

# プログラミング基礎 #06

## 関数

担当：向井 智彦

# 前回のおさらい

- for (初期化式; 継続条件式; 更新式) { }
- while (継続条件式) { }
- 反復の中断には break
- 反復対象ブロック内の後略には continue
- 変数のスコープは宣言した中括弧 { } 内
- スコープに対応したインデントで整形を
  - ついでに単語・演算子の前後の空白も

# 本日の内容

- 関数の使い方と作成
- 関数と配列
- ヘッダファイル
- ライブラリ

@ wandbox.org

```
#include <iostream>
#include <cmath>
using namespace std;
int main(void)
{
    cout << sin(0.52359877) << endl;
    cout << cos(0.52359877) << endl;
    return 0;
}
```

# 文法図解

```
#include <iostream>
#include <cmath> 数学関連ヘッダファイル
using namespace std;
戻り値 int main(void)
{
    関数名 sin(引数 0.52359877) << endl;
    cout << cos(0.52359877) << endl;
    return 0;
}
```

The diagram illustrates the grammar of the provided C++ code. It uses arrows to link Japanese labels to specific tokens or expressions:

- 戻り値** (Return Value) points to the `int` type specifier in the `main` function signature.
- 関数名** (Function Name) points to the `main` identifier.
- 引数** (Argument) points to the `void` parameter type in the `main` signature.
- 引数** (Argument) points to the numerical value `0.52359877` in both `sin` and `cos` function calls.

# main関数, 使ってますか？

```
int main(void)
{
    cout << "Hello world!" << endl;
    return 0;
}
```

これって何？

# main関数 = OSが呼び出す関数

1. Windows エクスプローラーで xxx.exe を起動
2. Windows が、xxx.exe の中にある **main** 関数を探し出す.
3. 見つかったmain関数をWindowsが**呼び出す**
4. main関数で”Hello world”出力処理を実行
5. main関数から 0 という**戻り値**がwindowsへ戻り、xxx.exeは正常終了

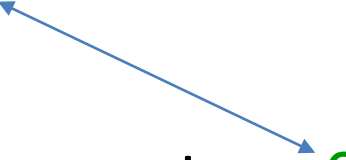
# 関数とは？

- 別の関数から呼ばれて,
  - OS calls main function
- 何かしらの処理を実行し,
  - `cout << "hello world" << endl;`
- 呼び出し元に何らかの値を戻す,
  - `return 0;`
- 処理のひとまとまり



# 何もしないmain関数の作り方

```
int main(void)
{
    return 0;
}
```

A blue arrow points from the green '0' in 'return 0;' to the green 'int' in 'int main(void)'.

- **int**: 戻り値の型
- **main**: 関数の名前
- **void**: 引数なし
- **return 0;**
  - 値を戻してmain関数を終了

## 関数の基本形

```
int main(void) { return 0; }
```



戻り値の型名    関数名 (引数リスト)    { 処理; }

# 関数とは？

- 別の関数から呼ばれて,
  - OS calls main function
- 何かしらの処理を実行し,
  - `cout << "hello world" << endl;`
- 呼び出し元に何らかの値を返す, ←何型？
  - `return 0;`
- 処理のひとまとまり ←名前は何？

# 関数の引数

- 名前(○○, × ×)
  - の「○○」とか「× ×」
- 呼び出し元から, 呼び出し先に引き渡す数
  - 例: `atan2(1.0, 0.5);`
    - 引数1: 1.0 という浮動小数
    - 引数2: 0.5 という浮動小数

$$\tan^{-1} \frac{1.0}{0.5}$$

## 関数の基本形

```
int main(void) { return 0; }
```



戻り値の型名    関数名 (引数リスト)    { 処理; }

# 戻り値と引数 補足

戻り値の型名 関数名 (引数リスト) { 処理; }

- 戻り値は1つのみ (2つ以上戻す方法は来週)
  - `int function() { return x; }`
- 戻り値がない場合は `void`
  - `void function() { ... };`
- 引数は0個以上の任意の数
  - `int Sum(int v0, int v1, ..., int v99) { ...; return sum; }`
- 引数がない場合は `void`
  - `void PushButton(void) { ... }`

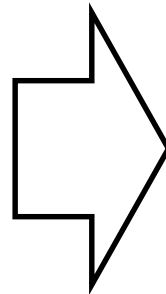
# 関数の使い方 (数学の例)

- 絶対値関数 `double fabs(double n);`  
– `cout << fabs(-52.0) << endl;`
- 正弦 `double sin(double x); //rad`  
– `cout << sin(3.14 / 6.0) << endl;`
- 逆正接 `double atan2(double y, double x);`  
– `cout << atan2(1.0, 1.0) << endl;`
- べき乗 `double pow(double x, double y);`  
– `cout << pow(3.2, 0.7) << endl;`

# 関数作成法：mainを修正

戻り値の型名   関数名 (引数リスト)   {   処理;   }

```
int main(void)
{
    cout << "Hello!";
    return 0;
}
```



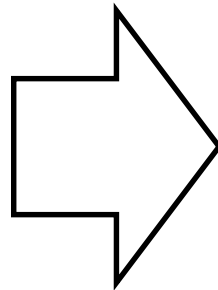
```
int Hello(void)
{
    cout << "Hello!";
    return 0;
}
```

Hello関数完成

# Helloを呼び出す

戻り値の型名 関数名 (引数リスト) { 処理; }

```
int Hello(void)
{
    cout << "Hello!";
    return 0;
}
```



main関数を  
追加し、  
Hello関数を  
呼び出す

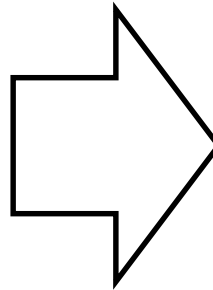
```
int Hello(void)
{
    cout << "Hello!";
    return 0;
}
int main()
{
    Hello();
    return 0;
}
```

# 引数を加える

戻り値の型名 関数名 (引数リスト) { 処理; }

戻り値は不要→void

```
void Hello(void)
{
    cout << "Hello!";
    return 0;
}
int main()
{
    Hello();
    return 0;
}
```



関数名を修正

```
void Print(int hiki)
{
    cout << hiki;
}

int main()
{
    int x = 5;
    Print(x);
    return 0;
}
```

変数の持つ  
値を渡す



# 来週の予告：関数とスコープ

戻り値の型名 関数名 (引数リスト) { 処理; }

```
void ClearPrint(int hiki) {  
    hiki = 0;  
    cout << hiki << endl;  
}  
  
int main() {  
    int hiki = 10;  
    ClearPrint(hiki);  
    cout << hiki << endl;  
    return 0;  
}
```

## 変数宣言の重複

```
#include <iostream>  
int main(void) {  
    int sum = 0;  
    for (int i = 0; i < 10; i = i + 1) {  
        int sum = 0;  
        sum = sum + i;  
    }  
    std::cout << sum << std::endl;  
}
```

青sumと赤sumは別物

赤sumのスコープが優先

# 配列と関数

空の大括弧

配列の要素数

```
double Sum(double a[], int n) {  
    double sum = 0.0;  
    for (int i = 0; i < n; ++i)  
        sum += a[i];  
    return sum;  
}
```

どんな動作？

```
int main() {  
    double data[4] = {0.0, 1.0, 2.0, 3.0};  
    cout << Sum(data, 4) << endl;  
}
```

配列名を指定

# 配列と関数 contd.

```
void Clear(double a[], int n) {  
    for (int i = 0; i < n; ++i)  
        a[i] = 0.0;  
}
```

どんな動作？

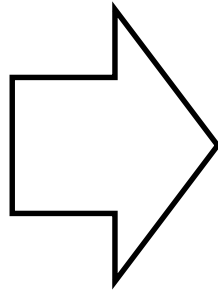
```
int main() {  
    double data[3] = {5.0, 4.0, 3.0};  
    Clear(data);  
    cout << data[0] << endl;  
    cout << data[1] << endl;  
    cout << data[2] << endl;  
}
```

# 配列と関数：できないこと

```
int[] CreateArray() {  
    int a[3] = {0, 0, 0};  
    return a; // 配列全体は戻せない  
}  
  
int main() {  
    double data[] = CreateArray(data); //エラー  
    cout << data[0] << endl;  
    cout << data[1] << endl;  
    cout << data[2] << endl;  
}
```

# 少し実験

```
void Print(int hiki)
{
    cout << hiki;
}
int main()
{
    Print(5);
    return 0;
}
```

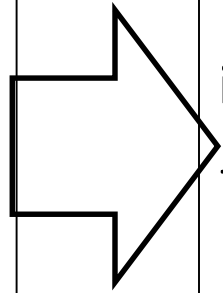


main関数と  
Print関数の  
順番を  
入れ替える

```
int main()
{
    Print(5);
    return 0;
}
int Print(int hiki)
{
    cout << hiki;
}
```

# プロトタイプ宣言

```
void Print(int hiki)
{
    cout << hiki;
}
int main()
{
    Print(5);
    return 0;
}
```



```
void Print(int hiki);
```

```
int main()
```

```
{
```

```
    Print(5);
```

```
    return 0;
```

```
}
```

```
int Print(int hiki)
```

```
{
```

```
    cout << hiki;
```

```
}
```

↑  
{ 処理; } が無い代わりに  
「;」がある

{ 処理; } がある以外は  
プロトタイプ宣言と一緒に



変数の宣言 ⇔ 関数のプロトタイプ宣言 というアナラジー

# #include および ヘッダファイルの正体

- ヘッダファイル＝関数プロトタイプ宣言の羅列
  - sin 関数にはどんな引数がある？
  - 引数の数は？引数の型は？
  - 戻り値はどんな型？
- ライブラリが提供する関数のカタログリスト
  - ...のようなもの
  - 中身のコードは知らずとも使うための情報
  - #include することで呼び出し可能に

# 関数を活用するメリット

- プログラムを部品化（モジュール化）
  - よく使う処理をまとめておく
  - バグを減らす, 読みやすくする
  - 開発作業量の最小化
  - 過去の資産の使い回し
- 他者の力を借りる・他者に力を借す
  - ライブラリ, API, ミドルウェア等の機能呼び出して活用可能



# まとめ

戻り値の型名 関数名 (引数リスト) { 処理; }

- 関数
  - 処理のひとまとまり
  - 数値を受け取って(引数)、出力する(戻り値)
- 関数の戻り値と引数
  - 戻り値は1つ以下、引数は0個以上
  - なにも無い場合は void
  - 配列は引数に指定できる、戻り値にはできない
- 関数プロトタイプ宣言とヘッダファイル

# 演習課題01：数学関数

提出期限：11/12(月)、ファイル名：06-01.cpp

下記5つの数値について、それぞれ数学関数を用いて計算するプログラムを作成

1. 9234893.0 の平方根
2. 2349.0 の自然対数
3. 3.1 の 1.6乗
4.  $\sin(28.3)$  [degree] ←単位に注意
5.  $-20 \sim 30$  の整数の絶対値の和 ← for文利用

# 演習課題02：平均値算出関数

提出期限：11/12(月)、ファイル名：06-02.cpp

- 5つの浮動小数の平均値を計算して戻す関数「average」の作成
  - main関数の中身は変更しない
  - average関数をゼロからコーディング
    - main関数の後に書き足す場合は関数プロトタイプ宣言を忘れずに
  - エラーが生じないこと&正しく動作することを確認

# 演習課題Extra：二項分布

提出期限：11/12(月)、ファイル名：06-ex.cpp

- 二項分布を計算する関数「Binomial」を作成
  - $Bin(m, M, \mu) = {}_M C_m \cdot \mu^m \cdot (1 - \mu)^{M-m}$
  - ${}_M C_m = \frac{M!}{m! \cdot (M-m)!}$
- ※二項分布
  - コインをM回投げたときに、表がm回出る確率
  - 1回投げたときに表が出る確率は $\mu$ (0~1.0)
- コインを5回投げた時に表が何回出るか実験
  - コイントス関数  $CoinToss(\mu)$  & 実験コードは用意済
  - 二項分布の確率と比較