

C/C++プログラミング

第2回

C言語のプログラミングと実行/変数

津村幸治・邵肖偉

画面への出力を行うプログラム: C++

VSCoideでのソースコードはデフォルトでは**UTF-8**で保存される。
Windows の場合、**UTF-8**で保存されるプログラム内の日本語をVSCoide
の「ターミナル」、つまり Windowsのコマンドプロンプトで文字化け
させずに表示させるために次の前準備が必要

「コントロールパネル」→「時計と地域」→「地域」→「管理」
→「システムロケールの変更」→「ワールドワイド言語サポートで...」
のチェックボックスにチェック→「OK」→「今すぐ再起動」

以上で Windows の再起動後は、コマンドプロンプトで **UTF-8** の日本語
を文字化けせずに表示する。

画面への出力を行うプログラム: C++

```
-----  
// 画面への出力を行うプログラム  
#include <iostream>  
using namespace std;  
  
int main()  
{  
    cout << "初めての C++プログラム。\\n";  
    cout << "画面に出力しています。\\n";  
    return 0;  
}
```

VSCode 上で以上のC++のプログラムを作成しコンパイル&実行させる.

初めての C++プログラム。
画面に出力しています。

と表示されることを確認する.

Visual Studio Code interface showing a C++ program and its execution.

EXPLORER

- CPP
 - test1.c
 - test1.exe
 - test2-1.cpp
 - test2-1.exe
 - test2.cpp
 - test2.exe
- OUTLINE
- TIMELINE

test2-1.cpp

```
1  #include <iostream>
2  using namespace std;
3
4  int main()
5  {
6      cout << "初めてのC++プログラム。 \n";
7      cout << "画面に出力しています。 \n";
8      return 0;
9  }
10
```

TERMINAL

```
PS C:\Users\sxwin\Documents\Lecture\cpp> cd "c:\Users\sxwin\Documents\Lecture\cpp\" ; if ($?) { g++ test2-1.cpp -o test2-1 } ; if ($?) { .\test2-1 }
● 初めてのC++プログラム。
  画面に出力しています。
○ PS C:\Users\sxwin\Documents\Lecture\cpp>
```

Ln 10, Col 1 Tab Size: 4 UTF-8 CRLF {} C++ Go Live Win32

画面への出力を行うプログラム: C++

ソースコードの意味

「// 画面への出力を行うプログラム」

コメント文, `"/"`がコメントの始まりを示し, コメントの最後は改行まで. `"/* ... */` もコメント文.

「`#include <iostream>`」

`<iostream>` の中身をここへ挿入する (インクルードする) という意味. `<iostream>` はヘッダの一つであり, 標準入出力を用いる際に必要な情報が記述されている. `input/output stream` の意味.

「`using namespace std;`」

この行以降で `std` という名前空間を使うという意味. `std::cout`, `std::endl` などは `iostream` を `include` することで用いることができる. さらに `using namespace std` を実行することで, `std::cout`, `std::endl` は `cout`, `endl` と省略表記できる.

画面への出力を行うプログラム: C++

「`int main()`」

このプログラムのメインの関数という意味. 「関数」は何かしらの動作・処理を行うプログラムの意味. ここから後の中括弧で囲まれた中に, このプログラムの処理の中身を記述する.

「`main`」はメイン関数の名称で, この名前以外は使えない.

「`int`」はこの関数の返値, つまり実行した最後に`main`関数から帰ってくる値の型を表し, "`int`"は整数型を表す. このプログラムの場合 `0` が返ってくる.

「`main()`」の小括弧 "`()`" の中にはこの関数に渡される変数などを記述するが, この例の場合は渡される引数がないことを意味する.

画面への出力を行うプログラム: C++

「`cout << “初めての C++プログラム。\\n”;`」

「`cout`」は「`std::cout`」の省略形。「標準出力」(console output)を意味し、通常はプログラムを実行するコンピュータに接続されたモニター。

「`<<`」は「挿入演算子」と呼ばれるもので、`cout`に`<<`の右側に示された文字列や値を渡すことを意味する。この場合、「初めての C++プログラム。\\n」という「文字列リテラル」が`cout`つまりモニター画面に表示されることを意味する。

一つの文字列リテラルの途中で改行は禁止。

文字列の最後の「\\n」は改行を意味する記号。

最後のセミコロン「`;`」は、ここで一つの処理の終わりを表す。

画面への出力を行うプログラム: C++

「**return 0;**」

main関数の処理の最後に「0」をコンピュータの**OS**に返す. ここではプログラムが正常に処理が完了したかどうかを確認するために用いる.

画面への出力を行うプログラム: C

// 画面への出力を行うプログラム

#include <stdio.h> // 標準入出力用ヘッダをインクルード

int main()

```
{  
    printf("初めての C プログラム。 \n"); // printf関数で文字列を出力  
    printf("画面に出力しています。 \n"); // printf関数で文字列を出力  
    return 0; // 正常終了を示す  
}
```

VSCode 上で以上のCのプログラムを作成しコンパイル&実行させる。
モニターに

初めての C プログラム。
画面に出力しています。

と表示されることを確認する。

The image shows the Visual Studio Code interface with a C program being edited and executed. The Explorer sidebar on the left shows a project structure with files `test1.c`, `test1.exe`, `test2-1.cpp`, `test2-1.exe`, `test2-2.c` (selected), `test2-2.exe`, `test2.cpp`, and `test2.exe`. The main editor window displays the code for `test2-2.c`:

```
C test2-2.c > ...
1  #include <stdio.h> // 標準入出力用ヘッダをインクルード
2
3  int main()
4  {
5      printf("初めてのCプログラム。\\n"); // printf関数で文字列を出力
6      printf("画面に出力しています。\\n"); // printf関数で文字列を出力
7      return 0; // 正常終了を示す
8  }
9
```

Below the code editor, the TERMINAL panel shows the command prompt output:

```
PS C:\Users\sxwin\Documents\Lecture\cpp> cd "c:\Users\sxwin\Documents\Lecture\cpp\" ; if
($?) { gcc test2-2.c -o test2-2 } ; if ($?) { .\test2-2 }
● 初めてのCプログラム。
  画面に出力しています。
○ PS C:\Users\sxwin\Documents\Lecture\cpp> |
```

The status bar at the bottom indicates the cursor is at line 9, column 1, with 4 spaces, UTF-8 encoding, CRLF line endings, and the C language mode.

画面への出力を行うプログラム: C

ソースコードの意味

「`#include <stdio.h>`」

`<stdio.h>` の中身をここへ挿入する（インクルードする）という意味。
`<stdio.h>` は C++ の `<iostream>` と同等のヘッダで、標準入出力を用いる際にインクルードする。standard input/output の意味。

C++ に存在した「`using namespace std;`」は C のソースでは無い。C では名前空間という仕組みが無いから。

「`printf(“初めての C プログラム。\\n”);`」

`printf` は関数の名前であり、標準出力（モニター）に小括弧内の文字列を表示する、という関数。

変数：C++

int型変数の宣言

// 二つの変数xとyの合計と平均を表示

```
#include <iostream>
```

```
using namespace std;
```

```
int main()
```

```
{
```

```
    int x;        // xはint型の変数  
    int y;        // yはint型の変数
```

```
    x = 61;        // xに61を代入  
    y = 19;        // yに19を代入
```

```
    cout << "xの値は" << x << "です。 \n";           // xの値を表示  
    cout << "yの値は" << y << "です。 \n";           // yの値を表示  
    cout << "合計は" << x + y << "です。 \n";         // xとyの合計を表示  
    cout << "平均は" << (x + y) / 2 << "です。 \n";    // xとyの平均を表示
```

```
}
```

```
xの値は61です。  
yの値は19です。  
合計は80です。  
平均は40です。
```

変数：C++

int型変数の宣言: ソースコードの意味

「**int x;**」

「型」が整数型**int**の変数**x**を確保するよう「宣言」という意味。
「変数」は数値や文字列の格納場所。実際にコンピュータのメモリ上に、**32ビット/64ビット**のコンピュータであれば**4バイト**（**1バイト=8ビット**，**4バイト=32ビット**）の長さのビット列が確保される。

「**64ビットのコンピュータ**」とは**ALU**で一度に処理できるビット列のビット長が**64ビット=8バイト**ということ。レジスタのビット長でもある。

「**int**」は変数の型を表す。ここでは「整数型」を表し、整数を扱う格納場所を意味する。

変数：C++

int型変数の宣言: ソースコードの意味

「**x = 61;**」

整数型変数xに数値「61」を代入するという意味. 実際にコンピュータのメモリ上の格納場所に61の2進数表示されたビット列111101が格納される.

「**cout << "xの値は" << x << "です。 \n";**」

文字列「xの値は」と変数xの値(=61)と文字列「です。 \n」をモニターに表示する. 「<< x+y」とするとxとyに格納された値の和を表示する.

変数：C

int型変数の宣言

// 二つの変数xとyの合計と平均を表示

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

```
int main() {
```

```
    int x; // xはint型の変数
```

```
    int y; // yはint型の変数
```

```
    x = 61; // xに61を代入
```

```
    y = 19; // yに19を代入
```

```
    printf("xの値は%dです。 \n", x); // xの値を表示
```

```
    printf("yの値は%dです。 \n", y); // yの値を表示
```

```
    printf("合計は%dです。 \n", x + y); // xとyの合計を表示
```

```
    double average = (double)(x + y) / 2.0; // double型にキャストして計算
```

```
    printf("平均は%.2fです。 \n", average); // 平均を表示
```

```
    return 0;
```

```
}
```

xの値は61です。
yの値は19です。
合計は80です。
平均は40.00です。

キーボードからの入力：C++

キーボードからの入力・演算子・オペランド・2項演算子

// 二つの整数値を読み込んで加減乗除した値を表示

```
#include <iostream>
```

```
using namespace std;
```

```
int main()
```

```
{
```

```
    int x;        // 加減乗除する値
```

```
    int y;        // 加減乗除する値
```

```
    cout << "xとyを加減乗除します。\\n";
```

```
    cout << "xの値：";        // xの値の入力を促す
```

```
    cin >> x;                // xに整数値を読み込む
```

```
    cout << "yの値：";        // yの値の入力を促す
```

```
    cin >> y;                // yに整数値を読み込む
```

```
    cout << "x + yは" << x + y << "です。\\n";    // x + yの値を表示
```

```
    cout << "x - yは" << x - y << "です。\\n";    // x - yの値を表示
```

```
    cout << "x * yは" << x * y << "です。\\n";    // x * yの値を表示
```

```
    cout << "x / yは" << x / y << "です。\\n";    // x / yの値を表示 (商)
```

```
    cout << "x % yは" << x % y << "です。\\n";    // x % yの値を表示 (剰余)
```

```
}
```

xとyを加減乗除します。

xの値：10

yの値：20

x + yは30です。

x - yは-10です。

x * yは200です。

x / yは0です。

x % yは10です。

キーボードからの入力：C++

キーボードからの入力・演算子・オペランド・2項演算子
ソースコードの意味

「`cin >> x;`」

「`cin`」は「`std::cin`」の省略形。「標準入力」(console input)を意味し、通常はプログラムを実行するコンピュータに接続されたキーボードからのキー入力。「`cin >> x;`」はキー入力による数字を変数`x`に格納.

キーボードからの入力：C

キーボードからの入力・演算子・オペランド・2項演算子

#include <stdio.h> // 標準入出力ライブラリをインクルード

```
int main() {  
    int x;    // 加減乗除する値  
    int y;    // 加減乗除する値  
  
    printf("xとyを加減乗除します。\\n");    // printfで文字列を表示  
  
    printf("xの値：");    // xの値の入力を促す  
    scanf("%d", &x);    // scanfで整数値を読み込み、xのメモリアドレスを渡す  
  
    printf("yの値：");    // yの値の入力を促す  
    scanf("%d", &y);    // scanfで整数値を読み込み、yのメモリアドレスを渡す  
  
    printf("x + yは%dです。\\n", x + y);    // 整数値をフォーマットして表示  
    printf("x - yは%dです。\\n", x - y);    // 整数値をフォーマットして表示  
    printf("x * yは%dです。\\n", x * y);    // 整数値をフォーマットして表示  
    printf("x / yは%dです。\\n", x / y);    // 整数値をフォーマットして表示 (商)  
    printf("x %% yは%dです。\\n", x % y);    // 整数値をフォーマットして表示 (剰余)  
  
    return 0; // プログラムの終了  
}
```

```
xとyを加減乗除します。  
xの値：30  
yの値：10  
x + yは40です。  
x - yは20です。  
x * yは300です。  
x / yは3です。  
x % yは0です。
```

キーボードからの入力：C++

浮動小数点 float型, double 型の変数の宣言

// 二つの実数値を読み込んで加減乗除した値を表示

```
#include <iostream>
```

```
using namespace std;
```

```
int main()
```

```
{
```

```
    float x;        // float型の変数  
    double y;       // double型の変数  
    int z;          // int型の変数
```

```
    cout << "xとyを加減乗除します。\\n";
```

```
    cout << "xの値：";        // xの値の入力を促す  
    cin >> x;                 // xに実数値を読み込む
```

```
    cout << "yの値：";        // yの値の入力を促す  
    cin >> y;                 // yに実数値を読み込む
```

```
    cout << "zの値：";        // zの値の入力を促す  
    cin >> z;                 // zに実数値を読み込む
```

```
    cout << "x / yは" << x / y << "です。\\n";    // x / yの値を表示  
    cout << "z / yは" << z / y << "です。\\n";    // z / yの値を表示
```

```
}
```

```
xとyを加減乗除します。  
xの値：10  
yの値：20  
zの値：5  
x / yは0.5です。  
z / yは0.25です。
```

キーボードからの入力：C++

浮動小数点 **float**型, **double** 型の変数の宣言

ソースコードの意味

「**float x;**」

単精度浮動小数点型（小数）の変数 **x** の宣言. 4バイト=32bit の長さの変数を確保するという意味.

「**double y;**」

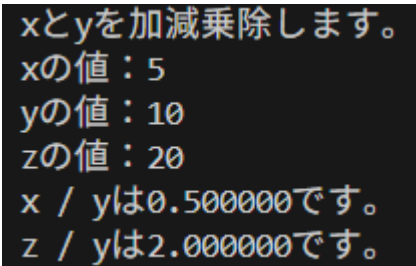
倍精度浮動小数点型（小数）の変数 **y** の宣言. 8バイト=64bit の長さ（**float**型4バイトの2倍の長さ）の変数を確保するという意味.

キーボードからの入力：C

浮動小数点 float型, double 型の変数の宣言

```
-----  
#include <stdio.h> // C言語の標準入出力ライブラリ
```

```
int main() {  
    float x;          // float型の変数  
    double y;         // double型の変数  
    int z;            // int型の変数  
  
    printf("xとyを加減乗除します。\\n");    // printf関数で表示  
  
    printf("xの値：");  
    scanf("%f", &x);                        // %fでfloat型を読み込み、&でメモリアドレスを渡す  
  
    printf("yの値：");  
    scanf("%lf", &y);                       // %lfでdouble型を読み込み、&でメモリアドレスを渡す  
  
    printf("zの値：");  
    scanf("%d", &z);                        // %dでint型を読み込み、&でメモリアドレスを渡す  
  
    // 浮動小数点数での演算  
    // C言語では、特に指定がない場合、double型で演算されることが一般的です。  
    // float型を演算に使用する場合、doubleにキャストするとより正確な結果が得られます。  
    printf("x / yは%fです。\\n", (double)x / y);    // doubleにキャストして表示  
    printf("z / yは%lfです。\\n", (double)z / y);    // doubleで表示  
  
    return 0;                                // プログラムの正常終了を示す  
}
```



```
xとyを加減乗除します。  
xの値：5  
yの値：10  
zの値：20  
x / yは0.500000です。  
z / yは2.000000です。
```

乱数の生成：C++

乱数の生成

// 0～9のラッキーナンバーを乱数で生成して表示

```
#include <ctime>
```

```
#include <cstdlib>
```

```
#include <iostream>
```

```
using namespace std;
```

```
int main()
```

```
{
```

```
    srand(time(NULL));                // 乱数の種を設定
```

```
    int lucky = rand() % 10;          // 0～9の乱数
```

```
    cout << "今日のラッキーナンバーは" << lucky << "です。 \n";
```

```
}
```

乱数の生成：C++

乱数の生成

ソースコードの意味

「`#include <ctime>`」

「`#include <cstdlib>`」

「`<ctime>`」は日付・時間のライブラリ用ヘッダ

「`<cstdlib>`」は乱数生成関数を含む汎用的な関数用のライブラリ用ヘッダ

「`srand(time(NULL));`」

次の「`rand`」と組み合わせて使う。 `rand` は設定された数（「種」という）を元に乱数を生成関数であり，「種」の値を変えると異なる乱数が生成される。反対に同じ「種」の値を用いると同じ乱数が生成されてしまう。

「`srand`」はその「種」を設定する関数であり，「`srand(time(NULL))`」とすると，このプログラムが実行された時刻「`time`」を入力として「種」の値を設定する。「`time(NULL)`」は現時刻を返す関数。よってプログラムが実行されるたびに異なる「種」が生成される。

乱数の生成：C++

```
「int lucky = rand() % 10;」
```

「rand()」は「種」の値に従って乱数を生成し、「rand() % 10」はその乱数を10で割った余りの0, 1, ..., 9のどれかとなる（10の剰余系）。
よって0, 1, ..., 9のどれかの整数が「ランダム」にint型「lucky」に代入される。

1文字の読み込み：C++

1文字の読み込み

// 文字を読み込んで表示

```
#include <iostream>
```

```
using namespace std;
```

```
int main()
```

```
{
```

```
    char c;    // 文字
```

```
    cout << "文字を入力してください：";    // 文字の入力を促す  
    cin >> c;    // 文字を読み込む
```

```
    cout << "打ち込んだ文字は" << c << "です。\\n";    // 表示
```

```
}
```

文字を入力してください：ha
打ち込んだ文字はhです。

1文字の読み込み：C++

1文字の読み込み

ソースコードの意味

「`char c;`」

1文字を格納する `char` 型の変数 `c` の宣言。メモリ上の大きさは1バイトで、1文字に対応するASCII「文字コード」が格納される。

例えばアルファベットの「A」を `char` 型の変数に格納するということは、「A」のASCIIコードである16進数2桁（＝8ビット＝1バイト）の「41」, つまりそれに対応する2進数の0, 1のビット列が `char` 型の変数に格納される, という事。

「`cin >> c;`」

「`cout << “打ち込んだ文字は” << c << “です。\\n”;`」

キーボードから1文字の入力とモニターへの表示。

1文字の読み込み：C

1文字の読み込み

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

```
int main() {  
    char c;                                // 文字を格納する変数  
    printf("文字を入力してください：");    // 文字の入力を促す  
    scanf(" %c", &c);                      // 文字を読み込む（先頭に空白  
    を入れることで改行文字の読み込みを回避）  
    printf("打ち込んだ文字は%cです。\\n", c); // 表示  
    return 0;  
}
```

```
文字を入力してください：ka  
打ち込んだ文字はkです。
```

文字列の読み込み：C++

文字列の読み込み ソースコードの意味

// 名前を読み込んで挨拶する

#include <string>

#include <iostream>

using namespace std;

int main()

{

string name;

// 名前

cout << "お名前は：";

// 名前の入力を促す

cin >> name;

// 名前を読み込む（スペースは無視）

cout << "こんにちは" << name << "さん。\\n";

// 挨拶する

}

お名前は：reitaku
こんにちはreitakuさん。

文字列の読み込み：C++

文字列の読み込み
ソースコードの意味
「`#include <string>`」

標準ライブラリの「クラス」 `std::string` を扱うためのヘッダ

「`string name;`」

`string`型（文字列）変数の宣言. `std::string` の省略形. 可変長の文字列を格納する変数を用意する.

C言語では`char`型を用いても文字列が扱えるが格納する予定の語長に合わせる必要があり扱いが難しい. 想定外の入力がある場合にプログラムが予想外の処理を進めることがあり危険.

C++ の`string`は語長に合わせて自動で格納場所の大きさを調整してくれるので安全.

文字列の読み込み:C

文字列の読み込み ソースコードの意味

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

```
int main() {  
    char name[256]; // 名前を格納する文字配列（十分なサイズを確保）  
  
    printf("お名前は："); // 名前の入力を促す  
    scanf("%s", name); // 名前を読み込む（スペースは無視）  
  
    printf("こんにちは%sさん。 \n", name); // 挨拶する  
    return 0;  
}
```

```
お名前は： university  
こんにちはuniversityさん。
```

データ型

ここまで見てきた型はC++で扱うデータ型の一部である．それ以外のデータ型や詳しい説明は，教科書，第4章を参照せよ．

課題

スライドの説明に基づいて課題を行ってください。

- Slide 16：ex02-1.cpp を作成すること。
- Slide 21：ex02-2.c を作成すること。
- コードに出力処理を追加し、最初の出力行を <学籍番号><氏名> としてください。

プログラムを実行し、その結果のスクリーンショットをUNIPAに提出してください。

```
PS C:\Users\sxwin\Documents\Lecture\cpp\02> cd "c:\Users\sxwin\Documents\Lecture\cpp\02\"  
; if ($?) { g++ test2-7.cpp -o test2-7 } ; if ($?) { .\test2-7 }  
1234567890 yourname  
xとyを加減乗除します。  
xの値：5  
yの値：10  
zの値：20  
x / yは0.5です。  
z / yは2です。
```

完成例