プログラミング及び実習2

第 3 回 関数（main() と return）、変数型

佐野 彰

本日の目標

1. C 言語の関数を理解する（`main()` 以外の関数定義は後半の回で）
    - `main()` 関数、 `return` 文
2. 整数型・実数型・ `void` 型の変数とキャストを理解する

1. ソースコードのデバッグ

ソースコードをデバッグ（debug）する

```
$ cc ex03.c -o ex03
ex03.c:1:9: error: expected ‘=’, ‘,’, ‘;’, ‘asm’ or
‘__attribute__’ before ‘<’ token
    include <stdio.h>
            ^
```

- `ex03.c` というファイルの **1 行目の 9 文字目**で error が発生した
- `expected ～` — ここには「～」が来るはずだった、という意味
- `before '<'` — `<` の前のあたりで、コンパイラがソースコードを解釈できなくなった

error と warning

```
ex03.c:3:1: warning: type specifier missing, defaults to 'int' [-Wimplicit-int]
main(){
^
ex03.c:5:5: error: implicit declaration of function 'tScanf' is invalid in
C99 [-Werror,-Wimplicit-function-declaration]
    tScanf("%d",&n);
    ^
1 warning and 1 error generated.
```

- error はエラー：コンパイルに失敗した
- warning は警告：コンパイルはできたが、注意が必要

2. main() 関数と return 文

関数（function）

数学の関数 $f(x),\ f : \mathbb{R} \to \mathbb{R}$（例: $f(x) = x^2$）と同じように、プログラミング言語の関数も「入力を受け取り、出力を返す」処理のかたまりです。

```
int a;
int v = func(a);   // 例: printf(), scanf()
```

- 入力（引数）がない場合もある
- 出力（返り値）がない場合もある

入力がない関数、出力がない関数

入力（引数）	出力（返り値）	例
あり	あり	`sqrt(2.0)` は 2.0 を受け取り、平方根 1.41421… を返す
なし	あり	`rand()` は何も受け取らず、乱数を返す。`getchar()` は何も受け取らず、キーボードから 1 文字読んで返す
あり	なし	`srand(1)` は乱数の種 `1` を受け取るが、何も返さない
なし	なし	後の回で自作する（例: 区切り線を表示するだけの関数）

- 返り値がある関数は、`int v = rand();` のように結果を変数に代入して使える
- 返り値がない関数は、`srand(1);` のように呼び出すだけ。`int v = srand(1);` とは書けない

main() 関数

- C 言語で**最初に呼び出される関数**
- 実行可能な C プログラムを書く ＝ `main()` 関数を定義する
- `return` 文で関数の値（**返り値**）を返す

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

- `main` という名前の関数で、返り値の型は `int`。`0` を返す

main() or int main()

```
$ cc foo.c
foo.c:1:1: error: type specifier missing, defaults to 'int';
ISO C99 and later do not support implicit int [-Wimplicit-int]
    1 | main() {
      | ^
      | int
```

- 古い C 規格では、返り値の型を省略すると `int` と見なされた（`main()` と書けた）
- 最近の C コンパイラでは、省略が許されないことがある（例: macOS 26 の clang）

→ **`int main()` のように、返り値の型を省略せずに書きましょう。**

C 言語の代表的な標準規格（参考）

規格	main() の書き方
K&R C (1978)	`main()` が一般的
ANSI C89 / ISO C90	`main()` も許容、正式には `int main(void)`
C99 以降（C99/C11/C23）	`main()` は規格違反、`int main(void)`（`void` は省略可）

コンパイラやバージョンによって、警告・エラーの出方は異なります。

関数と返り値と return 文

- C 言語の関数は `型 関数名() { ... }` の形で定義する
  - 今回は `main()` だけ。詳細は後の回で扱う
- 関数は `関数名()` の形で呼び出す（使う）
- 実は、`scanf()` と `printf()` も `int` 型の値を返す
  - `scanf()` は、入力できた値の個数を返す
  - `printf()` は、表示した文字数を返す（改行 `\n` も 1 文字）

```c
int a, n;
n = scanf("%d", &a);                // 5 を入力すると n は 1
n = n + printf("a is %d\n", a);     // "a is 5\n" は 7 文字なので、n は 1 + 7 = 8
```

main() 関数の return 文に意味はあるのか？

`main()` の返り値は、プログラムを実行した処理系（システム）やユーザーに、そのプログラムが**どういった状況で終了したか**を伝えます。

```c
int main() {
    if (/* エラー A が発生した！ */) return 1;
    if (/* ユーザーに終了が指示された！ */) return 99;
    return 0; // 正常終了
}
```

main() の返り値をシェルで確認する

直前に実行したプログラムの返り値は、シェル（bash, zsh）の `echo $?` で表示できます。

```c
// ret.c
int main() {
    return 99;
}
```

```
$ cc ret.c -o ret
$ ./ret
$ echo $?
99
```

- `return 0;` のプログラムなら、`echo $?` は `0` と表示する
- 慣習として `0` は正常終了、0 以外は何らかの異常を表す

3. 変数の型

変数の型（type）

- **基本型**
  - 整数型（`char`, `short`, `int`, `long`）— `char` は後日
  - 浮動小数点型（`float`, `double`, `long double`）
- void 型

整数型（integer types）

整数を保持する変数や定数の型。型ごとに違うのは、基本的に保持できるサイズだけです。

型	サイズ	範囲（目安）
`char`	1 バイト = 8 ビット	-128〜127
`short`	2 バイト以上	-32,768〜32,767
`int`	2 または 4 バイト	（処理系依存）
`long`	4 バイト以上	-2,147,483,648〜2,147,483,647
`long long`（C99〜）	8 バイト以上	

大きさは処理系によって異なりますが、大小の順序は決まっています。

```
char <= short <= int <= long <= long long
```

浮動小数点型（floating point types）

実数を保持する変数や定数の型。型ごとに、保持できるサイズと精度が異なります。

型	サイズ
`float`	4 バイト = 32 ビット
`double`	8 バイト = 64 ビット
`long double`	処理系により 8〜16 バイト

大きさは C 言語の規格では決められていません。
ただし、ほとんどの処理系は IEEE 754 という規格に沿っています。
そのため、`float` と `double` は上の表の通りだと考えてよいです。

void 型

値がないことを明示的に示すための型。

```
void func(void) {   // 引数なし（void）、返り値なし（void）の関数
    int a;          // int 型の変数 a は定義できるが、
    void v;         // ✗ void 型の変数は定義できない（エラー）

    return;         // 返り値がないので、return 0; とは書かない
}
```

- 引数の `void` は「引数がない」、返り値の `void` は「返り値がない」ことを表す

void main() は書いてはいけない

```c
void main() {        // ✗ ダメな例
    printf("Hello\n");
}
```

- `main()` は、実行したシステムに終了状況を伝える関数。返り値の型は **`int` と決まっている**
- `void main()` は C 言語の規格に反する。コンパイラが警告を出す（環境によってはエラー）
  - 例: `warning: return type of 'main' is not 'int'`
- 引数がないことを明示するなら、`int main(void)` と書く

```c
int main(void) {    // ○ 正しい書き方
    return 0;
}
```

unsigned（符号なし変数）

`int` などの整数型の変数は、型名の前に `unsigned` を付けると**符号なし変数**として宣言できます。

```
int a;                  // int 型 (-2,147,483,648〜2,147,483,647)
unsigned int ua;        // unsigned int 型 (0〜4,294,967,295)

ua = -1;                // 負数を代入できてしまうので注意！
printf("ua = %u\n", ua);  // %u で unsigned int として表示
```

4. 型の変換

異なる型での算術演算

異なる型どうしの演算では、値がより上位の型に自動的に（暗黙的に）変換されてから計算されます。
`char` と `short` の値は、演算のときにすべて自動的に `int` に変換されます。

```
int < long < long long < float < double < long double
```

```
int i = 2; long l = 3; float f = 2.1; double d = 3.0;

i + l;   // i は long に変換され、結果は long (5)
i * f;   // i は float (2.0) に変換され、結果は float (4.2)
i * 0.1; // i は double (2.0) に変換され、結果は double (0.2)
```

代入による型変換

型の異なる変数に代入すると、値は自動的に（暗黙的に）その変数の型に変換されてから代入されます。

```
int i = 2; long l = 3; float f = 2.1; double d = 3.0;

i = 0.1;   // 0.1 は int に変換され、i は int (0)
i = f;     // f は int に変換され、i は int (2)
l = f * i; // f*i は float (4.2) → long に変換され l は long (4)
```

`a = b;` の代入では、`b` の値は常に `a` の型に変換されます。

明示的な型の変換（キャスト）

キャスト（cast） を使うと、型を明示的に変換できます。

```
(type)x;  // x の型は明示的（強制的）に type 型に変換される
```

```
int i = 2; long l = 3; float f = 2.1; double d = 3.0;

(int)0.1;       // 0.1 は int に変換され int (0)
l / (double)i; // i は double に変換され、結果は double (1.5)
(int)(f * d);  // f*d は double (6.3) → int に変換され int (6)
```

演算の順序によって、暗黙の変換とキャストのタイミングが変わる点に注意してください。

次回予告と今日やること

次回（10/15）：文字型、書式付き入出力、繰り返し処理、標準ライブラリ

今日の 5 講時（実習）

1. 第 2 回課題の解答例を演習ページで確認する（if / while の復習）
2. `main()` の書き方と `return` の意味を確認する
3. 演習課題 ex03 に取り組み、100 点以上を取る

質問は **Teams「★ 質問用チャネル」** か、教室の TA へ。