

1 プログラムの作成

- ① 何をどのように処理させたいのか、どのようなデータを入力し、どのような結果を出力させるのか問題を明確にする。
- ② 問題の内容どおりに処理させるための手順を考える。(フローチャートの作成) ～アルゴリズム(算法) の作成
- ③ 論理的に誤りがなければ、文法にしたがってプログラムを作成する（もしくは人に頼む）。（文法上の誤りは、実行させると表示されるのでそれを修正する。論理的な誤りは、入力データ出力結果から判断しなければならない。）

※プログラムが得意なところ、不得意なところを理解しておくことが大事

2 配列を使うということ

プログラムは、問題設定をいかに“配列”に落とし込むかという作業になる。つまり、プログラムは人間のような抽象的な思考はできないため、記憶や論理を配列を使って表現する必要がある。

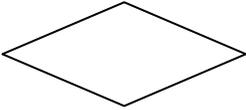
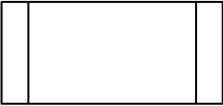
例えば $A(1), A(2), \dots, A(N)$ という配列を大きさ順に並べる時は、 $N(1)=3, N(2)=6, N(3)=5 \dots$ などと、 i 番目の $A(i)$ の大きさの順番の配列 $N(i)$ を用意したり、 $N2(1)=2, N2(2)=3, N2(3)=1, \dots$ などと、 i 番目に大きい A の番号の配列 $N2(i)$ を用意する。

自由に配列をたくさん作って有効に活用できるかがキーとなる。

※ただし、C言語の配列は、 $A[0], A[1], \dots$ とゼロからはじめることに注意が必要である。

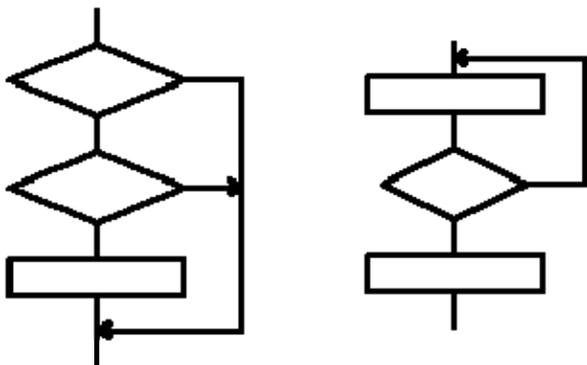
3 フローチャートの記号

フローチャートの記号は JIS 規格で決められている。

記号	名前	機能	Fortran	C
	端子	フローチャートのはじめと終わりを表す	Program test stop end	int main(...){ return 0; }
	処理	計算、代入などの処理を表す	A=B*C	a=b*c;
	判断	条件によって分岐する	if (A(1).gt.A(2)) else endif	if(a[0]>a[1]){... }else{... }
	ループ	繰り返しの始まりと終わり	do i = 1,N enddo	for(i=0;i<N;i++){ } while(...){ } do{ }while(...)
	入力	キーボードからの入力	read(5,*)	scanf("%lf",&a);
	表示	ディスプレイへの表示	write(6,*)	printf("%lf",a);
	定義済 み処理	あらかじめ定義された処理のまとめ(関数, サブルーチン)	Subroutine test (a, b, c)	double test (double a, double b){ ... return c;}
	入出力	ファイルへの出入力	open(1, file='test.dat') read(1,*) write(1,*)	fp=fopen("in.dat","r"); fscanf(fp,"%lf",&a); fclose(fp); fp=fopen("out.dat","w"); fprintf(fp,"%lf\n",a); fclose(fp);

4 フローチャートのルール

- ・ 処理の流れは原則上から下へ、左から右へ、それに逆行する場合には矢印にする
- ・ 線が立体交差しないようにする
- ・ 二つ以上の線を集めてひとつにして良い。



(例題：直線型フローチャート)

長方形の2辺の長さから面積を求めるプログラムを作成せよ。

① 2辺の長さ(X, Y)から面積を次の式で求める。

$$\text{面積} = X \times Y$$

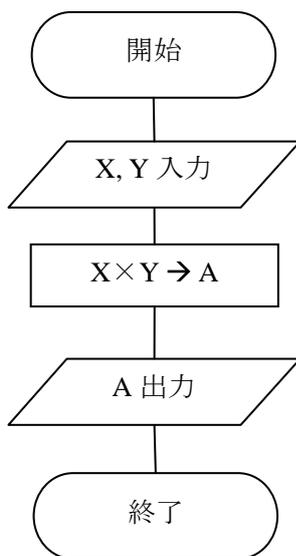
② 入力データ … 長方形の縦の長さとの横の長さ。

出力結果 … 面積。

③ 手順

- (1) 2つのデータを入力する。
- (2) 面積を計算する。
- (3) 計算の結果を表示する。

フローチャート



プログラム(FORTRAN)

```

Program rect
write(6,*)'X, Y?'
read(5,*)X, Y
A=X*Y
write(6,*)A
stop
end
    
```

プログラム(C)

```

#include <stdio.h>
int main(void){
    double x, y, a;
    printf("x=");
    scanf("%lf", &x);
    printf("y=");
    scanf("%lf", &y);
    a = x * y;
    printf("A=%lf\n", a);
    return 0;
}
    
```

フローチャートの上から下へ処理の順序が一直線に進む（流れる）形の処理である。基本的には、①データ入力（データの記憶） → ②計算 → ③結果出力、の順である。

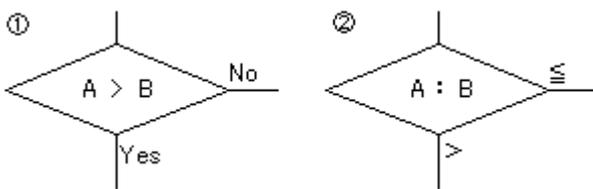
5 フローチャートの作成

5.1 分岐型

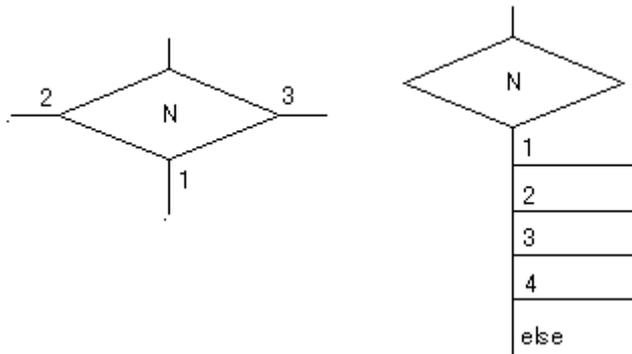
条件を判定した結果によって2つ以上の方向に処理の流れが分かれるものである。条件は、①等号、不等号の式で表し、成立、不成立によって流れを変える。②比較する値（式）を記述し、比較結果によって流れを変える。の2通りが考えられる。

フローチャート内の表記の方法が違うが、①、②とも同じフローチャート記号である。

A の値の方が B より大きければ、下の流れ線へ進む。逆に B の値の方が A より大きければ、右の流れ線へ進む。

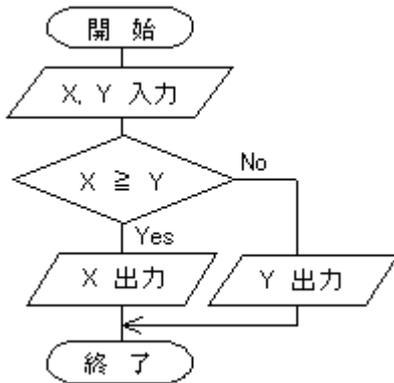


分岐が3つ、4つになる場合は、以下のように書かれる。



例題 2つの値 X,Y を入力して大きい方を出力する。X と Y が同じときは X を出力する。フローチャートを作成せよ。

フローチャート



プログラム(FORTRAN)

```

Program bunki
write(6,*)'X, Y?'
read(5,*)X, Y
if (X.ge.Y) then
write(6,*)X
else
write(6,*)Y
endif
stop
end
  
```

プログラム(C)

```

#include <stdio.h>
int main(void){
    double x, y;
    printf("x=");
    scanf("%lf", &x);
    printf("y=");
    scanf("%lf", &y);
    if(x>=y){
        printf("%lf¥n", x);
    }else{
        printf("%lf¥n", y);
    }
    return 0;
}
  
```

5.2 複雑なフローチャート (繰り返し型)

処理のある範囲を繰り返す処理である。

下図フローチャートでは、[i の値出力] の処理を 10 回繰り返す。

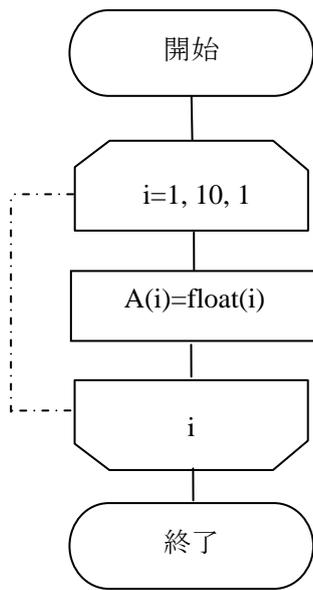
「ループ端」に書かれている数値は次のような意味である。i = 1, 10, 1

変数 = 初期値, 終値, 増分値

最初に、変数(i)に初期値(1)を記憶して繰り返し範囲を処理する。二度目以降は変数(i)に増分値(1)を加算して終値(10)以下であれば繰り返し範囲を処理する。これを終値を超えるまで繰り返す。

ここでは、「ループ端」の組 (ペア) がわかりやすいように記号を点線でつないで示しているが、本来はなくても良い。

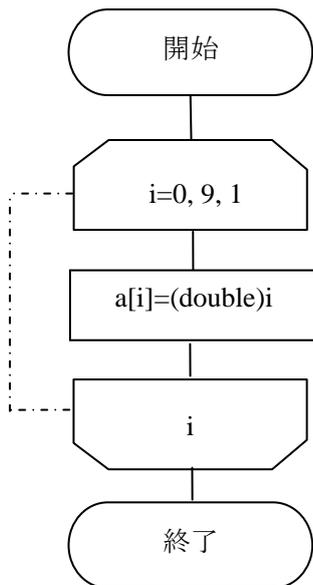
フローチャート(FORTRAN)



プログラム(FORTRAN)

```
Program loop
dimension a(10)
do i=1,10
a(i)=float(i)
enddo
stop
end
```

フローチャート(C)

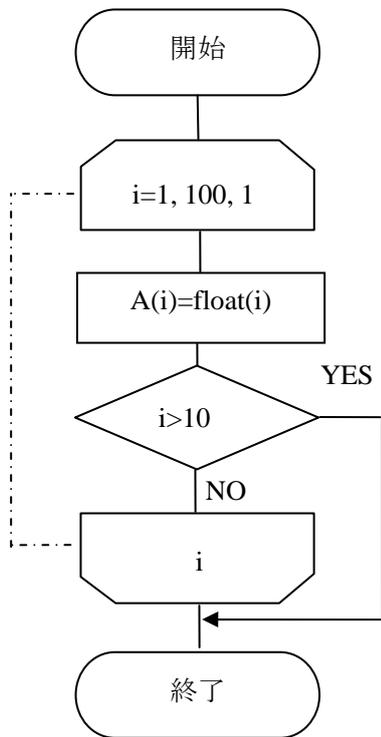


プログラム(C)

```
#include <stdio.h>
#define N 10
int main(void){
    int i;
    double a[N];
    for(i=0;i<N;i++){
        a[i] = (double)i;
    }
    return 0;
}
```

繰り返しの回数を分岐を利用して制御する場合もある。

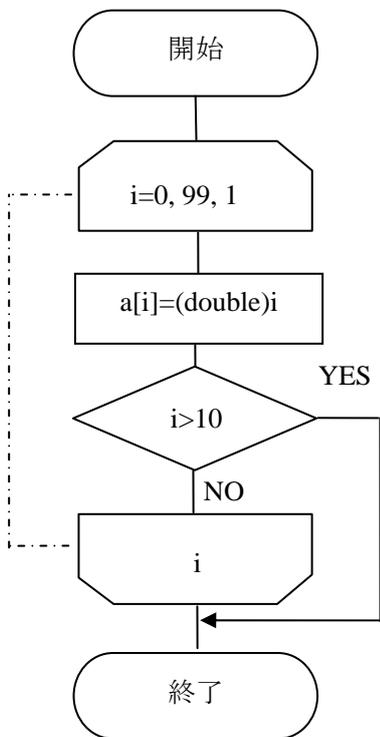
フローチャート



プログラム(FORTRAN)

```
Program loop
dimension a(100)
do i=1,100
a(i)=float(i)
if (i.gt.10) exit
enddo
stop
end
```

フローチャート(C)

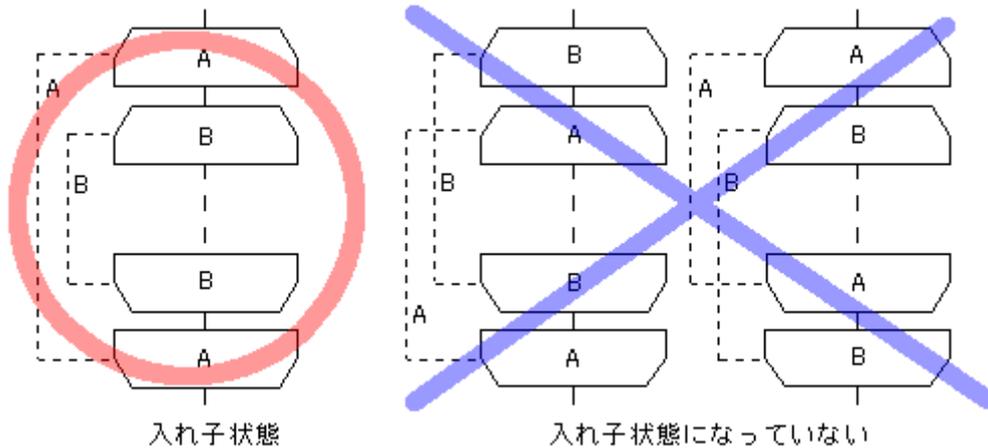


プログラム(C)

```
#include <stdio.h>
#define N 100
int main(void){
    int i;
    double a[N];
    for(i=0;i<N;i++){
        a[i]=(double)i;
        if(i>10){
            break;
        }
    }
    return 0;
}
```

※二重ループ

繰り返しを入れ子にしたループである。下図のように、「ループ端」の各ペアが交差してはいけない。

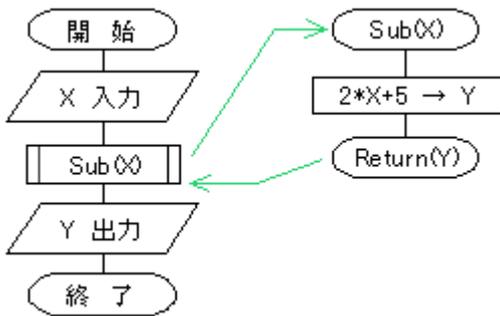


5.3 サブルーチン

処理が複雑になってくると、ひとつのフローチャートでは表しにくくなる。そこで、まとまった処理を別のフローチャートとして表し、主（メイン）となるフローチャートから呼び出して利用できるようにすると都合がよい。

例題 式 $2x+5$ を計算する処理を定義済み処理として作成する。計算に必要なデータ x の入力と結果の出力はメイン側で行うものとする。フローチャートを作成せよ。

フローチャート



プログラム(FORTRAN)

```

Program sub_routine
  read(5,*)X
  Y=sub(X)
  write(6,*)Y
  stop
end

real function sub(X)
  sub=2*X+5
end function
    
```

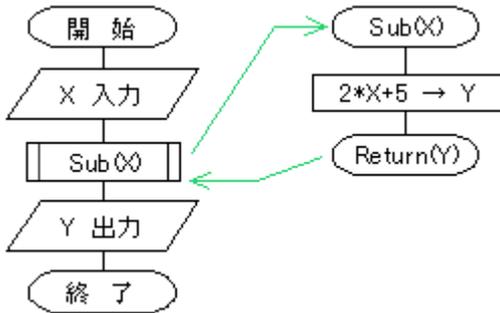
プログラム(C)

```

#include <stdio.h>
double func(double x);
int main(void){
  double x, y;
  printf("data=");
  scanf("%lf", &x);
  y = func(x);
  printf("result=%lf\n", y);
  return 0;
}
double func(double x){
  double y;
  y = 2.0 * x + 5.0;
  return y;
}
    
```

配列の受け渡し

フローチャート



プログラム(FORTRAN)

```

Program sub_routine
parameter (nn=2)
dimension x(2), y(2)
read(5,*)x(1),x(2)
call=sub(nn, x, y)
write(6,*)y(1),y(2)
stop
end

subroutine sub(nn, x, y)
dimension x(nn), y(nn)
y(1)=2*x(1)+5
y(2)=2*x(2)+5
return
end
  
```

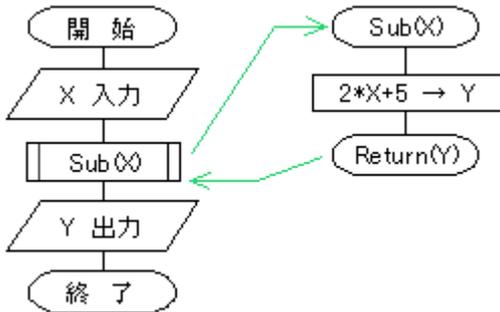
プログラム(C)

```

#include <stdio.h>
void sub(double x[],double y[]);
#define N 2
int main(void){
    double x[N], y[N];
    printf("data=");
    scanf("%lf %lf",&x[0],&x[1]);
    sub(x,y);
    printf("result=%lf %lf\n",y[0],
y[1]);
    return 0;
}
void sub(double x[],double y[]){
    y[0]=2.0*x[0]+5.0;
    y[1]=2.0*x[1]+5.0;
    return;
}
  
```

サブルーチン形式で書いたらどうなる？ポインタ???

フローチャート



プログラム(FORTRAN)

```

Program sub_routine
read(5,*)x
call sub(x, y)
write(6,*)y
stop
end

subroutine sub(x,y)
y=2*x+5
return
end
  
```

プログラム(C) 動的確保

```

#include <stdio.h>
#include<stdlib.h>
void sub(double x,double *py);
int main(void){
    double x,*py;
    py=(double*)malloc(sizeof(double));
    printf("data=");
    scanf("%lf", &x);
    sub(x,py);
    printf("result=%lf¥n", *py);
    free(py);
    return 0;
}
void sub(double x,double *py){
    *py = 2.0 * x + 5.0;
    return;
}
  
```

プログラム(C) 簡易版

```

#include <stdio.h>
void sub(double x,double *py);
int main(void){
    double x,*py,y;
    py=&y;
    printf("data=");
    scanf("%lf", &x);
    sub(x,py);
    printf("result=%lf¥n", *py);
    return 0;
}
void sub(double x,double *py){
    *py = 2.0 * x + 5.0;
    return;
}
  
```

C言語のよくやるバグ

1. ;がないケアレスミス
2. 全角で文字を書く → 極力全角文字はつかわない!