2013 年 5 月 16 日（木）実施

プログラムの制御構造

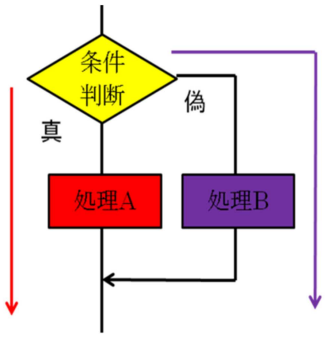
1960 年代後半にダイクストラが提唱した**構造化プログラミング**という考え方では，プログラムを記述する際には，**順次**，**選択**，**反復**という標準的な**制御構造**のみを用い，先ずプログラムの概略構造を設計し，その大まかな単位を段階的に詳細化して処理を記述していく。

順次構造

順次構造とは，プログラム中の文を処理していく順に記述したものである。これまで扱ったプログラムは，全て順次構造によって記述されたものであり，最も基本的な制御構造と言える。

プログラムの処理の流れを図示する手法の一つに**流れ図**がある。この流れ図で順次構造を表すと右図のようになる。(色矢印は処理の流れを補足)

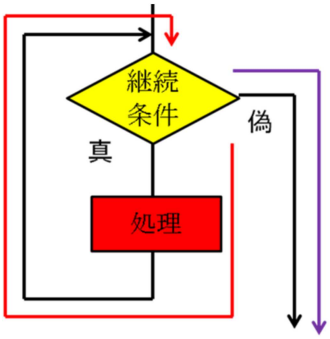
選択構造



選択構造とは，条件や式の値によってプログラムの処理の流れを分ける構造である。選択構造の基本は 2 分岐と呼ばれる構造で，この構造を流れ図で表すと右図のようになる。

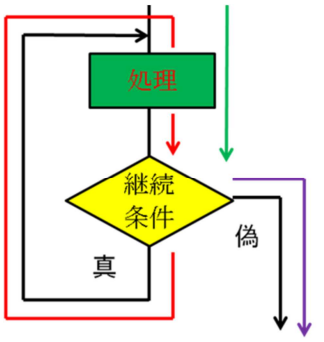
また，式の値によって，幾つもの異なる処理が必要なときには，**多分岐**という選択構造も利用可能である。

反復構造



反復構造とは，継続条件が満たされている間，定められた範囲内の文に記述された処理を繰り返して実行する構造である。(C 言語以外のプログラム言語では，終了条件が満たされない間，文を繰り返して実行する構造を持つものもある)

なお，反復構造には，右の流れ図で表される 2 種類の場合がある。



1）0 回以上の繰り返し（右上図）

先ず継続条件が判定され，真であれば定められた範囲内の文に記述された処理を実行する。始めから継続条件が満たされない場合には，文は全く実行されないため，0 回以上の繰り返しと呼ばれる。

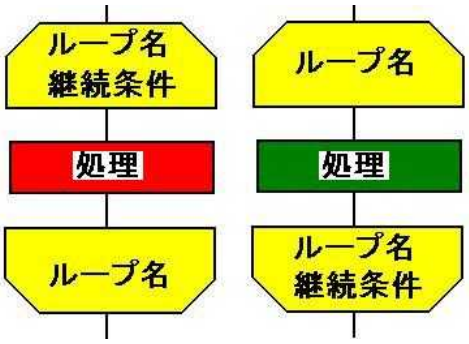
2）1 回以上の繰り返し（右下図）

先ず定められた範囲内の文に記述された処理が実行され，その後に継続条件が判定される。始めから継続条件が満たされない場合でも，最初の 1 回は定められた範囲内の文に記述された処理が実行されるため，1 回以上の繰り返しと呼ばれる。

＊ 反復構造に対する流れ図に表れる閉線図形をループと呼ぶことから，反復構造のプログラム構

造をループと称することがある。

複雑な処理は，順次構造，選択構造，反復構造の組み合わせで実現される。プログラムの構造は，最も大掴みにした概略構造で見ると順次構造となる。選択構造や反復構造を中間の概略構造と看做した場合，その詳細構造として，それぞれの構造の流れ図で『処理』と書かれた箇所に，選択構造や反復構造を埋め込んだ構造も可能である。そこで，反復構造の開始位置と終了位置とを『ループ端』によって明示し，構造を見易くした右のような流れ図も利用される。



2 分岐のプログラム

if 文

C 言語で 2 分岐のプログラムを実現するための文として，**if 文**が用意されている。if 文の構文は次のようになる。なお，真文または偽文に if 文を埋めて多分岐の構造を実現することができる。

```
if ( 制御式 ) 真文 [ else 偽文 ]
```

ここで，[]内は省略可能であり，省略時は制御式が偽のときは何もしない場合を表す。また，真文または偽文を{}で囲んだブロックとして，複数の文で構成することができる。

制御式

制御式には，大小関係等を判定する**条件式**（例 x > 0）と加減乗除等の**算術式**（例 x - 1）とがある。条件式では，条件が満たされる（真となる）場合には，値が 1 となり，条件が満たされない（偽となる）場合には，値が 0 となる。算術式では，値が 0 以外の場合，真の扱いとなり，値が 0 の場合，偽の扱いとなる。

条件式で用いられる**関係演算子**及び**等価演算子**を次に挙げる。

関係演算子	書式	意味
<	x < y	x が y より小さければ真，それ以外は偽
>	x > y	x が y より大きければ真，それ以外は偽
<=	x <= y	x が y より小さいか，両者が等しければ真，それ以外は偽
>=	x >= y	x が y より大きいか，両者が等しければ真，それ以外は偽

等価演算子	書式	意味
==	x == y	x と y とが等しければ真，それ以外は偽
!=	x != y	x と y とが等しくなければ真，それ以外は偽

また，複数の条件式を組み合わせるために用いられる**論理演算子**を次に挙げる。

論理演算子	書式	意味
&&	式１ && 式２	式１及び式２が共に真であれば真，それ以外は偽
||	式１ || 式２	式１または式２のどちらか一方が真であれば真（両者が真の場合も含む），それ以外は偽
!	!式	式が真であれば偽，式が偽であれば真

例題１（**if 文を用いた２分岐**）

次のソースプログラムをテキストエディタで入力して，prog5-1.c の名前を付けて保存する。それを翻訳・編集して実行形式のファイルを作成し，実行せよ。また，実行は３回行い，次の例にならった３通りの入力によって，プログラムの不備を確かめること。(演習１で修正する)

```
1) prog5-1.exe jimbo masato
2) prog5-1.exe jimbo
3) prog5-1.exe
```

```
/* prog5-1.c */
#include <stdio.h>

int main(int argc, char *argv[])
{

    printf("入力された姓は，%s です。¥n", argv[1]);

    if (argc < 2)
        printf("利用法： prog5-1.exe 姓 名¥n");
    else
        printf("入力された名は，%s です。¥n", argv[2]);

    return 0;
}
```

例題２（**多段の if 文を用いた場合分け**）

次のソースプログラムをテキストエディタで入力して，prog5-2.c の名前を付けて保存する。それを翻訳・編集して実行形式のファイルを作成し，次の様に６回，実行せよ。

```
1) prog5-2
2) prog5-2 +
3) prog5-2 -
4) prog5-2 *
5) prog5-2 /
6) prog5-2 %
```

```
/* prog5-2.c */
#include <stdio.h>
#include <stdlib.h>

void wa(int, int);
void sa(int, int);
void seki(int, int);
void shou(int, int);
void amari(int, int);

int main(int argc, char *argv[])
{
    int x,y;

    if (argc != 2)
    {
        printf("利用法：¥n prog5-2 +¥n prog5-2 -¥n prog5-2 *¥n prog5-2 /¥n");
        printf(" prog5-2 %%¥n");
        exit(1);
    }

    printf("1つ目の整数を入力してください： ");
    scanf("%d", &x);
    printf("2つ目の整数を入力してください： ");
    scanf("%d", &y);

    if (argv[1][0] == '+' && argv[1][1] == '¥0')
        wa(x, y);
    else if (argv[1][0] == '-' && argv[1][1] == '¥0')
        sa(x, y);
    else if (argv[1][0] == '*' && argv[1][1] == '¥0')
        seki(x, y);
    else if (argv[1][0] == '/' && argv[1][1] == '¥0')
        shou(x, y);
    else if (argv[1][0] == '%' && argv[1][1] == '¥0')
        amari(x, y);
    else
        printf("第1プログラム引数は, +, -, *, /, %%のいずれかです。¥n");

    return 0;
}

void wa(int a, int b)
{
    printf("%d + %d => %d¥n", a, b, a+b);
}

void sa(int a, int b)
{
    printf("%d - %d => %d¥n", a, b, a-b);
}

void seki(int a, int b)
{
    printf("%d * %d => %d¥n", a, b, a*b);
}
```

```
void shou(int a, int b)
{
    printf("%d / %d => %d¥n", a, b, a/b);
}

void amari(int a, int b)
{
    printf("%d %% %d => %d¥n", a, b, a%b);
}
```

【解説】

1. exit(1)は異常終了状態の形式を返して，プログラムを終了させるライブラリ関数で，利用するためには，stdlib.h を組み込んでおく必要がある。

2. if 文の偽文に if 文が埋め込まれた多段の if 文が用いられている。

3. 等価演算子(==)は論理演算子(&&)より優先度が高い。

4. argv[1][0]は第１プログラム引数の１文字目を指す。

5. argv[1][1] == '¥0'は第１プログラム引数が１文字目のみで文字列が終わるかどうかを判定している。('¥0'は文字列の終端)

6. Windows では prog5-2.exe と入力する代わりに，『.exe』を省略した prog5-2 を入力することで，プログラムの実行が可能である。

【注意】プログラム実行時に，prog5-2＊とした場合，UNIX 系の OS では，'*'は任意の文字列を表すワイルドカードとして扱われるため，積を呼び出すための第１プログラム引数を'x'とする等の工夫が必要である。

演習１

例題１のプログラムには不備がある。これを改良して，プログラム名の後に『姓』,『名』が共に入力された場合でなければ使用法を表示する，という仕様としたものが次のプログラムである。この空欄 1)　　，2)　　，3)　　を埋めてソースプログラムを完成させ，テキストエディタで入力して，ex5-1.c の名前を付けて保存する。それを翻訳・編集して実行形式のファイルを作成し，実行せよ。

```
/* ex5-1.c */
#include <stdio.h>

int main(int argc, char *argv[])
{
```

```
    if (argc < 1)     )
        printf("利用法： ex5-1.exe 姓 名¥n");
    else {
        printf("入力された姓は，%s です。¥n", argv[2)     ]);
        printf("入力された名は，%s です。¥n", argv[3)     ]);
    }

    return 0;
}
```

演習２（余裕のある人向け）

次の仕様のプログラムをテキストエディタで入力し，ex5-2.c の名前を付けて保存する。それを翻訳・編集して実行形式のファイルを作成し，実行せよ。(提出するファイルは ex5-2.c の完成版とする)

1. 「何歳ですか？」というガイドコメントに続いて，キーボードから入力した値を変数に格納し，その値が 20 以上かどうかを比較する。
2. 値が 20 未満であれば，「質問を終わります。」と表示する。
3. 値が 20 以上であれば，「喫煙者ですか？(Y or N)」というガイドコメントに続いて，キーボードから入力した値を変数に格納し，その値が'Y'，'N'，それ以外に場合分けする。
4. 値が'Y'の時，「回答者は喫煙者です。」と表示する。
5. 値が'N'の時，「回答者は非喫煙者です。」と表示する。
6. 値がそれ以外の時，「回答者が喫煙者か非喫煙者か不明です。」と表示する。

提出物：

1) 例題 1，2 の出力結果をコピーして貼り付けたテキストファイル res5.txt
2) 演習 1 のソースプログラムのファイル ex5-1.c の完成版