プログラミング論I
(14) 変数の種類と，その他の演算子 など

電子情報工学専攻　日浦 慎作

ログインしておいてください

本日のお題

これまでに紹介出来なかった，C言語のいろいろ

- グローバル変数と static 変数
  - プログラム全体から見える変数など
- ビット演算子・シフト演算子
  - 変数を2進数として，1ビットずつ扱うための演算子
- ３項演算子
  - if文のかわりに，式だけで条件判断して動作できる
- switch-case文
  - 一度に，多数の分岐を行うことができる

グローバル変数(大域変数)

```
#include <stdio.h>

int g;
void func(void);

int main(void) {
  g = 100;
  printf("%d¥n", g);
  func();
  printf("%d¥n", g);
  return 0;
}

void func(void) {
  g = 200;
}
```

グローバル変数(大域変数)

- すべての関数から見える変数
  - 関数の外で定義した変数は,
それ以降, どこからでも見える
  - 引数や戻り値を経由しなくても,
値のやり取りができる
- 使いすぎに注意!
  - 思わぬ「副作用」に悩むことも.
- 通常の変数は局所変数と呼ぶ.

例1　グローバル変数の働き ex1.c

```c
#include <stdio.h>

int g;
void func1(void);
void func2(void);

int main(void) {
  g = 100;
  printf("%d¥n", g);
  func1();
  printf("%d¥n", g);
  func2();
  printf("%d¥n", g);
  return 0;
}

void func1(void) {
  g = 200;
}

void func2(void) {
  int g;
  g = 300;
}
```

実行結果

```
100
200
200
```

200に書き換えられてる
300に書き換えられていない

グローバル変数 g に
200を書き込んでいる

関数内で局所変数 g を定義して
いるため，グローバル変数 g に
アクセスができなくなる

static変数（静的変数）

```
#include <stdio.h>

void func1(void);

int main(void) {
  func1();
  func1();
}

void func1(void) {
static int s = 0;

  printf("s = %d¥n", s);
  s++;
}
```

static変数（静的変数）

- 関数から出ても，値が保持される変数
  - 関数の外からはアクセスできない
  - 初期化は，最初の１回だけ行われる

実行結果

```
s = 0
s = 1
```

１回目の呼び出し時に1増やされ，
2回目の呼び出し時は1を保っている

例2　static変数の働き ex2.c

```c
#include <stdio.h>

int g = 0;
void func1(void);

int main(void) {
  func1();
  func1();
  func1();
}

void func1(void) {
  int l = 0;
  static int s = 0;

  printf("l = %d, s = %d, g = %d¥n", l, s, g);
  s++;
  g++;
}
```

実行結果

```
l = 0, s = 0, g = 0
l = 0, s = 1, g = 1
l = 0, s = 2, g = 2
```

- 局所変数は，毎回初期化される
- 初期化しなくても，次回呼び出し時には前回の値は失われる

- static変数は，最初の1度しか初期化されない

l : local (局所変数，ローカル変数)
s : static (静的変数，static変数)
g : global(大域変数，グローバル変数)

ビット演算子・シフト演算子

- 2進数のビットごとの処理を行う
- 2項演算子
  - OR　　|　　（各ビットのORで，
  - AND　　&　　（各ビットのANDで，
  - 排他的論理和（XOR）　^
  - 左シフト　　<<
  - 右シフト　　>>
- 単項演算子
  - NOT　　~　　（各ビットの0と1を反転）

|| とは異なる）

&& とは異なる）

|| や && は論理演算子で，等号・不等号の判定結果の「または」や「かつ」の役割をする．0かどうかを扱う

```c
#include <stdio.h>

void printBin(char c) {
  int i;

  for(i = 7; i >= 0; i--) {
    printf("%d", (c >> i) & 1);
  }
  printf("¥n");
}

int main(void) {
  char a = 0b00001111;
  char b = 0b00110011;

  printf("a       :"); printBin(a);
  printf("b       :"); printBin(b);
  printf("a | b   :"); printBin(a | b);
  printf("a & b   :"); printBin(a & b);
  printf("a ^ b   :"); printBin(a ^ b);
  printf("a << 1  :"); printBin(a << 1);
  printf("a >> 1  :"); printBin(a >> 1);
  printf("~a      :"); printBin(~a);
}
```

実行結果

```
a       :00001111
b       :00110011
a | b   :00111111 OR
a & b   :00000011 AND
a ^ b   :00111100 XOR
a << 1  :00011110 左シフト
a >> 1  :00000111 右シフト
~a      :11110000 反転
```

c の先頭から1ビットずつ取り出す処理
(c を i だけ右シフトし，
最下位ビットを & で取り出している)

2進数の定数は
0b01011111
のように書ける．
(16進数は0x5Fのように)

b : binary

例3 ビット演算 ex3.c

3項演算子

- 条件により式を選択することができる演算子

```
a = b > 1 ? 100 : 200;
```

は，以下と同じ動作をする

```
if(b > 1)
   a = 100;
else
   a = 200;
```

- 代入演算子の次に優先順位が低い
  - ただし，不安なときは( )を用いれば良い（見やすい）

```
a = (b > 1) ? (a * 2) : (a * 3);
```

練習問題

1. char型の変数を引数にとり，その値の1のビット数を数えて返す関数を作成せよ.
   - 関数のプロトタイプ宣言は
     `int countBit(char c);`
   - ex3.c の printBin()を参考にすると良い.
2. 3項演算子を用い，変数iが偶数ならeven，奇数ならoddと表示するプログラムを1行で作成せよ.
   - `printf("%s", _____);`
     の下線部を埋めればよい.

switch-case文

- if文をたくさん連ねるのは面倒くさい・・
  - 変数の値によって，一度に多数の分岐を行う

```
int a = 0;
switch(a) {
case 0:
        printf("a is 0¥n");
        break;
case 1:
        printf("a is 1¥n");
        break;
default:
        printf("a is neither 0 nor 1¥n");
        break;
}
```

整数の変数や式を書くことができる

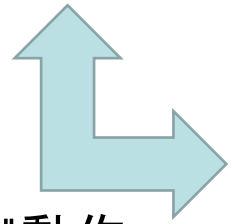
整数の定数を書く（式は書けない）

switch-case文から抜ける

どの定数にも当てはまらない場合はココ

注意点
- コロン : が使われる
- break がないと下へ落ちる

```
switch(a) {
case 0:
        printf("a is 0¥n");
        break;
case 1:
        printf("a is 1¥n");
        break;
default:
        printf("a is neither 0 nor 1¥n");
        break;
}
```



同じ動作

```
if(a == 0) {
        printf("a is 0¥n");
}
else if(a == 1) {
        printf("a is 1¥n");
}
else {
        printf("a is neither 0 nor 1¥n");
}
```

例4　季節の表示　　ex4.c

```
#include <stdio.h>

int main(void) {
        int month;

        printf("enter 1 - 12:");
        scanf("%d", &month);
        switch(month) {
        case 3:
        case 4:
        case 5:
                printf("spring¥n");
                break;
        case 6:
        case 7:
        case 8:
                printf("summer¥n");
                break;
        case 9:
        case 10:
        case 11:
                printf("autumn¥n");
                break;
        case 12:
        case 1:
        case 2:
                printf("winter¥n");
                break;
        default:
                printf("out of range¥n");
                break;
        }
        return 0;
}
```

case 3: と case 4: の間に
break; がないので，下へ落ちていく

ここの break; で
3,4,5の処理が終了する

月の番号として不正な値が与えられたら
ここに来る

この break; はいまのところ不要だが，もし後に，
この後ろになにかを書き足したときのことを考え，
念のため break; を書いておくのがよい

練習問題

1．月と日を入力させ，その組み合わせがおかしい場合に，その旨を表示するプログラムを作成せよ．

- うるう年は考慮しなくて良い（2月は28日までとする）
- 月がおかしい場合に対処（13月10日など）
- 日が多すぎる場合に対処（11月31日など）