ソフトウェア工学

基本的なアルゴリズムとデータ構造

本日はC言語の復習を兼ねて，以下の項目について学習していきます.

（１）非ゼロは真でありゼロは偽である（p.29）

（２）論理演算子の短絡評価とド・モルガンの法則（p.33）

（３）配列の要素数の求め方（p.44）

（４）割付け記憶域期間と動的なオブジェクトの生成（pp.46-49）

（５）関数形式マクロ（pp.59-61）

（６）前置増分演算子と後置増分演算子（p.65）

（１）非ゼロは真でありゼロは偽である（p.29）

関係演算子や等価演算子による条件式の判定が真であれば int 型の 1 を返し，偽であれば int 型の 0 を返します．また，条件式に使用される値 0 は偽と判定され，0 でない全ての値は真とみなされます.

例　　if(a) printf(“ABC”);

a の値が 0 以外（1 でも 100 でも-2 でも）ならば「ABC」と表示されます.

（２）論理演算子の短絡評価とド・モルガンの法則（p.33）

１．論理演算子の短絡評価

(a)||演算子の左オペランドを評価した値が 1 であれば，右オペランドの評価は行われません.

例　　if(no < 10 || no > 99)

no < 10 が真の場合は，no > 99 の結果に係らず，条件式全体が真になります

(b)&&演算子の左オペランドを評価した値が 0 であれば，右オペランドの評価は行われません.

例　　if(no < 10 && no > 99)

no < 10 が偽の場合は，no > 99 の結果に係らず，条件式全体が偽になります

※この短絡評価を意識して，条件式の組み合わせを考えると式の評価が効率的になります.

２．ド・モルガンの法則

「“条件式の否定をとって，論理積・論理和を入れ替えた式”の否定」が，もとの条件式と同じになることをド・モルガンの法則と言います．この法則は一般的に示すと以下になります.

(a)x && y と!(!x || !y)は等しい

(b)x || y と!(!x && !y)は等しい

例　　no < 10 || no > 99　→　!(no >= 10 && no <= 99)

継続条件　　終了条件の否定

（３）配列の要素数の求め方（p.44）

配列の要素数は seizeof 演算子を使って求めることができます.

例　　int a[5] = {1, 2, 3, 4, 5};

```
int na = sizeof(a) / sizeof(a[0]);
```

※sizeof(a) / sizeof(int)とすると，int の部分は配列の宣言 int a[5]の型が変わるたびに変更することになるので避けます.

（４）割付け記憶域期間と動的なオブジェクトの生成（pp.46-49）

記憶域確保のための calloc 関数と malloc 関数，記憶域解放のための free 関数について学びます．これらの記憶域確保は一般にヒープと呼ばれる特殊な空き領域から記憶域を確保します．通常の変数や配列の宣言の場合はスタックと呼ばれる領域に確保されます．ヒープは使用するサイズとタイミングをアプリケーションで決めることができ，アプリケーションの実行中に使用するメモリサイズが変化させることが可能です．また，スタックの領域よりも大きいので，使用するメモリサイズが比較的大きい時に利用すると良いです.

例
```
int *x;
x = calloc(1, sizeof(int));     // x = malloc(sizeof(int));
*x = 57;
free(x);
```

例
```
int *x;
x = calloc(3, sizeof(int));     // x = malloc(3 * sizeof(int));
x[0] = 0; x[1] = 1; x[2] = 2;
free(x);
```

calloc 関数の使用方法を List2-3，List2-4，イメージ図を Fig.2-5，Fig.2-6 で確認してください．（pp.47-48）

演習：List2-4 の作成と実行（p.49）

※calloc で確保された領域は全てのビットが 0 で初期化されますが，malloc の場合は不定です.

（５）関数形式マクロ（pp.59-61）

マクロ処理に式を利用することができます．マクロ処理はマクロ名で指定した文字列をプリプロセッサによって置き換えることで実行されます．以下は 2 値を交換する関数形式マクロの例です.

例
```
#define swap(type, x, y) do{ type t = x; x =y; y = t; }while(0)
```

使用例
```
swap(int, a[i], a[n-i-1]);
```

※関数は呼び出しの手間があるため，関数形式マクロの方が効率良く実行されます.

演習：List2-7 の作成と実行（p.60）

（６）前置増分演算子と後置増分演算子（p.65）

前置増分演算子++a

式全体の評価が行われる「前」にオペランドの値がインクリメントされます.

a が 3 として，b=++a が実行されると a が 4 になってから，b に 4 が代入されます.

後置増分演算子 a++

式全体の評価が行われた「後」にオペランドの値がインクリメントされます.

a が 3 として，b=a++が実行されると b に 3 が代入されてから，a が 4 になります.

課題 1

calloc で要素 10 の int 型配列を生成し，100 から 189 の乱数を生成して格納します．格納したら，表示し，関数形式マクロの swap を使って逆順に並べ替え，再び表示します．乱数生成は p. 56 を参考にしてください.

課題 1 は続きがありますので，締め切りは後で提示します.