4 擬似言語

1 擬似言語

- コンパイラなどの言語プロセッサは存在しないが，アルゴリズムを記述するために用いる擬似的なプログラム言語。
- 擬似言語の記述形式（仕様）に定められたものはない。

試験時に，仕様が与えられる。

〔擬似プログラムの例〕

```
宣言部
  ○ プログラム名：　平均値の計算
        ｛175ページの〈流れ図3〉を擬似言語で書いた例｝
  ○ 関数　　：　表示関数(a)
  ○ 整数型：　合計
  ○ 整数型：　平均
  ○ 整数型：　番号
  ○ 整数型：　T(5)

処理部
      ｛配列にデータを代入する｝
  • T(1) ← 90
  • T(2) ← 100
  • T(3) ← 80
  • T(4) ← 60
  • T(5) ← 70

      ｛ここからが，〈流れ図3〉の処理｝
  • 合計 ← 0

  • 番号 ← 1
  ■ 番号 ≦ 5
      • 合計 ← 合計 + T(番号)
      • 番号 ← 番号 + 1
  ■

  • 平均 ← 合計 ÷ 5
  • 表示命令(平均)
```

使用する変数や配列を最初に宣言する。

流れ図とは逆向きの矢印で，右から左へ代入することを意味する。

（注）ここでこのプログラムを理解する必要はない。

1 擬似言語の問題が出題されている

最近，「流れ図を勉強しても仕方がない」という意見が耳に入りますが，午前の問題やプログラム設計の問題では，流れ図が出題されます。流れ図をきちんと理解して，アルゴリズムの力をつけることが大切です。アルゴリズムの力は，プログラム言語の問題やプログラム設計問題の基礎力になります。

確かに，午後のアルゴリズムの問題で，「擬似言語」が用いられることが多くなりました。慣れないと難しく感じますが，流れ図をきちんと学び，アルゴリズムの力があれば，恐れることはありません。プログラム開発の実務経験がある人は，特別な対策は必要ないでしょう。

実務経験がない独学者の場合，腰をすえた学習が必要です。アルゴリズムの問題は，暗記では対応できません。理想的には数十本，少なくとも20本ぐらいの擬似言語のプログラムを，納得ができるまで読み込んでおかなければ，得点力をつけることができません。

市販の問題集は過去問題の流用が多く，いきなり難しい試験問題を解かなければなりません。そこで，この章では擬似言語プログラムを基礎から解説していきます。流れ図とそれに対応する擬似言語プログラムで重要なアルゴリズムを解説し，読解力の養成ができるように工夫しました。

2 本来，擬似言語は自由に書くことができる

本来は，コンピュータが理解できるのは機械語プログラムだけで，それ以外の言語を広義には擬似言語と呼びます。C言語やJava，COBOLなども擬似言語です。ただし，試験の「擬似言語」は，プログラム記述言語（PDL：Program Description Language）のことで，プログラム設計段階でプログラムの処理手順を記述するために用いる架空の言語です。

定められたPDLの公的な規格はありません（特定のCASEツールでは定めていることもあります）。設計者が自由に書くことができるものです。例えば，平均を求める処理なら，「平均←合計÷５」と書かなくても，「合計を5で割って平均を求める」と，日本語で手順を示してもいいのです。さらにいえば，日本語で簡潔に書けるので擬似言語を使う意味があります。

しかし，出題される擬似言語は，プログラムの命令行と１対１で対応付けられるものがほとんどで，「これなら擬似言語を使わずに，直接C言語などで書いたほうが速い」と思うものばかりです。つまり，試験では，特定のプログラム言語の経験がなくても解ける共通問題にするために，擬似言語が用いられています。

本書は，試験用の受験参考書ですから，理屈をこねて批判しても意味がありません。試験に出題される形式に合わせた擬似言語のプログラムを掲載していきます。

5 擬似言語の宣言部

1 宣言部の記述形式

● 手続，変数の名前や型，ファイルなどを宣言する領域である。

〔本書で採用する擬似言語の仕様〕

形式と例	説　明
○	・プログラム名や手続，変数などの名前や型などを宣言する。
(例) 名前と種類の宣言 ○ プログラム名： 成績処理 ○ 手続名： 集計 (a, b) ○ 関数名： 平方根 (a)	 ・プログラムの名は，「成績処理」である。 ・仮引数a，bをもつ「集計」という手続（定義済み処理）である。他のプログラムから呼び出される。 ・仮引数aを受け取る「平方根」という名の関数である。他のプログラムから呼び出される。 (注) 引数の型は指定してもしなくてもいい。
(例) 手続や関数の宣言 ○ 手続： 表示 (a, b) ○ 関数： 入力 (a, b)	 ・引数a，bをもつ「表示」という手続（定義済み処理）を使用することを宣言する。 ・引数a，bをもつ「入力」という関数を使用することを宣言する。 (注) 引数の型は指定してもしなくてもいい。 この宣言を省略することもある。
(例) ファイルの宣言 ○ ファイル： 給料マスタ 給料レコード (a, b)	 ・「給料マスタ」というファイル名で，そのレコードは，aとbという項目からなる。 (注) 項目の型は指定してもしなくてもいい。
(例) 使用する変数の型宣言 ○ 整数型： A ○ 実数型： B ○ 文字型： C ○ 整数型： D (10)	※いずれの型も型が同じ場合には，コンマで区切って名前を列挙できる。 ・小数点のない整数型の変数Aを宣言する。 ・小数点のある実数型の変数Bを宣言する。 ・文字型の変数Cを宣言する。1文字か，文字列かは，その時々に指定される。 ・10個の要素をもつ整数型の配列を宣言する。添字は，1から始まり，D (1) ～D (10) までの要素ができる。

2 プログラムの名前と種類

● 1行目は，そのプログラムの種類を表す。

①プログラム名……単独で実行されるか最上位の手続（メインルーチン）。

	(例)	(説明)
宣言部	○ プログラム名： サンプル	・サンプルという名のプログラムである。
	○ 実数型： 誤差 ○ 整数型： 数 ○ 文字型： 単語	} 3つの変数を宣言している。

②手続名……他のプログラムから呼び出される手続（サブルーチン）。
通常，引数が渡される。

	(例)	(説明)
宣言部	○ 手続名： 整列(データ，件数)	・整列という名の手続である。データと件数が，呼び出したプログラムから渡される。
	○ 手続： 表示処理( )	・この手続で，表示処理という手続を呼び出すことを示している。
	○ 文字型： 単語(100)	・単語という文字型の配列を宣言。単語(1)～単語(100)までの100個の要素ができる。

③関数名……他のプログラムから呼び出される関数。
通常，引数が渡され，処理部で呼んだ側に値を返す。

	(例)	(説明)
宣言部	○ 関数名： 平方根(値)	・平方根という名の関数である。値が，呼び出したプログラムから渡される。
	○ 整数型： 数	・数という整数型変数を宣言。

1 変数の宣言で，メモリ上に領域が確保される

流れ図では，変数の宣言なしに自由に変数を用いますが，一般的なプログラム言語では変数を宣言してから使用します。変数の宣言とは，変数の値を記憶する領域をメモリ上に確保することです。すでに述べたとおり，擬似言語に定められた規格はありません。試験でも，毎回，異なる言語仕様の擬似言語が出題されています。

学習においては，1つの擬似言語の仕様を決め，たくさんの擬似言語のプログラムを読み込むことが大切です。そこで，非公開になった問題も含め，過去の出題例を分析して，どのような仕様の擬似言語が出題されても，受験時に即対応できるような擬似言語の仕様を作りました。この章では，この仕様の擬似言語を用いて，頻出アルゴリズムを紹介していきます。

6 擬似言語の処理部と分岐

1 処理部の記述形式

● 処理手順を記述する領域である。

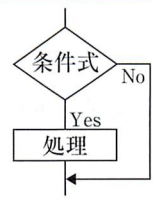
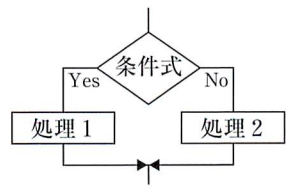
〔本書で採用する擬似言語の仕様〕

形式と例	説　明
• 変数←式	・変数に式の値を代入する。
＋，－，×，÷ ＊，／	・式に使用する算術演算子である。なお，＊はかけ算，／は割り算で，それぞれ×と÷と同じ演算子である。
＝，≠ ＜，≦，＞，≧	・条件式に使用する比較演算子である。
And，Or，Not	・条件式に使用する論理積（「かつ」），論理和（「または」），否定（「でない」）の論理演算子である。
{ 文 }	・説明のための注釈である。実行されない。

2 分岐処理の記述形式

● 2分岐制御文をもつ。

〔本書で採用する擬似言語の仕様〕

形式と例	説　明
▲ 条件式 Yes 　処理 ▼	・条件式が真のときに処理を実行する。 
▲ 条件式 Yes 　処理1 ◇ No 　処理2 ▼	・条件式が真のときに処理1を実行する。 ・条件式が偽のときに処理2を実行する。 

3 分岐処理の実行例

宣言部
○ プログラム名： 分岐処理
○ 手続： 表示処理(“文字列”)
｛引数で指定した文字列を表示する手続である｝
○ 整数型： 条件

処理部
• 条件 ← 1
条件 = 1 真
Yes
• 表示処理(“通過１”)
条件 = 2 偽
Yes
• 表示処理(“通過２”)
◇No
• 表示処理(“通過３”)
条件 ≧ 0 真
Yes
• 表示処理(“通過４”)
条件 = 1 真
Yes
• 表示処理(“通過５”)
◇No
• 表示処理(“通過６”)

“通過1”と表示する。C言語のprintf関数やCASLⅡのOUT命令と思えばよい。

〔実行結果の例〕

```
通過１
通過３
通過４
通過５
```

1 流れ図の判断記号に相当する制御構造を覚えよう

擬似言語の代入文は，流れ図とは逆で，一般的なプログラム言語と同様に左←右です。算術演算子の＊や／が使われたこともありましたので，高水準言語の経験がない人は，覚えておいてください。

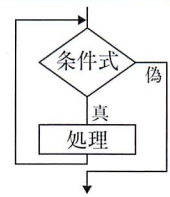
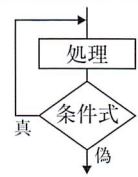
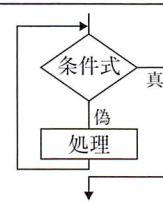
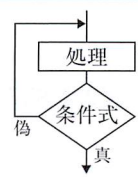
分岐処理は，一般のプログラム言語でいえば，If文とIf－Else文です。真の条件のときに直後に書いた処理が実行され，偽の条件がないときは◇以降は省略できます。If文のほうが読みやすいのですが，試験によって仕様が違うものの，こんな感じで出題されています。また，入れ子（分岐処理の中に分岐処理を書くこと）にすることができます。上のプログラムを見て，擬似言語の感触をつかんでくださいね。

7 擬似言語の繰返し処理

1 繰返し処理の記述形式

● 4種類の繰返し制御文をもつ。

〔本書で採用する擬似言語の仕様〕

形式と例	説　明
■ 条件式 　処理 ■	・前判定型繰返し ・**条件式**の値が**真**の間，**処理**を繰り返し実行する。 ・はじめから条件式が偽の場合は，1回も処理が実行されない。 
■ 　処理 ■ 条件式	・後判定型繰返し ・**条件式**の値が**真**の間，**処理**を繰り返し実行する。 ・どんな条件式でも，必ず1回は処理が実行される。 
■ ▽ 条件式 　処理 ■	・前判定型繰返し ・**条件式**の値が**真**になるまでの間，**処理**を繰り返し実行する。 ・はじめから条件式が真の場合は，1回も処理が実行されない。 
■ 　処理 ■ △ 条件式	・後判定型繰返し ・**条件式**の値が**真**になるまでの間，**処理**を繰り返し実行する。 ・どんな条件式でも，必ず1回は処理が実行される。 

（注）最近は，上2つだけの出題が多いが，下2つのように終了条件を書く「繰返し制御文」が出題されたことがある。

2 繰返し処理の実行例

```
宣言部
  ○ プログラム名： 分岐処理
  ○ 手続： 表示処理(引数)      ｝表示処理は，引数で指定した
  ○ 整数型： 条件                文字列や変数の値を表示する。
  ○ 整数型： 数                  表示処理の手続は省略｝

処理部
  • 条件 ← 1
  ■ 条件 ＜ 1 の間
      • 表示処理(“通過１”)
      • 条件 ← 条件 ＋ 1
  ■
  • 条件 ← 1
  ■
      • 表示処理(“通過２”)
      • 条件 ← 条件 ＋ 1
  ■ 条件 ＜ 1 の間
  • 条件 ← 2
  ■▽ 条件 ＞ 1 まで
      • 表示処理(“通過３”)
      • 条件 ← 条件 ＋ 1
  ■
  • 条件 ← 2
  ■
      • 表示処理(“通過４”)
      • 条件 ← 条件 ＋ 1
  ■△ 条件 ＞ 1 まで
  • 数 ← 1
  ■ 数 ≦ 5 の間
      • 表示処理(数)
      • 数 ← 数 ＋ 1
  ■
```

〔実行結果の例〕

```
通過２
通過４
 1
 2
 3
 4
 5
```

試験場では，「まで」と鉛筆で書き込むと間違わない。

試験場では，「の間」と鉛筆で書き込むと間違わない。

1 繰返し制御には４つのパターンがある

繰返し（ループ）の制御には，①条件式をループの前で判定する**前判定**，②最後に判定する**後判定**，③条件式が真の間繰り返す**繰返し条件**(while型)，④真になったらループを抜ける**終了条件**(Until型)，以上４種類のパターンがあります。上の例では，１回目から「条件 ＜ 1」を満たさないので，処理（通過１）は実行されません。

試験では，与えられた仕様から，これらの違いをすばやく読み取り，擬似言語のプログラムを読む能力が必要です。慣れないと大変ですね。

8 擬似言語の手続呼び出し

1 手続呼び出しの記述形式

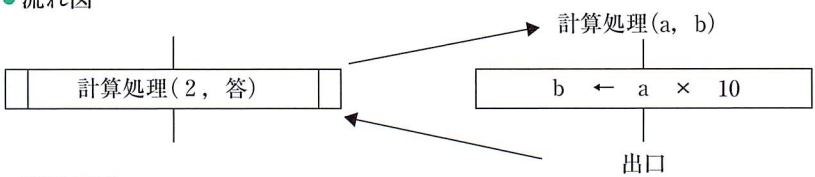
- 別の手続（サブルーチン）を呼び出すことができる。
- 手続内で宣言された変数は，手続内だけで通用する。

〔本書で採用する擬似言語の仕様〕

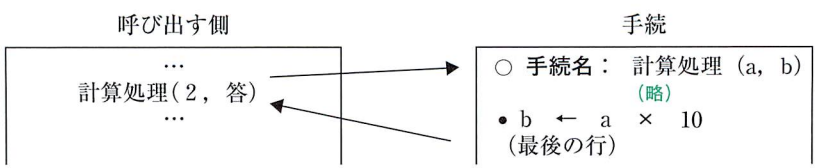
形式と例	説　明
• 手続名(実引数，…)	・実引数を指定して，手続を呼び出す。 ・参照呼び出しか，値呼び出しかは，その都度指定される。指定されない場合は，参照呼び出しである。
(例) • 計算処理(数値)	・実引数に変数の数値を指定して，計算処理という手続を呼ぶ。
• 集計処理(データ( ))	・実引数に配列のデータを指定して，集計処理という手続を呼ぶ。

2 手続の呼び出し

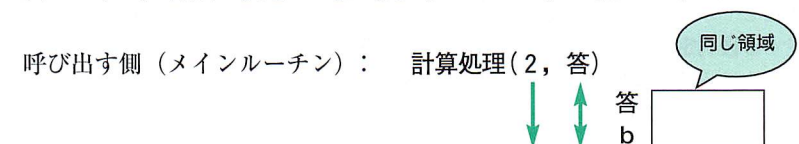
- 流れ図



- 擬似言語

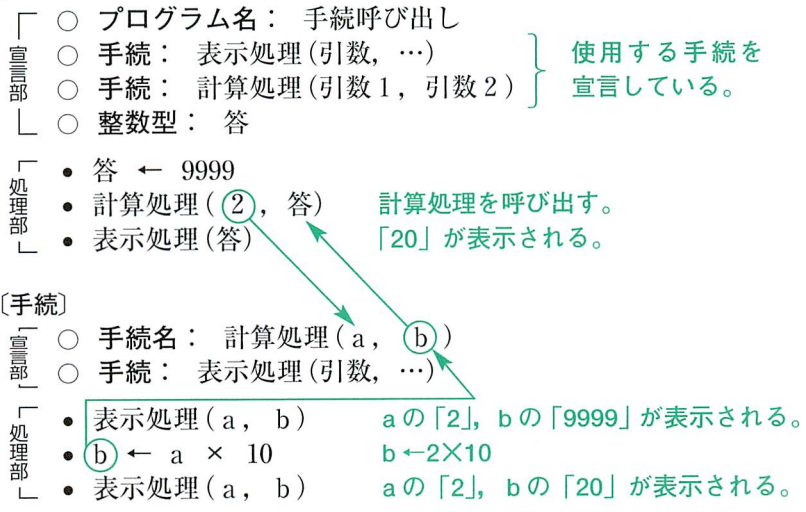


- 次のように，引数に指定した値（実は，アドレス）が渡ってくる。



3 参照呼び出し

- 実引数の主記憶装置のアドレスが，仮引数に渡される。
- 呼び出す側の実引数と呼び出される手続で，同じ領域を参照している。



1 流れ図の定義済み処理が手続だ

流れ図で，他の流れ図の処理を行うときに，定義済み処理記号を用います。この定義済み処理を，一般的なプログラム言語ではサブルーチンとか，プロシージャと呼びます。ここで用いる擬似言語では，プロシージャの日本語訳である「**手続**」という用語を用いています。手続の処理を実行することを「**呼び出す**」といいます。

手続の呼び出しでは，括弧の中に引数を指定して，それを呼び出される側の手続で受け取ることができます。呼び出す側の引数を**実引数**，呼ばれる側の引数を**仮引数**と呼びます。

2 参照呼出しでは，同じ値を参照する

2 の手続呼び出しの例では，実引数に「2」と「答」を指定しています。それぞれ手続の仮引数aとbに渡され，aが2，bが答の値になります。そして，a×10を計算した値（2×10＝20）をbに代入します。この値が答に渡されて，呼び出した側に戻ったとき答の値は20になっています。

実は，答からbに9999という値を渡しているのではなく，主記憶装置上に取られた答の領域のアドレスを渡しています。簡単にいえば，答とbとで名前は違うけど，同じ領域です。したがって，bの値を変更すると答の値も変わります。このような引数の渡し方を**参照呼び出し**（アドレス渡し）といいます。

9 擬似言語の関数呼び出し

1 関数呼び出しの記述形式

- 別の関数を呼び出すことができる。
- 関数内で宣言された変数は，関数内だけで通用する。

〔本書で採用する擬似言語の仕様〕

形式と例	説　明
• 関数名(実引数，…) (例) • c←計算関数(a，b)	・実引数を指定して，関数を呼び出す。 ・実引数は，値渡しである。 ・実引数aとbを指定して，計算関数という関数を呼び出し，結果の値をcで受け取る。

2 戻り値の記述形式

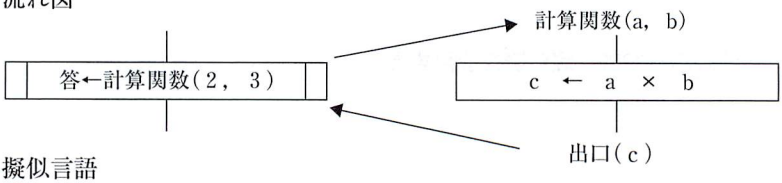
- 自身が関数の場合は，値を戻すことができる。

〔本書で採用する擬似言語の仕様〕

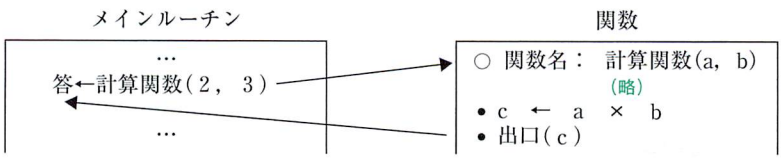
形式と例	説　明
• 出口(戻り値) (例) • 出口(a)	・自身が関数の場合に用い，この関数を呼び出したプログラムに戻り値を返す。 ・呼び出したプログラムに，aの内容を返す。

3 関数の呼び出し

- 流れ図



- 擬似言語



4 値呼び出し

- 実引数の値を，仮引数にコピーして渡される。
- 呼び出す側の実引数と呼び出される手続の仮引数は，異なる領域にとられる。
- 関数内で仮引数の値を変更しても，呼び出した側の実引数に影響を与えない。

〔メインルーチン〕

宣言部
- ○ **プログラム名：** 関数呼び出し
- ○ **手続：** 表示処理(引数，…)
- ○ **関数：** 関数値＝計算関数(引数１，引数２)
- ○ **整数型：** 金額
- ○ **整数型：** 単価
- ○ **整数型：** 数量

処理部
- 単価 ← 100
- 数量 ← 3
- 金額 ← 計算関数(単価，数量)　計算関数を呼び出す。
- 表示処理(単価，数量，金額)　100, 3, 300が表示される。

〔関数〕

宣言部
- ○ **関数名：** 計算関数(a，b)
- ○ **手続：** 表示処理(引数，…)
- ○ **整数型：** c

処理部
- 表示処理(a，b)　aに100，bに3がコピーされる。
- c ← a × b　c←100×3
- a ← 8888　わざと，aを8888に変更。
- b ← 9999　わざと，bを9999に変更。
- 表示処理(a，b)　8888，9999が表示される。
- 出口(c)　cの値300を戻す。

1 値呼び出しは値がコピーされるので，呼び出した側に影響を与えない

関数呼び出しでは，実引数の単価100と数量３が，仮引数ａとｂにコピーされて渡り，100×3を計算した300がｃに代入されます。このｃを出口で指定しているので，呼び出した側に戻ったときの金額は300になっています。

値をコピーして渡す**値呼び出し**（値渡し）は，単価とａ，数量とｂはそれぞれ異なる領域にとられるので，ａやｂの値をわざと8888や9999に変更しても，単価や数量の値は変更されません。参照呼出し（アドレス渡し）と値呼び出しの違いは，396ページでも図を用いて説明しています。

10 変数の通用範囲

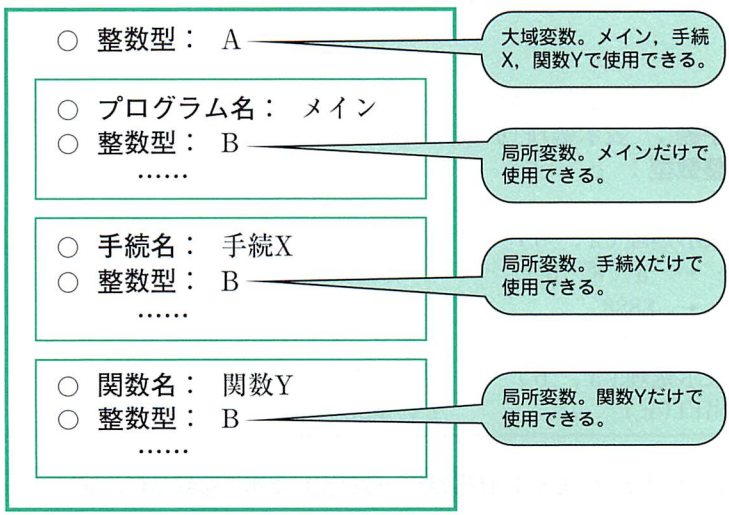
1 通用範囲からみた変数の種類

①大域変数（グローバル変数）
- 複数の手続や関数で通用する変数。

②局所変数（ローカル変数）
- その手続や関数内だけで通用する変数。

〔本書で採用する擬似言語の仕様〕

- 手続や関数外で宣言した変数は，すべての手続や関数で通用する大域変数である。「グローバル」と明示することもある。
- 手続や関数内で宣言した変数は，その手続や関数内だけで通用する局所変数である。



1 局所変数は，名前は同じでも異なる領域に割り当てられた別の変数だ

大域変数のAは主記憶装置上に1つの領域が取られ，すべての手続や関数から参照されます。局所変数のBは，名前は同じでも別の変数で，主記憶装置上に手続や関数ごとに3つの領域が取られます。宣言した手続や関数内からしか参照できません。

2 大域変数と局所変数の擬似言語プログラム

○ 整数型： A	大域変数。
○ プログラム名： メイン	
○ 手続： 表示処理(引数, …)	
○ 手続： 手続X( )	
○ 関数： 戻り値＝関数Y( )	
○ 整数型： B	局所変数。
○ 整数型： C	局所変数。
• A ← 999	
• B ← 10	
• Call 手続X	手続Xを呼び出す。
• 表示処理(A, B)	888と10 を表示。
• C ← 関数Y	関数Yを呼び出す。
• 表示処理(A, B, C)	777, 10, 100を表示。
○ 手続名： 手続X( )	
○ 整数型： B	局所変数。
• 表示処理(A, B)	999と0※を表示。
• A ← 888	
• B ← 20	
• 表示処理(A, B)	888と20を表示。
○ 関数名： 関数Y( )	
○ 整数型： B	局所変数。
• 表示処理(A, B)	888と0※を表示。
• A ← 777	
• B ← 30	
• 表示処理(A, B)	777と30を表示。
• 出口(100)	100を返す。

※0で初期化されていると想定。

2 大域変数は，すべての手続や関数で更新することができる

メインから手続Xを呼び出すと，手続Xの中でAに888，Bに20を代入していますが，局所変数のBはメインのBには影響を与えず10のままです。

メインから関数Yを呼び出すと，関数Yの中でAに777，Bに30を代入していますが，局所変数のBはメインのBには影響を与えず10のままです。

これに対して，大域変数のAは，手続や関数で値を変更すると，同じ領域なのでメインのAの値も変更されます。

擬似言語の問題に限れば，ここまで知らなくても解けるものが多いですが，変数の通用範囲（変数のスコープ）に関する問題が「午後」で出題されています。