第6章　「関数」のまとめ

6.1　用語のまとめ

教 p.132

関数とは　繰返しつかうプログラムの一部の命令列を部品として、再利用できるようにしたもの。（他の言語ではサブルーチン・手続き・副プログラムなどとも呼ばれる。）

教 p.133

関数定義とは

分類	一般形
関数定義	**型** 関数名 (**型** 変数名 $_1$, . . . , **型** 変数名 $_n$) 複合文

関数定義には型が必要である。

かっこの中の変数（変数名 $_1$〜変数名 $_n$）は _____ (parameter) と呼ばれる。

C 言語の関数定義は必ずプログラムのトップレベルに書く。（つまり、関数定義の中に関数定義は書けない。）

教 p.134

関数呼出し式とは

分類	一般形
関数呼出し式	関数名 (式 $_1$, . . . , 式 $_n$)

関数呼出しには型は不要である。

かっこの中の式（式 $_1$〜式 $_n$）は _____ (argument) と呼ばれる。

これで文法上、式 (expression) になる。

関数を呼出すと、プログラムの実行は呼び出された関数の定義の先頭に移り、実引数の値が仮引数の変数の初期値になる。

教 p.135

return 文

分類	一般形	補足説明
return 文	`return` 式 ;	値を返す場合
	`return` ;	値を返さない場合

関数の呼出し元に値を返す。つまり、プログラムの実行が関数の呼出し元に戻り、return 文の式の値が、関数呼出し式の値になる。

関数本体の最後の `}` にたどり着いた時も、プログラムの実行は関数の呼出し元に戻る。

教 p.140

値渡し（pass by value）、値呼び（call by value）　引数は基本的に値がやりとりされる。関数呼出しのたびに仮引数のための新しいメモリ領域（“箱”）が用意される。仮引数の変数に値の代入を行なっても、呼出し元の実引数は＿＿＿＿＿＿＿＿＿＿。

教 p.142

値を返さない関数　関数の定義の返却値型のところに＿＿＿＿と書く。

教 p.144

引数のない関数　関数の定義の仮引数のならびを書くところに＿＿＿＿と書く。関数を呼出すときは () のなかは空にする。

教 p.145

有効範囲 (スコープ、scope)　変数には有効範囲がある。同じ変数名でも有効範囲が異なれば別の変数になる。

- ブロックの中で宣言された変数（局所変数、ブロック有効範囲を持つ変数）は、宣言された場所から、＿＿＿＿＿＿＿＿＿＿までが有効範囲となる。
- 関数の仮引数は、その関数本体が有効範囲となる。
- 関数の外で宣言された変数（大域変数・グローバル変数、ファイル有効範囲を持つ変数）は、宣言された場所から＿＿＿＿＿＿＿＿＿＿までが有効範囲となる。

教 p.147

関数プロトタイプ宣言 (function prototype declaration)　関数を定義より前に使用する場合は、関数プロトタイプ宣言が必要である。

以下を “宣言” に追加する。

分類	一般形
関数プロトタイプ宣言	**型** 関数名 (**型** 変数名 , . . . , **型** 変数名) _

変数名は省略可能である。

関数定義がその呼出しよりも前にある場合は、定義が宣言を兼ねるのでプロトタイプ宣言は不要である。（いずれにしても、実行は常に `main` から開始される。）

教 p.148

ヘッダとインクルード

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

の stdio.h は、`printf`, `putchar` などの関数のプロトタイプ宣言が集められたもの（通常はファイル）である。このようにプロトタイプ宣言やマクロの定義が集められたものを＿＿＿＿＿＿と呼ぶ、

`#include` はヘッダの内容を、そっくりそのままその場所に読み込む（インクルードする）指令である。

処理系により標準のヘッダがおかれる場所は異なる。

ライブラリ関数（前もって用意された関数）を利用する時は、適切なヘッダをインクルードする必要がある。例えば、sin, cos, sqrt などの数学関数を利用する時は math.h というヘッダをインクルードする。

関数の汎用性　できるだけ大域変数を使わないようにする。 教 p.149

教 p.149

Warning　発音は [wɔːniŋ] で、カタカナでは＿＿＿＿＿＿＿＿が近い。警告という意味で、エラーではないが間違っている可能性が高いことを示す。

教 p.150

配列の受渡し　関数の引数として配列を渡すこともできる。仮引数の宣言は、型名 引数名 [] としておき、実引数としては＿＿＿＿＿＿＿＿だけを書く。

- 関数に配列を引数として渡す場合、コピーではなく、配列そのもの（正確にいうと、配列の先頭要素のアドレス）が渡される。
  - 一方、int, double 型などの配列でない普通の型の引数の場合は、値がコピーされて渡される。
  - 関数の中で、配列の要素の値を変更すると、呼出し側の配列に反映される。
    int, double 型などの普通の型の引数の場合は、呼出し側には反映されない。
- 引数として渡された配列の要素数を関数の中で知る方法はないので、要素数も引数として渡す必要がある。

教 p.152

const 型修飾子　関数の引数の配列が書換えられないことを保証するためには、＿＿＿＿＿＿という型修飾子を仮引数の宣言につける。つけているのに書き換えようとするとコンパイル時にエラーになる。

教 p.155

番兵法 (sentinel)　探索の対象となっているデータ（＿＿＿＿ (sentinel)）をデータの最後に付け加えること。探索範囲の終わりのチェックをする必要がなくなるので、少し効率が良くなる。

教 p.161

有効範囲と識別子の可視性　同名の変数の有効範囲が重なる時、より内側のブロックで宣言されているものが優先する。

教 p.162

記憶域期間　C 言語の変数の寿命（記憶クラス, storage class）には 2 種類のものがある。

- 自動変数 (automatic variable) — 自動記憶域期間を持つ変数
  - ＿＿＿＿＿＿＿＿定義された変数で static という修飾子がついていないもの

  - プログラムの流れが宣言を通過する時に、変数のための領域（箱）が確保され、初期化される。有効範囲を抜ける時に箱が回収される。
  - 初期化子が与えられていない場合、その値は _____ となる。

- 静的変数 (static variable) — 静的記憶域期間を持つ変数
  - _____ で定義・宣言された変数、または関数の中で宣言された変数で、static という修飾子がついているもの
  - _____ に変数のための領域（箱）が生成され、初期化される。プログラムの終了時まで回収されない。
  - 初期化子が与えられていない場合、_ に初期化される。

6.2 プログラム例

値呼びの確認

```
1   #include  <stdio.h>
2
3   void i_set(int v) {
4       v = 0;
5   }
6
7   int main(void) {
8       int a1 = 1, a2 = 3;
9
10      i_set(a1);
11      i_set(a2);
12
13      printf("a1␣=␣%d\n", a1);
14      printf("a2␣=␣%d\n", a2);
15
16      return 0;
17  }
```