補足資料（１．５回目）　C言語の関数

☒ 補足資料：C言語の関数の基本

☒ 関数とは？

C言語の**関数**は、処理をひとまとめにして何度も呼び出せる仕組みです。
プログラムの**再利用性・可読性**を高めるために使います。

```
#include <stdio.h>

int main() {
    hello(); // 関数呼び出し
    return 0;
}

// 関数定義
void hello() {
    printf("Hello, world!\\n");
}
```

☒ 宣言と定義の違い

名称	役割	書く場所	例
関数宣言（プロトタイプ宣言）	関数があることをコンパイラに知らせる	mainより上	`int add(int a, int b);`
関数定義	実際の処理内容を書く	mainより下でもOK	`int add(int a, int b) { return a + b; }`

☒ ポイント

- 宣言と定義を分けることで、**関数を別ファイルに分割**できる。
- 宣言だけをヘッダファイル（`.h`）に書くのが一般的。

☒ 引数と仮引数の違い

種類	意味	例
仮引数（parameter）	関数の受け取り側の変数	`int add(int a, int b)` の a,b
実引数（argument）	関数を呼び出すときに渡す値	`add(3, 5)` の 3,5

☒ イメージ図

```
int add(int a, int b) ← 仮引数
      ↑    ↑
      |    |
  add(3, 5); ← 実引数
```

☒ ポインタ渡し

関数に変数の**アドレス（場所）**を渡すことで、呼び出し元の値を直接変更できます。

```
#include <stdio.h>

void addOne(int* x) {
    *x = *x + 1;
}

int main() {
    int num = 10;
    addOne(&num); // 変数のアドレスを渡す
    printf("%d\\n", num); // → 11
}
```

☒ ポイント

- `int* x` は「int型のポインタ」
- `*x` は「ポインタの指す実際の値」
- 呼び出し元の変数を直接書き換えることができる

⚙ 戻り値・void・手続きと関数

C言語の関数には、**値を返す関数**と**値を返さない手続き（void関数）**の2種類があります。

```
// 値を返す関数
int add(int a, int b) {
    return a + b;
}

// 値を返さない（void関数）
void printHello() {
    printf("Hello!\\n");
}
```

☒ return文

- 関数の実行を終了し、呼び出し元に値を返す。
- void関数では return を省略可能。

☒ C言語の関数まとめ

分類	説明	例
宣言（プロトタイプ）	関数の存在を知らせる	`int add(int a, int b);`
定義	実際の処理を書く	`{ return a + b; }`
実引数／仮引数	呼び出す値と受け取る変数	`add(3, 5)` ／ `int add(int a, int b)`
値渡し	値のコピーを渡す	元の変数は変わらない
ポインタ渡し	アドレスを渡す	元の変数が変わる
戻り値	処理結果を返す	`return` を使う
void関数	戻り値なしの手続き	画面出力など

☒ まとめ

- 宣言＝関数の存在を伝える
- 定義＝関数の中身を書く
- 引数はコピーか参照かで挙動が変わる
- ポインタ渡しはC言語の“参照渡し”のようなもの
- 戻り値の有無で「手続き」と「関数」を区別できる

練習問題！

☒ 第1問：平均点と評価を返す関数

3科目の点数を入力し、**平均点を返す関数**と、
その平均に応じて**評価（A〜D）を返す関数**を作れ。

```
float getAverage(int a, int b, int c);
char getRank(float avg);
```

☒ **仕様**

- `getAverage()` … 平均値を計算して返す
- `getRank()` … 80点以上→A、60点以上→B、40点以上→C、それ以下→D

☒ *例*
入力: `80 70 90`
出力:

```
平均点 = 80.0
評価 = A
```

☒ 第2問：配列の最大値と最小値を求める関数

配列とその要素数を受け取り、最大値と最小値を**ポインタ経由で返す関数**を作れ。

```
void getMinMax(int arr[], int n, int* min, int* max);
```

☒ **仕様**

- ループで配列を走査し、`*min`, `*max` に代入
- mainで表示する

☒ *例*
入力: `{3, 9, 2, 5}`
出力: `最大値=9, 最小値=2`

☒ ヒント：ポインタで関数内の変数を書き換える例

```
#include <iostream>
using namespace std;

void AddOne(int* x) {
    *x = *x + 1;  // *xを通して実際の変数の中身を変更する
}

int main() {
    int num = 10;
    AddOne(&num);  // numのアドレスを渡す
    cout << "num = " << num << endl;  // 出力：num = 11
}
```

☒ `*x` はアドレス先の実体、`&num` は「numの住所」です。

☒ 第3問：配列の平均値より大きい値を表示

整数配列と要素数を引数に受け取り、
平均値を求めて、その平均より大きい要素だけを出力する関数を作れ。

```
void printAboveAverage(int arr[], int n);
```

☒ **仕様**

- 関数内で平均値を計算
- 平均より大きい要素をすべて表示

☒ *例*

入力: `{60, 80, 70, 90}`
出力:

```
平均値: 75.0
平均より大きい値: 80 90
```

☒ 第4問：文字列中の特定文字を数える関数

文字列と文字を受け取り、その文字がいくつ含まれているか数える関数を作れ。

```
int countChar(char str[], char target);
```

または

```
int CountChar(const string& str, char target);
```

☒ 仕様

- `for` 文で `str[i] != '\0'` まで走査
- 一致したらカウントを増やす

☒ *例*
入力: `"banana"`, 文字 `'a'`
出力: `文字 'a' は 3 個含まれています`

☒ ヒント：`std::string` を1文字ずつ調べるには？

```
#include <iostream>
#include <string>
using namespace std;

int main() {
    string word = "banana";
    for (int i = 0; i < word.size(); i++) {
        cout << word[i] << " ";  // 1文字ずつ出力
    }
    cout << endl;
}
```

☒ `string` は `[]` で1文字ずつアクセスできる。
`word.size()` で文字数（終端の'\0'を除く）を取得できる。

☒ 第5問：関数を組み合わせて統計出力

以下の関数を組み合わせて、5人の点数から平均・最高・最低を出力するプログラムを作れ。

```
float getAverage(int arr[], int n);
int getMax(int arr[], int n);
int getMin(int arr[], int n);
```

☒ 仕様

- mainで配列を作り、関数を順に呼び出す
- 結果を整形して表示

☒ *例*
入力: `{70, 85, 60, 90, 75}`
出力:

```
平均点: 76.0
最高点: 90
最低点: 60
```

Revision #3
Created 8 October 2025 04:50:58 by youe2
Updated 2 June 2026 19:00:00 by youe2