2007/09/17

xDNCL 言語マニュアル

xDNCL は大学入試センターの「情報関係基礎」で用いられている試験用手順記述標準言語 DNCL に準拠しており、一部拡張したものである。

1 型宣言

- 変数を使用する場合は、前もって変数の型を宣言しなければならない。
- 使用できる変数の型は「整数」「実数」「文字列」の 3 種類である。
- 変数名は半角英字から始まり、2 文字目からは半角英数字のみが使用できる。
- 整数型の定数は、小数点を含めてはいけない (例: 12, –4)。
- 実数型の定数は、小数点を含めなければならない (例: 12.0, –4.0)。
- 整数型の変数の初期値は 0, 実数型の場合は 0.0, 文字列型の場合は NULL(空の文字列) である。

```
整数 ≪変数≫ , ≪変数≫ , ... , ≪変数≫
実数 ≪変数≫ , ≪変数≫ , ... , ≪変数≫
文字列 ≪変数≫ , ≪変数≫ , ... , ≪変数≫
```

使用例

```
整数 i, j, k  /* 変数名 i, j, k は整数型の変数であると宣言 */
実数 x, y     /* 変数名 x, y  は実数型の変数であると宣言    */
文字列 str    /* 変数名 str は文字列型の変数であると宣言    */
```

2 配列

整数型 count, 実数型 pos という変数名の配列を宣言する例

```
整数 count[5]  /* count[0]～count[5] までの 6 つの変数領域を確保 */
実数 pos[10]   /* pos[0]～pos[10] まで 11 個の変数領域を確保      */
```

2 次元以上の配列を宣言する例

```
整数 case[8, 8]        /* 9 × 9 の変数領域を確保する     */
実数 real[2, 3, 5]     /* 3 × 4 × 6 の変数領域を確保する */
```

3　演算子

3.1　算術演算子

下記の算術演算子が利用できる。

演算子	意味	例	式の値
＋, +	加算	8 + 3	8 に定数 3 を加えた値 (=11)
－, –	減算	x – 2	変数 x の値から 2 を引いた値
×, *	積算	y * 1.5	変数 y の値を 1.5 倍した値
÷, /	除算	z / 2	変数 z の値を 2 で割った値
%, %	剰余	z % 5	変数 z の値を 5 で割った余りの値

3.2　比較演算子

下記の比較演算子が利用できる。

演算子	意味	使用例	式の値
＝, =	等しい	x ＝ 0	x が 0 ならば真, それ以外ならば偽
＞, >	より大きい	y ＞ 5	y が 5 より大きければ真, 以下なら偽
≧, >=	以上	y ≧ 5	y が 5 以上ならば真, ちいさければ偽
＜, <	より小さい	z ＜ 1.2	z が 1.2 より小さければ真, 以上なら偽
≦, <=	以下	z ≦ 1.2	z が 1.2 以下ならば真, 大きければ偽
≠, !=	等しくない	z ≠ 6	z が 6 以外ならば真, 同じであれば偽

3.3　論理演算子

下記の論理演算子が利用できる。

演算子	意味	使用例	式の値
かつ	積集合	a ≧ 0 かつ a ≦ 10	a が 0 以上 かつ 10 以下ならば真, それ以外は偽
または	和集合	b ＜ 0 または b ＞ 100	b が 0 未満 または 100 より大きければ真, それ以外は偽
でない	否定	c = 5 でない	c が 5 でないなら真, 5 ならば偽

3.4　演算結果のデータ型について

　演算結果のデータ型は演算対象のデータ型によって決まる。
なお、演算前に演算結果のデータ型に変換してから、演算が行われる。

演算例	演算結果のデータ型
整数 ＋ 整数	整数
実数 ＋ 整数	実数
実数 ＋ 実数	実数
文字列 ＋ 整数	文字列として、文字列結合される
文字列 ＋ 実数	文字列として、文字列結合される
文字列 ＋ 文字列	文字列として、文字列結合される

4　コメント文 (注釈)

```
/*　《コメント》　*/
```

コメント文として書かれた文字列は、プログラム実行時には無視される。

5　出力文

5.1　改行あり出力

≪出力文≫ を表示する

≪出力文≫で指定された式や文字列定数をコンソール画面に表示し、その後改行する。
文字列定数とは **「　」** または **"　"** で囲んだ文字を示している。

5.2　改行なし出力

≪出力文≫ を改行なしで表示する

≪出力文≫で指定された式や文字列定数をコンソール画面に表示し、その後改行をしない。
文字列定数とは **「　」** または **"　"** で囲んだ文字を示している。

5.3　複数の変数・文字列の出力

　変数や文字列を一緒に出力する場合、それらを「**と**」で結び列挙する。

使用例

「答えは」 と ans と 「です」 を表示する

変数 ans が 50 の場合の出力例

答えは 50 です

6　代入文

6.1　代入

```
≪変数≫ ← ≪式≫
```

≪式≫に書かれた定数や式の演算結果を≪変数≫に指定された変数へ代入する。
なお、右辺の≪式≫の値の型のいかんに関わらず、代入によって、
左辺の≪変数≫のデータ型に自動変換される。
≪式≫の演算結果が実数型で、左辺の≪変数≫が整数型の場合、小数点以下は切り捨てられる。

6.2　入力

```
≪変数≫ ← input()
```

input() で キーボードからの入力が文字列として代入される。
左辺の≪変数≫のデータ型に合わせて型変換が行われる。

7　条件分岐

7.1　1 分岐の条件文 (if〜then 文)

```
もし ≪条件式≫ ならば
  | ＜処理＞
を実行する
```

≪条件式≫が成立した場合、＜処理＞を実行する。
成立しない場合は「を実行する」の次の行へ進む。

使用例

```
もし x = 10 ならば
  | 「条件が成立しました」を表示する
を実行する
「終了」を表示する
```

変数 x が 10 の場合の出力例

```
条件が成立しました
終了
```

変数 x が 5 の場合の出力例

```
終了
```

7.2　2 分岐の条件文 (if～then～else 文)

```
もし ≪条件式≫ ならば
   | ＜処理１＞
を実行し，そうでなければ
   | ＜処理２＞
を実行する
```

≪条件式≫が成立した場合、＜処理 1 ＞を実行し、
成立しない場合は＜処理 2 ＞を実行する。

使用例

```
もし x = 20 ならば
   | 「条件が成立しました」を表示する
を実行し，そうでなければ
   | 「条件が成立しませんでした」を表示する
を実行する
```

変数 x が 20 の場合の出力例

```
条件が成立しました
```

変数 x が 10 の場合の出力例

```
条件が成立しませんでした
```

7.3　多分岐の条件文 (else～if 文)

```
もし ≪条件式１≫ ならば
   | ＜処理１＞
を実行し，そうでなくもし ≪条件式２≫ ならば
   | ＜処理２＞
を実行し，そうでなければ
   | ＜処理３＞
を実行する
```

≪条件式 1 ≫が成立した場合、＜処理 1 ＞を実行し、
成立しなくて≪条件式 2 ≫が成立した場合、＜処理 2 ＞を実行し、
成立しない場合は＜処理 3 ＞を実行する。

使用例 1

```
もし x >= 80 ならば
  | 「x は 80 以上です」を表示する
を実行し，そうでなくもし x >= 60 ならば
  | 「x は 79〜60 の間です」を表示する
を実行し，そうでなければ
  | 「x は 59 以下です」を表示する
を実行する
```

変数 x が 95 の場合の出力例

```
x は 80 以上です
```

変数 x が 70 の場合の出力例

```
x は 79〜60 の間です
```

変数 x が 30 の場合の出力例

```
x は 59 以下です
```

使用例 2

```
もし x >= 80 ならば
  | 「x は 80 以上です」を表示する
を実行し，そうでなくもし x >= 70 ならば
  | 「x は 79〜70 の間です」を表示する
を実行し，そうでなくもし x >= 60 ならば
  | 「x は 69〜60 の間です」を表示する
を実行し，そうでなければ
  | 「x は 59 以下です」を表示する
を実行する
```

変数 x が 85 の場合の出力例

```
x は 80 以上です
```

変数 x が 75 の場合の出力例

```
x は 79〜70 の間です
```

変数 x が 62 の場合の出力例

```
x は 69〜60 の間です
```

変数 x が 25 の場合の出力例

```
x は 59 以下です
```

8　繰り返し

8.1　前条件判定の繰り返し文 (while-do 文)

```
≪条件式≫ の間,
   | ＜処理＞
を繰り返す
```

≪条件式≫が成立していれば、＜処理＞を実行する。
＜処理＞の実行終了後、再び≪条件式≫の判定を行い、
成立すれば＜処理＞を再び実行し、これを繰り返す。
≪条件式≫が成立しない場合は「を繰り返す」の次の行へ進む。

使用例

```
x < 5 の間,
   | x を表示する
   | x ← x + 1
を繰り返す
```

変数 x が 1 の場合の出力例

```
1
2
3
4
```

8.2　後条件判定の繰り返し文 (repeat-until 文)

```
繰り返し,
   | ＜処理＞
を,　≪条件式≫ になるまで実行する
```

＜処理＞を実行した後、≪条件式≫の判定を行う。
条件式が成立していなければ、＜処理＞を再び実行し、
成立した場合は次の行へ進む。

使用例

```
繰り返し,
   | x を表示する
   | x ← x + 1
を, x > 5 になるまで実行する
```

変数 x が 1 の場合の出力例

```
1
2
3
4
5
```

8.3 範囲指定の加算型繰り返し文 (for 文)

```
≪変数≫ を ≪数値１≫ から ≪数値２≫ まで ≪増加値≫ ずつ増やしながら,
   | ＜処理＞
を繰り返す
```

≪変数≫の部分に指定されたループ変数に≪数値１≫の値を代入し＜処理＞を実行する。
＜処理＞の実行後、ループ変数に≪増加値≫の値を加算し、
ループ変数の値が≪数値２≫になるまで繰り返す。

使用例

```
x を 1 から 5 まで 1 ずつ増やしながら,
   | x を表示する
を繰り返す
```

出力例

```
1
2
3
4
5
```

8.4 範囲指定の減算型繰り返し文 (for 文)

```
≪変数≫ を ≪数値１≫ から ≪数値２≫ まで ≪減少値≫ ずつ減らしながら,
   | ＜処理＞
を繰り返す
```

≪変数≫の部分に指定されたループ変数に≪数値１≫の値を代入し＜処理＞を実行する。
＜処理＞の実行後、ループ変数から≪減少値≫の値を減算し、
ループ変数の値が≪数値２≫になるまで繰り返す。

使用例

```
x を 5 から 1 まで 1 ずつ減らしながら,
    | x を表示する
を繰り返す
```

出力例

```
5
4
3
2
1
```

8.5 繰り返し文の脱出

繰り返しを抜ける

繰り返し文を途中で強制的に抜け出す命令です。

9 組み込み関数

9.1 文字列操作関数

書式	意味	使用例	戻り値	データ型
str2int(str)	str の 1 文字目を ASCII コードの数値に変換	`str2int("A")`	65	整数型
int2str(int)	引数 int を ASCII コードと見なしたときの対応する文字を返す	`int2str(70)`	F	文字列型
length(str)	str の文字列の長さを返す	`length("PEN")`	3	整数型
append(str1, str2)	str1 と str2 を結合した文字列を返す	`append("Mr.", "PEN")`	Mr.PEN	文字列型
substring(str, i)	str の先頭から i 文字よりも後の文字列を返す	`substring("smiles", 3)`	les	文字列型
substring(str, i, len)	str の先頭から i 文字よりも後ろの len 文字を返す	`substring("smiles", 1, 4)`	mile	文字列型
insert(str1, i, str2)	str1 の i 文字目の後に str2 を挿入する	`insert("abc", 2, "123")`	ab123c	文字列型
replace(str1, i, len, str2)	str1 の i 文字目の後から len で示される文字数を str2 で置き換える	`replace("abc", 1, 2, "123")`	a123	文字列型
extract(str, delim, i)	文字列 str を delim で区切り, i+1 個目にある文字列を返す	`extract("a:b:c", ":", 2)`	c	文字列型

- 引数 str, delim のデータ型は 文字列型 でなければならない
- 引数 int, i, j, len のデータ型は 整数型 でなければならない

9.2 数学関数

書式	意味	使用例	戻り値	戻り値のデータ型
random(x)	0〜x の乱数値 (整数) を返す	random(10)	0〜10 の値を返す	整数型
floor(x)	x の小数点以下 切り捨て	floor(24.64)	24.0	実数型
ceil(x)	x の小数点以下 切り上げ	ceil(24.64)	25.0	実数型
round(x)	x の小数点以下 四捨五入	round(24.64)	25	実数型
abs(x)	x の絶対値	abs(-234)	234	引数と同じ
int(x)	x の型を「整数」に変換	int(10.2345)	10	整数型
sin(x)	角度 x (ラジアン) の正弦を返す	sin(95.0)	0.683261714736121	実数型
cos(x)	角度 x (ラジアン) の余弦を返す	cos(50)	0.15425144988758405	実数型
tan(x)	角度 x (ラジアン) の正接を返す	tan(70)	1.2219599181369434	実数型
sqrt(x)	x の平方根の値を返す	sqrt(5)	2.23606797749979	実数型
log(x)	x の自然対数値 (底は e) を返す	log(2)	0.6931471805599453	実数型

- 引数 x のデータ型は 実数型 であるが、 整数型 で渡した場合、 実数型 に型変換されて取り扱われる
- abs(x) の戻り値のデータ型は、引数のデータ型によって決定される