第4回

- C言語による初級プログラミング (2)
  - 前回の復習
  - Forによる繰り返しとIfによる分岐
  - Whileによる繰り返しとSwitchによる分岐
  - 配列
  - 配列を使ってプログラムを書いてみよう
  - フローチャートを作って考えると
  - プログラムの制御構造をPADで表現
  - PAD法の例

- **来週の予定**
  - **今日の予定が全て終わった場合:**

    windowsの希望があればやります。

    その希望が無い場合は、

    Cの二次元配列とunixのコマンド、シェル等などをやります。

前回の復習

- テキストエディターでソースプログラムを書く
  - 大文字・小文字、全角・半角に注意！
  - 変数はデータをしまう箱

```
main()
{
  printf("こんにちは¥n");
}
```

- コンパイル
  - コンパイルは　gcc hogehoge.c
  - メッセージが無ければコンパイルはOK

```
c2ja0rat(109)% gcc test.c
test.c: In function `main':
test.c:4: parse error before `)'
c2ja0rat(110)%
```

- 実行
  - 実行して問題ないか確認
  - 正常に動作しない時は要注意
    ex1.cで多く見られた例：

```
scanf("%d %d", &Data1, &Data2);
```

```
c2ja0rat(110)% gcc test.c
c2ja0rat(111)% ls
a.out          intro.txt      student.txt
PC             test.c
```

```
c2ja0rat(113)% a.out
こんにちは
```

- 結果の保存方法
  - 実行時に　**a.out > kekka.txt**　としてください。

Forによる繰り返しとIfによる分岐

```
# include <stdio.h>
# define N  5
main()
{
  int i, Data, Total=0;
  float Average;

  for(i=1; i<=N; i++){
    scanf("%d", &Data);
    Total += Data;
  }
  Average = (float)Total/N;
  printf("Number of Data = %d¥n", N);
  printf("Total = %d¥n",Total);
  printf("Average = %5.2f¥n", Average);
  if (Average >= 60) {
    printf("You win.¥n");
  }
  else{
    printf("You lose.¥n");
  }
}
```

標準的な関数を利用するとの宣言
N という定数を5に定義
これから始まる

整数型の変数のi,Data, Totalの定義、Totalの初期値を0
実数型の変数 Averageを定義

iの最初のを1として、iがN以下の間は次の{}を繰返す(*)
Dataを入力
現在のTotalの値にDataの値を加えて新たなTotalの値に
(Total = Total + Data と同じ)
Totalを実数型に変換してNで割ってAverageに代入
N(読み込む数)を表示
Total(総和)を表示
Average(平均)を5桁（小数点以下2桁）で表示
もし、Averageが60以上なら
“You win.”と表示

そうでなければ
“You lose.”と表示

終わり

ex2.c というファイル名で保存し
コンパイル、実行してみよう。

(*):i++はi=i+1と同じで、iを1づつ増加させる

Whileによる繰り返しとSwitchによる分岐

```
#include <stdio.h>

main()
{
 int i, Data, Total=0, n=0, c;
 float Average;

 while ( (c=scanf("%d", &Data))!=EOF){
  Total += Data;
  n++;
 }
 Average = (float)Total/n;
 printf("Number of Data = %d¥n", n);
 printf("Average = %5.2f¥n", Average);
 c= (int)(Average + 0.5);
 printf("4sha5nyuu = %d¥n", c);
 switch(c){
  case  0 ... 59: printf("D.¥n"); break;
  case 60 ... 74: printf("C.¥n"); break;
  case 75 ... 89: printf("B.¥n"); break;
  case 90 ... 100: printf("A.¥n"); break;
 }
}
```

ex3.c というファイル名で保存

キーボードからの入力の終わりは、ctrl-d を押す。

標準的な関数を利用するとの宣言
　ex3での扱うサンプル数を固定 (#define N 5)にしない
これから始まる

整数型の変数のi,Data,Total,n,cの定義、Total，cの初期値0
実数型の変数 Averageを定義

読み込んだﾃﾞｰﾀをcとし、cがEOF(ﾌｧｲﾙの最後)で無い間
　現在のTotalの値にDataの値を加えて新たなTotalの値に
　現在のnの値に1を加えて新たなnの値に
　(n = n + 1 と同じ)
Totalを実数型に変換してNで割ってAverageに代入
nを(読み込む数)を表示
Average(平均)を5桁(小数点以下2桁)で表示
Averageに0.5を加えて、整数型に変換して(切捨)cに代入
c(Averageを四捨五入したもの)を表示
もし、cが
　0から59の場合は、 Dと表示
　60から74の場合は、Cと表示
　75から90の場合は、Bと表示
　91から100の場合は、 Aと表示
終わり

注意：このswitch-caseはgccでの拡張で、範囲を“ … ”で指定している。しかし、他のcでは動作しない可能性がある。通常は “case 59: printf等” 一つの値だけ受付ける。

配列

- 同じデータ型の複数個の変数をまとめて定義
  - 例：５科目の点数を表す変数

    int data1, data2, data3, data4, data5

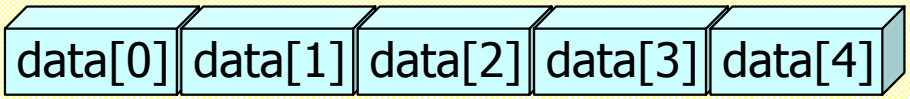
    ⇒書くのがメンドクサイ！

    ⇒まとめて定義しよう

    int data[5]　（型　配列名（要素数））

    ⇒data[0], data[1], data[2], data[3],data[4]
    の5つの変数

    ⇒配列の添字は0から「要素数-1」まで



5つの変数の箱を用意

  - 繰返しの記述が簡単になる。
    - 例：　for (i=0; i<5; ++1) Total+=data[i];
  - 初期化もできる。
    - 例：　int data[5] = { 60, 68, 70, 75, 66}

配列を使ってプログラムを書いてみよう

```
#include <stdio.h>

main()
{
  int i, data[10], Total=0, n=0, c;
  float Average;

  while (scanf("%d", &data[n])!=EOF) n++;
  for (i=0; i<n; i++) Total+=data[i];
  Average = (float)Total/n;
  printf("Number of Data = %d¥n", n);
  printf("Average = %5.2f¥n", Average);
  c= (int)(Average + 0.5);
  printf("4sha5nyuu = %d¥n", c);
  switch(c){
    case  0 ... 59: printf("D.¥n"); break;
    case 60 ... 74: printf("C.¥n"); break;
    case 75 ... 89: printf("B.¥n"); break;
    case 90 ... 100: printf("A.¥n"); break;
  }
}
```

標準的な関数を利用するとの宣言
　これから始まる

　整数型の変数のi,Data,Total,n,cの定義、Total，cの初期
　実数型の変数 Averageを定義

　読み込んだﾃﾞｰﾀをcとし、cがEOF(ﾌｧｲﾙの最後）で無い間
　　現在のTotalの値にDataの値を加えて新たなTotalの値
　　現在のnの値に1を加えて新たなnの値に
　　(n = n + 1 と同じ)
　Totalを実数型に変換してNで割ってAverageに代入
　n(読み込む数)を表示
　Average(平均)を5桁（小数点以下2桁）で表示
　Averageに0.5を加えて、整数型に変換して(切捨)cに代入
　c（Averageを四捨五入したもの）を表示
　もし、cが
　　0から59の場合は、 Dと表示
　　60から74の場合は、Cと表示
　　75から90の場合は、Bと表示
　　91から100の場合は、 Aと表示
終わり

ex4.c というファイル名で保存

キーボードからの入力の終わりは、
ctrl-d を押す。

注意：このswitch-caseはgccでの拡張で、範囲を“ ... ”で指定している。しかし、他のcでは動作しない可能性がある。
通常は “case 59: printf等” 一つの値だけ受付ける。

演算子（その2）

- **算術演算子**
  - +, -, *, /, %
  - ++:ｲﾝｸﾘﾒﾝﾄ(1を足す)
  - --:ﾃﾞｨｸﾘﾒﾝﾄ（1を引く）
  - ex. **for(i=1; i<=N; i++){**

- **代入演算子**
  - += 加算代入（足してから代入）
  - -= 減算代入（引いてから代入）
  - ex.　a+=10 （a = a+10 と同じ）
  - **ex. Total += Data;**
    (Total = Total + Data と同じ)

- **論理演算子**
  - == （等しいなら真、等しくないなら偽）
  - != （等しいなら偽、等しくないなら真）
  - & （and 演算）
  - | （or　演算）**ex. Total += Data;**
    (Total = Total + Data と同じ)

演算子はこの他多数あり

フローチャートを作って考えると

- フローチャート(flow chart, 流れ図)とは
  - ある問題を分析して、解決方法の処理の流れを書いたもの。
  - 四角を処理
    平行四辺形を入出力
    ひし型を判断
    で表す。
  - 線が自由にひけるため、流れが不明確で分りにくくなりやすい。
    ⇒スパゲッティプログラム（通称）

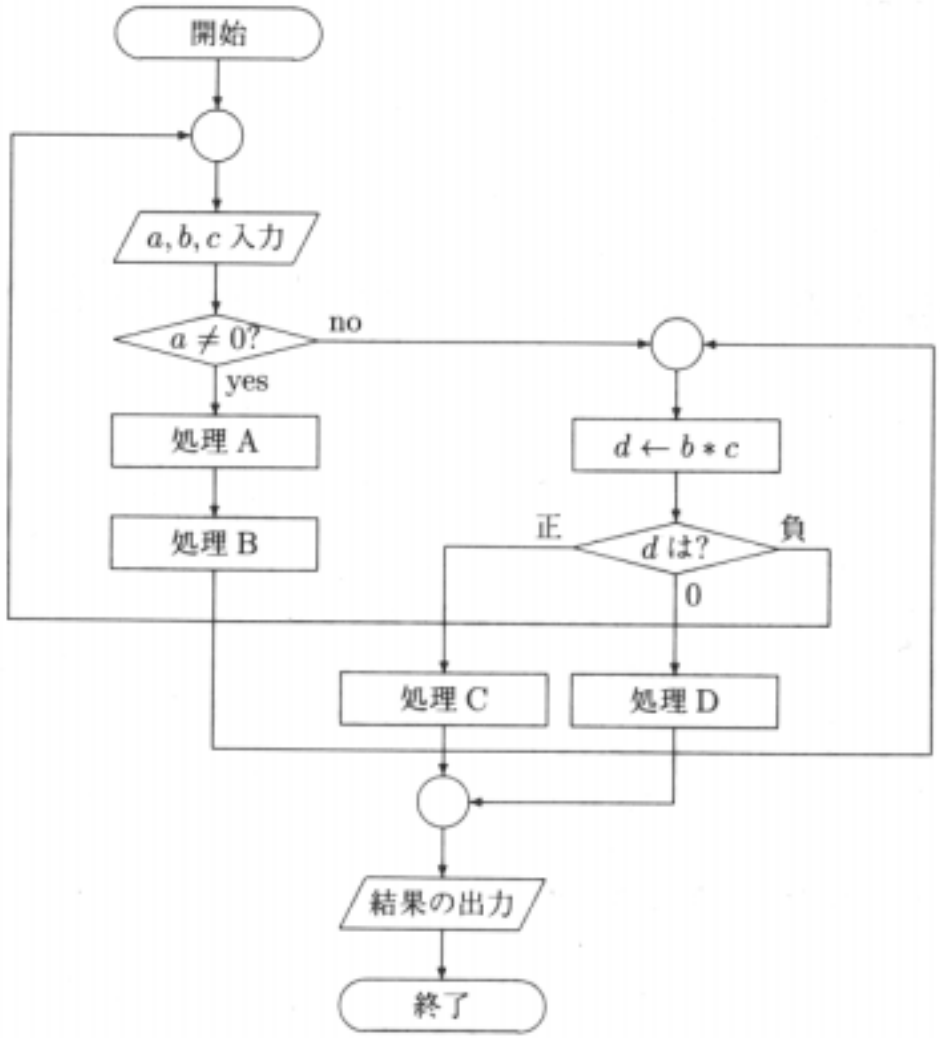
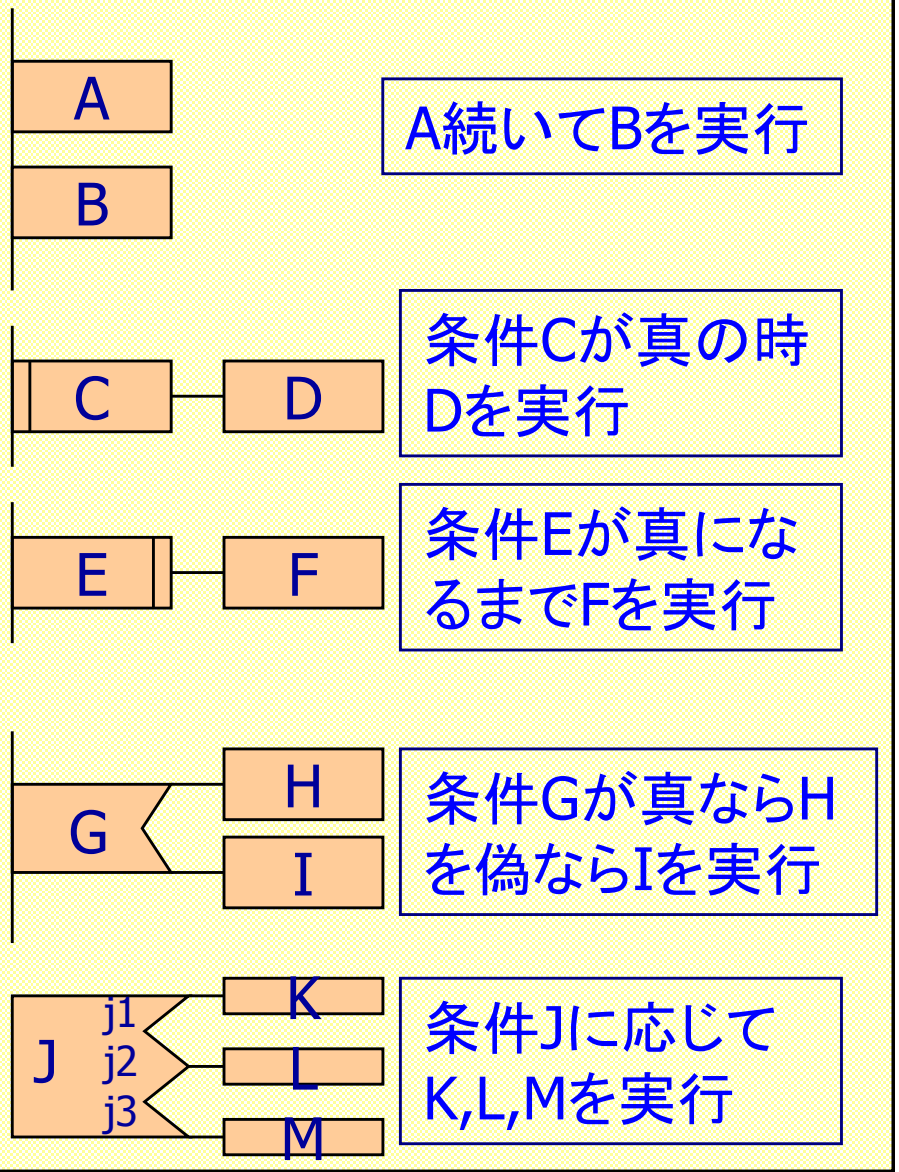


図 3.1　流れ図は分りにくいということが分る例

Cによる情報処理入門、p.18　昭晃堂

プログラムの制御構造をPADで表現

- PAD（Problem Analysis Diagram）の基本形
  - 連結
    - 並べた順番に逐次処理を行う
  - 反復
    - 前判定型繰返し
      while（式）文； for () 文:
    - 後判定型繰返し
      do 文 while（式）文;
  - 選択
    - if () else 型
    - case 型



PAD法の例

- 二次方程式($ax^2 + bx + c = 0$)を解く
  - 係数と定数の変数として、a, b, c
    解の変数として、xr1, xr2, xi1, xi2　を実数で定義
  - 二次式？、一次式？、不定？、解なし？
    の条件を整理する。
  - それぞれの場合を考えて、解を求めて表示する。

レポート

- 二次方程式を解くプログラムを、例にしめしたPAD図に基づいて作成し動作させる。

- プログラムと実行例をメールで送付する。
  - 注意：プログラム・実行例とも書類添付でなく、本文に挿入すること。
  - メール送付先：c2ja0rat@cs.ecip.tohoku.ac.jp
  - 題名（Subject)は、学籍番号（半角）としてください。

- 締切り：５月２１日昼まで