配列の宣言

配列を使用するためには変数宣言と同様に配列の宣言を行う

- **配列の名前**
- **配列の要素の型**
- **配列の添字の範囲**

配列の宣言と参照(Pascal)

宣言文

var **配列名**: array [**添字範囲**] of **要素の型**

添字範囲　[**最小添字**..**最大添字**]

二次元配列　[**最小**..**最大**, **最小**..**最大**]

要素の参照

配列名[**添字**(**整数値**)]

配列名[**添字**(**整数値**), **添え字**(**整数値**)]

配列の宣言と参照(Pascal)

例

```
var a: array [1..100] of integer;
  a[1], a[2], a[3], …, a[100]

var x: array [0..31, 0..31] of real;
  x[0,0], x[0,1], x[0, 2], …, x[0,31],
  x[1,0], x[1,1], x[1, 2], …, x[1,31],
   …,
  x[31,0], x[31,1], x[31,2], …, x[31,31]
```

配列の宣言と参照(FORTRAN)

宣言文

要素の型 配列名

DIMENSION **配列名**(**添字範囲**)

添字範囲　(**最小添字** : **最大添字**)

二次元配列　(**最小** : **最大**,　**最小** : **最大**)

要素の参照

配列名(**添字**(**整数値**))

配列名(**添字**(**整数値**), **添え字**(**整数値**))

配列の宣言と参照(FORTRAN)

例

```
INTERGER A
DIMENSION A(100)
  a[1], a[2], a[3], …, a[100]
DIMENSION X (0:31, 0:31)
  X(0,0), X(0,1), X(0, 2), …,X(0,31),
  X(1,0), X(1,1), X(1, 2), …, X(1,31),
   …,
  X(31,0), X(31,1),X(31,2), …,X(31,31)
```

配列の宣言と参照(c言語)

宣言文

要素の型 配列名[**要素の数**]

添字範囲　[0] **から**[**要素の数**–1]

二次元配列　[**要素の数**][**要素の数**]

要素の参照

配列名[**添字**(**整数値**)]

配列名[**添字**(**整数値**)][**添え字**(**整数値**)]

配列の宣言と参照(c言語)

例

```
int a[100];
  a[0], a[1], a[2], …, a[99]

double x[32][32];
  x[0][0], x[0][1], x[0][2], …, x[0][31],
  x[1][0], x[1][1], x[1][2], …, x[1][31],
   …,
  x[31][0], x[31][1], x[31][2], …, x[31][31]
```

文字データの扱い方

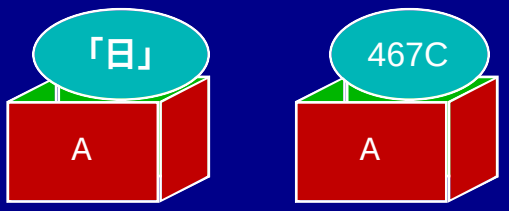
文字型変数

文字列

文字型変数の配列

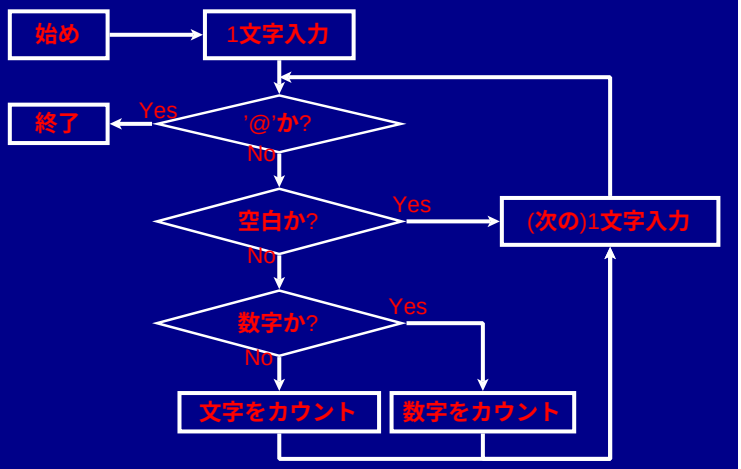
文字型変数

ASCII符号化文字集合(127文字)
→ 整数型/8ビット(1バイト)変数



例題

最後が@で終わる入力データの中の「空白」を除いて、「数字」とその他の「文字」との個数を計算しその比率を求めよ



文字変数(Pascal)

変数宣言 var 変数名 : char

```
var ch: char;
```

入力 read(文字型変数)

```
read(ch);
```

出力 write(文字型変数) ;
writeln(文字型変数);

```
write(ch);
writeln(ch);
```

文字変数(Pascal)

定数定義 const 文字定数 = '文字'

```
const eod = '@';
```

代入 文字型変数 := '文字'

```
ch := '9'
```

比較 整数として比較できる

```
if  ch <> '0' then begin … end;
```

```
program moji(input,output);
const blank=' '; eod='@';
var ch:char; dcnt, ccnt:integer;
begin
  dcnt=0; ccnt=0;
  read(ch);
  while ch <> eod do
    begin
      if ch <> blank then
        if ('0'<=ch) and (ch <= '9') then
          dcnt := dcnt + 1
        else
          ccnt := ccnt + 1;
      read(ch)
    end;
  writeln(dcnt/ccnt)
  end.
```

文字変数(FORTRAN)

変数宣言 CHARACTER 変数名

```
CHARACTER CH
```

入力 READ(n,*) 文字型変数

```
READ(5, *) CH
```

出力 WRITE(n, m) 文字型変数

```
WRITE(6, *) CH
```

文字変数(FORTRAN)

定数定義 PARAMETER文で行う

```
PARAMETER(CH='A')
```

代入 文字型変数 = '文字'

```
CH = '9'
```

比較 整数として比較できる

```
IF ((CH.GE.'0' ).AND.(CH.LE.'9')) THEN
…
END IF
```

```
      PROGRAM MOJI
      PARAMETER(BLANK=' ', EOD='@')
      CHARACTER CH
      INTEGER DCNT, CCNT, I
      DCNT=0
      CCNT=0
      DO 10 I=1, 1000
        READ(5,*)CH
        IF(CH.EQ.EOD) THEN
          WRITE(6,*)DFLOAT(DCNT)/DFLOAT(CCNT)
          STOP
        END IF
        IF(('0'.GE.CH).AND.(CH.GE.'9')) THEN
          DCNT=DCNT+1
        ELSE
          CCNT=CCNT+1
        END IF
   10 CONTINUE
      STOP
      END
```

文字変数(c言語)

変数宣言 char **変数名**;/ int **変数名**;

char ch; int c;

入力　　ライブラリ関数を使用

scanf("%c", &ch);

c = getchar();

出力　　ライブラリ関数を使用

printf("%c", ch);

putchar(c);

文字変数(c言語)

定数定義　マクロ命令を用いる

#define EOD '@'

代入　　文字型変数 = '文字'

整数型変数 = '文字';も可

ch = '9';

比較　　整数として比較できる

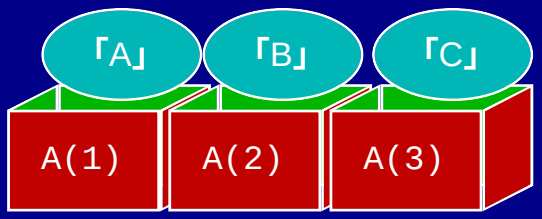
while (ch != EOD) {putchar(ch);}

```
#include <stdio.h>
#define BLANK  ' '
#define EOD    '@'
int main(void) {
  int ch, dcnt, ccnt;
  dcnt=ccnt=0;
  while ((ch = getchar()) != EOF) {
    if (ch != BLANK) {
      if(('0'<=ch)&&(ch<='9'){dcnt++;}
      else{ccnt++;};
    }
  }
  printf("%lf\n",(double)dcnt/(double)ccnt);
  return 0;
}
```

文字列

文字変数の配列

A(1)='A', A(2)='B', A(3)='C'



文字列(Pascal)

文字変数の配列

var c : array [1..25] of char

25文字収納できる文字変数の配列

var s : packed array[1..25] of char

25文字までの文字列を収納する文字変数の配列

文字列(Pascal)

var c : array [1..25] of char;

s: packed array[1..25] of char;

c[1] := 'A'; s[1] := 'A';

s := 'ABCDEFG' ;

c := 'ABCDEFG' ;

文字列(FORTRAN)

CHARACTER*数字 変数名

CHARACTER*8 STR

STR**は8文字収納可能な文字列変数**

CHARACTER*8 STRM(3)

STRM(1), STRM(2), STRM(3)**がそれぞれ8文字収納可能な文字列変数**

文字列(FORTRAN)

CHARACTER*3 S

STRM(1), STRM(2), STRM(3)**がそれぞれ**8**文字収納可能な文字列変数**

S='A' 「A　」
S='ABCD' 「ABC」
S(2:3) = 'ABC' 「 AB」

文字列(c言語)

char **変数名[文字数]**;

char s[256];

256**文字収容可能な文字変数の配列**
s[0]**から**s[255]**までに文字は収納され津**
s**は配列の始まる最初のアドレス** (&s[0])

文字列とポインタ(c言語)

char s1[8];
char *s2;

~~s1= "ABC" ;~~	strcpy(s1, "ABC")
s2 = "ABC";	~~strcpy(s2, "ABC")~~
s1[0] = 'A';	~~s1[0] = "A";~~
~~s2[0] = 'A';~~	~~s2[0] = "A";~~

[例題]

点数を読み込んでその平均値を出力するプログラム
ただし、データの個数は既にわかっているものとする

例題(Pascal)

```
program average(input, output);
const n = 10;
var x: array [1..n] of integer;
    i, sum: interger;
    a: real;
begin
  for i:= 1 to n do read(x[i]);
  sum:=0;
  for i:= 1 to n do sum:=sum+x[i];
  a:= sum/n;
  writeln(a)
end.
```

例題(c言語)

```
#include <stdio.h>
#define N 10
int main(void){
  int x[N];
  int i, sum;
  double a;
  for(i=0;i<N;i++)scanf("%d",&x[i]);
  sum=0;
  for(i=0;i<N;i++) sum+=x[i];
  a=(double) sum/ (double) N;
  printf("%lf\n", a);
  return 0;
}
```

例題(FORTRAN)

```
      PROGRAM AVERAG
      PARAMETER(N=10)
      INTEGER I, X, SUM
      DOUBLE PRECISION A
      DIMENSION X(1:N)
      READ(5, *) (X(I), I=1, N)
      SUM=0
      DO 10 I=1, N
   10 SUM=SUM+X(I)
      A=DFLOAT(SUM)/DFLOAT(N)
      WRITE(6,*) A
      STOP
      END
```

[例題]

点数を読み込んでその最大値を出力するプログラム
ただし、データの個数は既にわかっているものとする

例題(Pascal)

```
program average(input, output);
const n = 10;
var x: array [1..n] of integer;
    i, max: interger;
begin
  for i:= 1 to n do read(x[i]);
  max:=x[1];
  for i:= 2 to n do
    if x[i] > max then max:=x[i];
  writeln(max)
end.
```

例題(c言語)

```
#include <stdio.h>
#define N 10
int main(void){
  int x[N];
  int i, max;
  for(i=0;i<N;i++)scanf("%d",&x[i]);
  max=x[0];
  for(i=1;i<N;i++)
    if(x[i]>max) max=x[i];
  printf("%d\n", max);
  return 0;
}
```

例題(FORTRAN)

```
      PROGRAM AVERAG
      PARAMETER(N=10)
      INTEGER I, X, MAXX
      DIMENSION X(1:N)
      READ(5, *) (X(I), I=1, N)
      MAXX=X(1)
      DO 10 I=2, N
        IF (X(I).GT.MAXX) MAXX=X(I)
   10 CONTINUE
      WRITE(6,*) MAXX
      STOP
      END
```