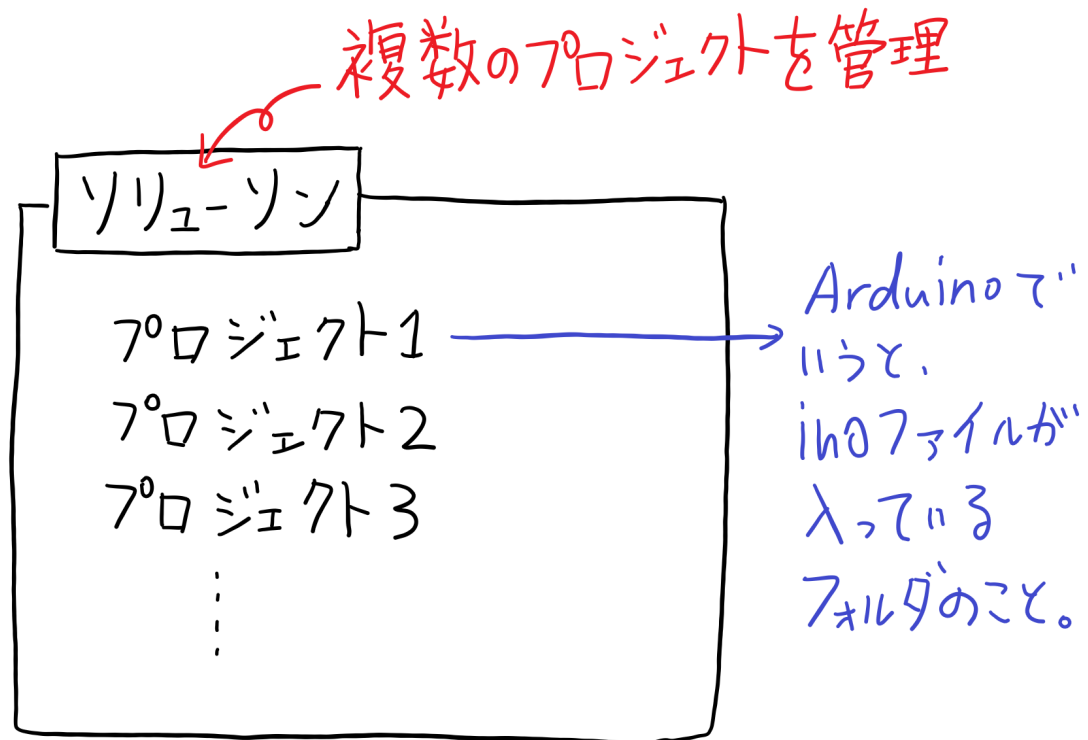
2-2、VisualStudio2019の用意

「VisualStudio2019 ダウンロード」と検索し、MicrosoftのWebサイトからCommunity Editionの「Visual Studio Installer」をインストールして下さい。

そして、「Visual Studio Installer」を実行し最初に表示される画面で「C++によるデスクトップ開発」にチェックを入れて「変更」をクリックし、ダウンロードとインストールを開始して下さい。（チェックを入れることでソフトとC/C++のコンパイラと一緒にインストールされます。）インストール終了後ライセンスの確認を行ったら完了です。

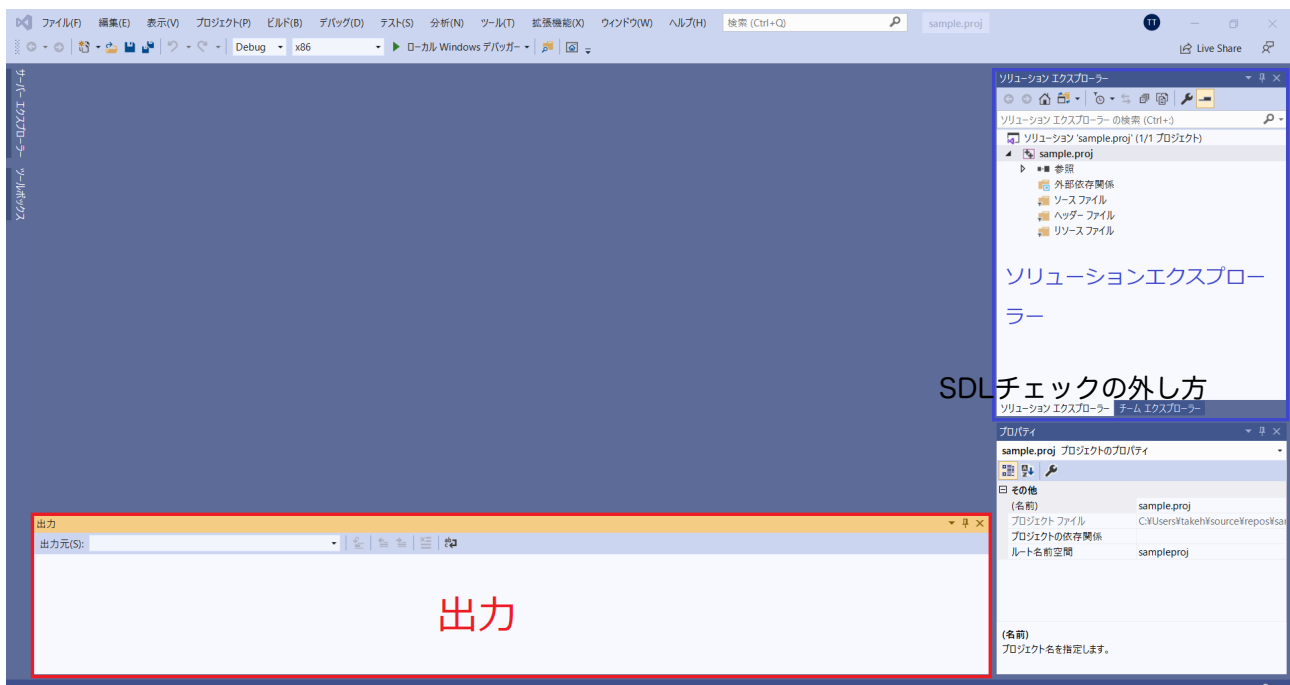
2-3、プログラムの作成

プロジェクトはプログラムを管理する単位で、最終的には「ビルド」という作業を通して最終的には1つのプログラムになる事を理解しておきましょう。また、複数のプロジェクトをソリューションという



それでは、実際にプログラムを作成します。まずはVisual Studio 2019を起動して下さい。そして「新しいプロジェクトを作成」をクリックし、「空のプロジェクト」をダブルクリックして新たなウィンドウを開いてください。ちなみに、「空のウィンドウ」の下に並んでいるのはWindowsのデスクトップアプリケーションなど、C++で特定のプログラムを制作する際の雛形です。Windowsのソフトウェアも作れるなんて、わくわくしますね。

すると、次の画面ではプロジェクト名とソリューション名を設定します。ここではプロジェクト名を「sample_proj」としましょう。最初、ソリューション名はプロジェクト名が自動的に反映されます。今回は1プロジェクトにつき1ソリューションとするのでこのままで「作成」をクリックして下さい。やっとプログラムを記述する画面が現れます。



下図において右上の「ソリューションエクスプローラー」ではプロジェクトやソースコードを管理します。パソコンのエクスプローラーみたいなものです。また、下部の「出力」はブ操作結果やエラーを表示します。Arduino IDEにも似た機能ですね。

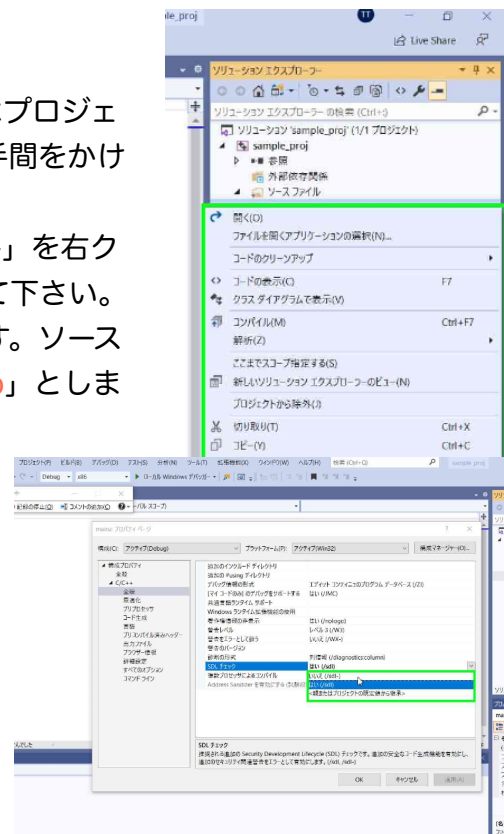
2-4、ソースコードの追加

0-8でプロジェクトを作成しました。しかし、プログラムはプロジェクトの中に書くものなので、プログラム作成というもう一手間をかけます。

まず、ソリューションエクスプローラーの「ソースファイル」を右クリックし、「追加」を選択、「新しい項目」をクリックして下さい。するとファイルの名前を設定できるウィンドウが出現します。ソースファイルの拡張子はCの場合は「.c」、C++の場合は「.cpp」とします。ここでは、「main.c」と入力して追加しましょう。それが終わるとmain.cのソースコード(注1)を編集する画面になります。

注1)「ソースコード」は人間が分かるプログラムの事を指します。

ソースコードを作成したら、SDLチェックを外します。ソリューションエクスプローラー上のmain.cを右クリックし、「プロパティ」をクリックします。するとウィンドウが開きますので、「C/C++」の「全般」にある「SDLチェック」という項目の状態を「はい」から「いいえ」に変更してください。SDLチェックを外すと、正常にプログラムを実行出来るようになります。



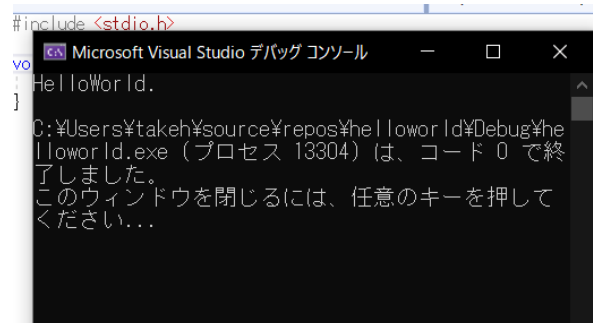
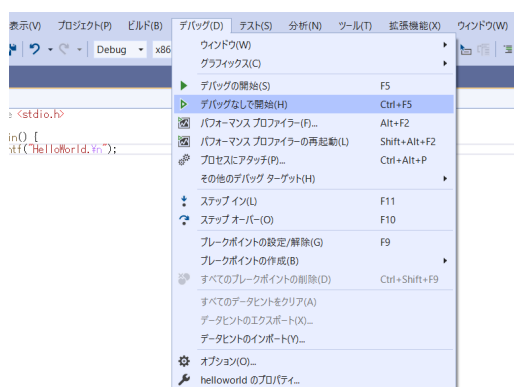
2-5、ソースコードを記述し実行

ソースコードとして以下の通り入力してください。

```
#include <stdio.h>
```

```
void main() {  
    printf("HelloWorld.\n");  
}
```

そして、画面上部のデバッグタグを選択し、「デバッグなしで実行」をクリックしてください。すると、プログラムが実行され、コマンドプロンプト上にて「HelloWorld.」と表示されます。これがプログラムを実行するまでの一通りの流れです。



豆知識) Ctrl+ホイール操作でソースコードの文字サイズを変更できます。

2-6、ヘッダファイルの追加

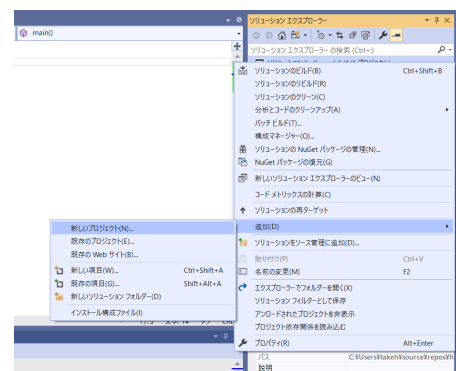
現段階ではソースファイルのみを使用していますが、そのうち「ヘッダファイル」というものを追加して記述する事が有ります。これもソースファイルと同様にソリューションエクスプローラー内の「ヘッダファイル」を右クリックし、「追加」を選択し、「新しい項目」をクリックしてください。そして、新しく出現したウィンドウで拡張子を「.h」としてファイルを作成します。詳しくは後ほど。

2-7、プロジェクト管理

基本的にこのテキストでは1ソリューション1プロジェクトですが、1つのソリューションに複数のプロジェクトを作成したい時もあります。0-8でソリューション「sample_proj」内にプロジェクト「sample_proj」を作成しましたね。ここでは、同じソリューション内に2つ目のプロジェクト

「sample_proj2」を作成します。

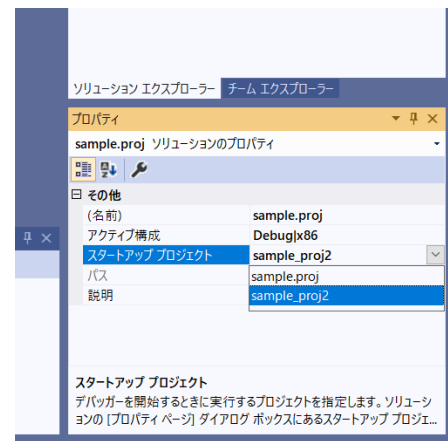
まず、ソリューションエクスプローラー内のソリューション名を右クリックし、「追加」を選択し、「新しいプロジェク



トを追加」をクリックします。すると、見覚えのあるプロジェクト追加を行うウィンドウが開くので、そこで新しいプロジェクト「sample_proj2」を追加してください。

これで2つ目のプロジェクトを追加しました。しかし。この状態では、2つ目のプロジェクトにソースファイルを追加し「デバックなし実行」をクリックしても、それが実行されることは有りません。それは、実行されるのは「スタートアッププロジェクト」に設定したプロジェクトのみだからです。

スタートアッププロジェクトを変更するのは、ソリューション名をクリックで選択した状態のソリューションエクスプローラーの下「スタートアッププロジェクト」という項目で変更できます。



B、C言語(基本)

1、初めてのC言語

1-1、printf関数(基本)

0-10で「HelloWorld.」と出力しましたね。以下がそのプログラムです。

```
#include <stdio.h>
```

```
void main() {  
    printf("HelloWorld.\n");  
}
```

まず、「**printf**("");」で""内の文字が出力される事を覚えてください。尚、文字列は""で、1文字は"で囲み、それらと他の部分を区別する必要があります。日本語も出力できます。

基本的に1文のプログラムの終わりには「**;**」をつける必要が有ります。

さらに、「**\n**」でその部分が改行されます。「**\n**」が出力されたり空白ができたりすることはありません。

例えば0-10で作成したプログラムを以下の様に変更し実行してください。

ただし、0-12でスタートアッププロジェクトを変更している事に留意して下さい。

```
#include <stdio.h>
```

```
void main() {  
    printf("HelloWorld.");  
    printf("こんにちは世界。");  
}
```



```
}
```

すると、「HelloWorldこんにちは世界。」と出力されます。これは「\n」を入れていないからです。

また、以下の様に文中で改行を入れることもできます。

```
#include <stdio.h>
```

```
void main() {  
    printf("Hello\nWorld.");  
}
```

すると以下の様に出力されます。

```
////////////////////////////////////  
Hello  
World.  
////////////////////////////////////
```

プログラム内で他に分からない点は幾つかあると思います。ただ、現段階では「main(){}」の波カッコ内が実行されるんだな、と理解していただけると幸いです。

Coffee Break ~Arduinoでは?~

\nはASCII(アスキー)コードという文字をデジタルに扱う上での決まりで定められている「改行コード」です。Arduino言語ではprintln();とすることで文末で改行出来るので余り使用頻度は高くありませんが

```
String s = "Hello\nWorld";  
Serial.print(s);
```

という様に改行コードを用いて改行する事が出来ます。

1-2、printf関数(応用)

ここでは、「他の所から文字や数字を取ってきて、出力する文字列に組み込む。」ことをします。以下の様にプログラムを入力してください。

```
#include <stdio.h>
```

```
void main() {  
    printf("入学おめでとう。私たちは%sです。 \n部員はは%d人です。 \n", "電工研", 15);  
    printf("活動場所は、 %c組の教室です。 \n", 'K');  
    printf("%f + %f = %f\n", 2.23, 2.64, 2.23 + 2.64);  
}
```

すると以下の様に出力されます。

```
////////////////////////////////////
```

入学おめでとう。私たちは電工研です。

部員はは15人です。

活動場所は、K組の教室です。

2.230000 + 2.640000 = 4.870000

```
////////////////////////////////////
```

プログラムにの「%~」の部分が別の文字に置き換わっているのが分かります。そして、置き換える為に用意した文字は本文の後ろにカンマによって区切られてセットされています。

例えば「入学おめでとう~」の文章では、1つめの「%~」に「"電工研"」が、2つめの「%~」に「15」が順番に対応するようにセットされています。他の行でも同様です。

ただし、「%~」の「~」の部分の英字が違います。これはそれぞれ置き換えるデータの種類のによって以下の様に使い分けます。

- ・%s: 文字列("電工研"など)
- ・%d: 整数(文字ではなく整数なので置き換える数字は""で囲みません。)
- ・%c: 1文字の英数字('k'など)
- ・%f: ある決まりによって表示される小数(2.14など)
- ・%lf: ある決まりによって表示される小数で%fより長い桁を表示可能。

加えて、足し算をしている行を見てください。ここでは「2.23 + 2.64」を3つめの%fに入れています。出力は「4.870000」です。これは、計算式の場合はその計算結果が出力されるからです。計算式をそのまま表示させるには""で囲んで文字列であると認識させなければなりません。尚、0が後ろについているのは「ある決まり」によるものです。今は気にしないで構いません。

Coffee Break ~なぜ「HelloWorld!」?~

多くのプログラム言語の入門では、「HelloWorld!」と出力させられます。なぜでしょう。

それは、世界初のC言語の教科書「プログラミング言語C」のサンプルで使用されていたからです。1978年刊行ということだととても古い本です。ちょっと読んでみたい。

1-3、ヘッダファイル

今まで記述したプログラムの先頭には「#include <stdio.h>」という記述があります。「.h」が拡張子のファイルはヘッダファイルと呼ばれ、「#include <>」で<>内のヘッダファイルを読み込むことが出来ます。

1-4、関数

今まで実行したいプログラムは「void main(){ }」の{ }内に書いてきました。C言語ではこれを**メイン関数**と呼び、この中にプログラムを書くことによって処理が行われます。

C言語に元々備わっている関数はあまたとあり、「printf();」も()内に出力したい内容を入れるとそれが表示される関数の一種です。また、自分で独自の機能を持った関数は付け足すこともできます。

1-5、記号

・セミコロン(;)

複数行に渡る処理を記述したい場合、各処理の末尾に必要です。

・エスケープエンス


「\」で始まる文字のこと。(円のマークとして表示される事が多い。)改行記号「\n」等があります。以下にその例を示します。

- ・\a: 警告音
- ・\b: バックスペース
- ・\n: 改行
- ・\t: タブ
- ・\\: 文字としての\
- ・\?: 文字としての?マーク
- ・\: ダブルクォーテーション
- ・\": シングルクォーテーション
- ・\0: ヌル文字

・書式指定

1-2でprintf関数内で使用する値を別の場所にセットする方法を紹介しました。これを図で表すと以下のようになります。

```
printf("入学おめでとう。私たちは%sです。¥n部員はは%d人です。¥n", "電工研", 15);
```



1-2で説明した通り、%の後の文字で文字の種類(書式)を設定しています。これを書式指定といい、これを正しき行わないとエラーになります。

・文字と文字列

文字はシングルクォーテーションで囲み、文字列はダブルクォーテーションで囲みます。

例) 'k'、"電工研"

2、演算・変数

2-1、演算(基本)

これ以降新しいプログラムを作成する際は「sol[項目の番号]」という名称でソリューションを都度生成する事をおすすめします。そして1つの項目内でもいくつかのプロジェクトを作成するのでその名称は「pro[項目内の通し番号]」としましょう。ここでは、「sol1-6」というソリューションの中に「pro1」、「pro2」とプロジェクトを生成していくことになりますね。

では演算の練習です。以下のプログラムを実行しましょう。

```
-----  
#include <stdio.h>
```

```
/*
```

```
  演算子を用いた計算のプログラム
```

```

*/

void main()
{
    // 各種演算
    printf("%d + %d = %d\n", 5, 2, 5 + 2);          /* 足し算 */
    printf("%d - %d = %d\n", 5, 2, 5 - 2);          /* 引き算 */
    printf("%d * %d = %d\n", 5, 2, 5 * 2);          /* 掛け算 */
    printf("%d / %d = %d 余り %d \n", 5, 2, 5 / 2, 5 % 2); /* 割り算 */
}

```

結果

```

////////////////////////////////////

```

5 + 2 = 7

5 - 2 = 3

5 * 2 = 10

5 / 2 = 2 余り 1

```

////////////////////////////////////

```

C言語での四則演算では、掛け算を「*」で、割り算を「/」で、2つの数を割った余りを出す計算を「%」で行います。

そして、ここではコメントを追加しています。コメントは1行のみの場合は「//」のあとに続け(行コメント)、複数行に渡ってコメントする場合は「/*」と「*/」でコメントを挟みます(ブロックコメント)。

2-2、変数(基本)

先程の以下のソースコードを書きましょう。

```

#include <stdio.h>

/*
    変数を用いた計算
*/

void main()
{
    /* 使用する変数の定義 */
    int a; // 変数の宣言
    int b = 3; // 初期化と代入を同時に行う。
    int add, sub; // 複数の変数を同時に宣言
    double avg; // int以外の変数を宣言
    a = 6; // 代入 (最初に値を入れるので、"初期化"と言う。
    add = a + b; // a,bの和を求める。
    sub = a - b; // a,bの差を求める。
}

```

```

    avg = (a + b) / 2.0;    // a,bの平均値を求める。
    printf("%d + %d = %d\n", a, b, add);
    printf("%d - %d = %d\n", a, b, sub);
    printf("%dと%dの平均値 : %f\n", a, b, avg);
}

```

5、配列

5-1、文字列と配列

以下のソースコードを書きましょう。

```

#include <stdio.h>

void main(){
    char s1[4] = { 'a','b','c','\0' }; // 文字列"abc"
    char s2[] = "HelloWorld.";        // 文字列"HelloWorld."
    char s3[10];                      // 最大10文字まで入る文字列
    // 文字列の入力
    printf("文字列を入力してください。:");
    scanf("%s",s3); // 文字列の入力
    printf("s1 = %s\n",s1);
    printf("s2 = %s\n",s2);
    printf("s3 = %s\n",s3);
}

```

```

////////////////////////////////////

```

文字列を入力してください。:khec

s1 = abc

s2 = HelloWorld.

s3 = khec

```

////////////////////////////////////

```

ここではchar型の配列を3つ生成しています。C言語では文字列をそのまま扱うことが出来ないの
で1文字ずつ並べた配列として扱います。尚、文字はシングルクォーテーションで囲む必要が有る
ことに注意して下さい。また、文字の配列の最後には**ヌル文字**と呼ばれる「\0」を入れる必要が
有ります。これは文字列の終端を表すもので、これがないと文字列が保存されている所より後ろ
のメモリまで読み込んでしまいます。

s2も文字の配列を生成しているのですが、文字列の配列に限りこのように省略した書き方が可能
になります。この場合ダブルクォーテーションで囲む必要が有ります。これには「\0」がついて
いませんが、自動で追加されるので問題ありません。勿論「\0」をつけても問題ありません。

試しにprintf("文字列を入力してください。:");の行にブレイクポイントを追加してデバッグしてみてください。配列の終端に「\0」が追加されているのが分かります。ちなみにs3の中身が「7」になっているのは初期化されていない(まだ何も代入されていない)からです。

また、scanfについて今までと使い方が違います。今までは「scanf("書式指定",&変数名);」としてきましたが、文字列などの配列は変数名につける&が不要です。これはポインタ等の関係なので後々説明します。

5-2、多重配列

以下のソースコードを書きましょう。

```
-----  
#include <stdio.h>
```

```
void main() {  
    int a[3][4];  
    int m, n;  
    // 二次元配列に値を代入  
    for (m = 0; m < 3; m++) {  
        for (n = 0; n < 4; n++) {  
            a[m][n] = m + n;  
        }  
    }  
    // 成分の表示  
    for (m = 0; m < 3; m++) {  
        for (n = 0; n < 4; n++) {  
            printf("%d ", a[m][n]);  
        }  
        printf("\n");  
    }  
}
```

```
-----  
////////////////////////////////////
```

```
0 1 2 3
```

```
1 2 3 4
```

```
2 3 4 5
```

```
////////////////////////////////////
```

今まで扱ってきた配列はデータが1列に並んだものでした。しかし、こちらは二次元配列です。要するにエクセルのシートの様になっていると理解して下さい。「int a[3][4];」で3行 × 4列の表を作っています。そして、その1つ1つのセルにfor文でデータを入れて、後ろのfor文で出力しています。たとえば2行目の3列目を指定する場合はa[1][2]とします。

| | |
|--|---|
| | n |
|--|---|

| | | | | |
|---|--|--|--|--|
| m | | | | |
| | | | | |
| | | | | |

三次配列などもできますが、次数が増えると扱いづらくなります。

6、関数

6-1、関数の作成

今までソースコードはmain関数の中に書いてきました。しかし、ソースコードの量が増えると、同じ処理を何度も書いたり、見づらくなったりしてきます。そこで、特定の処理のまとまりを1つの「部品」にしてしまう事でより快適にプログラミングが出来ます。

以下のソースコードを書きましょう。

```
-----
#include <stdio.h>

// 平均値を求める関数の定義
double avg(double l, double m) {
    // 引数l,mの平均値を求め、rに代入する。
    double r = (l + m) / 2.0;
    return r;
}

void main() {
    double d1, d2, d3;
    double a = 1.2, b = 3.4, c = 2.7;
    // 同じ計算が3回(関数を呼び出して計算)
    d1 = avg(a, b);
    d2 = avg(4.1, 5.7);
    d3 = avg(c, 2.8);
    printf("d1 = %f,d2 = %f,d3 = %f\n", d1, d2, d3);
```

```
}
```

```
-----  
////////////////////////////////////  
d1 = 2.300000,d2 = 4.900000,d3 = 2.750000  
////////////////////////////////////
```

このプログラムでは平均値を求めるavg()という関数が使われていますが、これはC言語に元々用意されている関数ではなく、プログラム作成者が定義した関数です。

関数には2つ必要なものがあります。

- ・ **引数**: 関数に与える値
- ・ **戻り値**: (引数を与えられている場合はそれを利用して)処理した結果

数学でも「 $y = 2x^2$ 」など関数を作りますよね。この場合はxが引数でyが戻り値です。

そして、先程のプログラムでは、「平均値を求める関数の定義」というコメントの下の部分関数の中身(定義)です。数式で言えば「 $2x^2$ 」の部分ですね。

ここでは()内でカンマで区切られた2つの変数(l, m)が引数となっています。そして、2つともdouble型で宣言されています。そしてその引数である変数は関数内の処理で使われています。但し、()内で宣言された変数は他の関数内やグローバルで使うことは出来ません。

そして、avg関数内ではrというdouble型の変数を宣言しています。ここに平均値(答え)を記録し、return r; の部分で戻しています。「return 値」で値を戻すことが出来ます。