5.2 流れ図

流れ図 (flow chart) はアルゴリズムを図式化したもので、コンピュータの処理手順となるデータの流れ・判定条件・実行の推移などを**流れ図記号**[5] (付録 C.1 参照) を用いて描きます。流れ図のようにアルゴリズムを図式化することで、問題の定義や分析または解法がより明確となり、プログラムの設計や作成に非常に役立ちます。また、第三者にも正確に伝えることができます。

それでは、流れ図について詳しく見ていきましょう。主な流れ図記号には表 5.1 のような記号があり、これらの記号をアルゴリズムの流れに従って実線で繋いで流れ図を描きます (流れの指向性を明示する場合は矢印で記述します)。

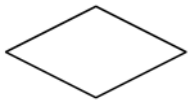
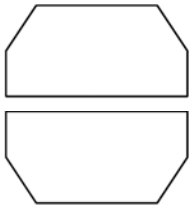
記号	記号名	記号の説明
	処理 (process)	任意の種類の処理を表します。特に、データの値・形・位置を変えるような定義された演算を表す場合に使用されます。
	定義済み処理 (predefined process)	サブルーチンやモジュールなど、別の場所で定義された 1 つ以上の演算または命令群からなる処理を表します。
	準備 (preparation)	変数の初期設定など、その後の動作に影響を与えるための命令または命令群の修飾を表します。
	判断 (decision)	1 つの入力と幾つかの択一的な出口の中で記号中の条件の評価に従って唯一の出口を選ぶ判断を表します。
	ループ端 (loop limit) 上図: ループ始端 下図: ループ終端	1 組のループ始端とループ終端の記号中には同じループ名が入り、記号中の終了条件を満たすまでループ始端からループ終端に挟まれた区間の処理の繰り返しを表します (図 5.7 の⑥参照)。ただし、1 組のループが別のループを含むような場合は必ず**入れ子**構造になります。
	結合子 (connector)	一連の流れ図を分割して描いた場合など、他の部分へ継続していることを表します。流れ図の結合を表します。
	端子 (terminator)	流れ図の始まりと終わりを表します。なお、始まりには「開始 (start)」、終わりには「終了 (end)」と記号中に表記します。

表 5.1: 流れ図記号

[5] このテキストでは JIS X 0121 (1986 年) 情報処理用流れ図記号を用います。この流れ図記号は情報処理技術者試験で使用されています。

また、流れ図はアルゴリズムに従って基本的に上から下に左から右に記述します。具体例として、前節の平方根を求めるアルゴリズムと最大公約数を求めるアルゴリズムの流れ図を挙げておきます。

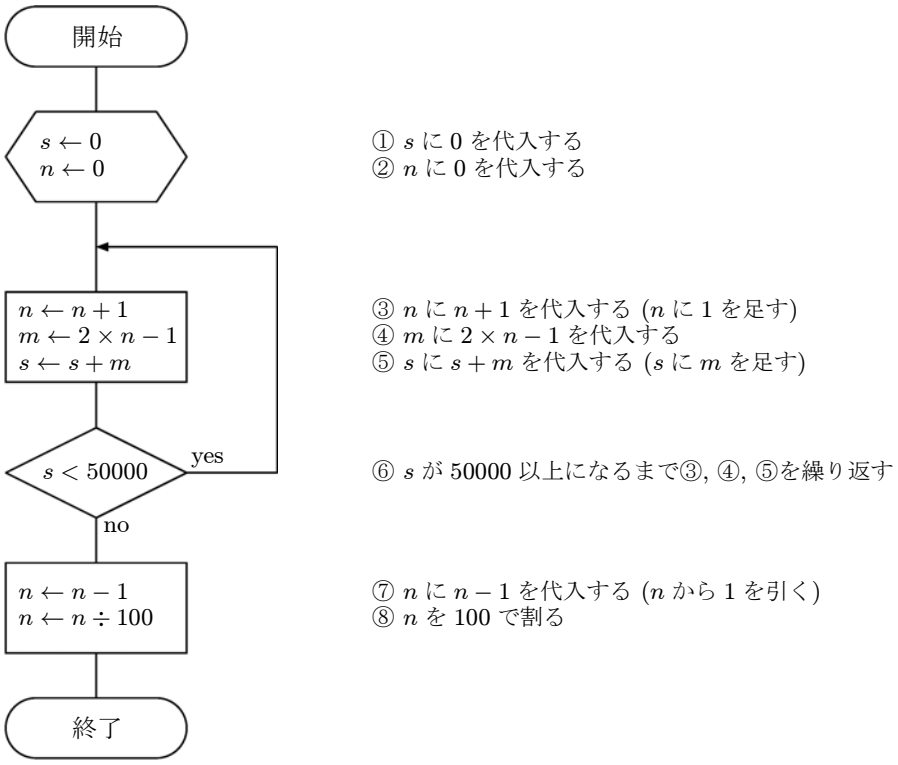
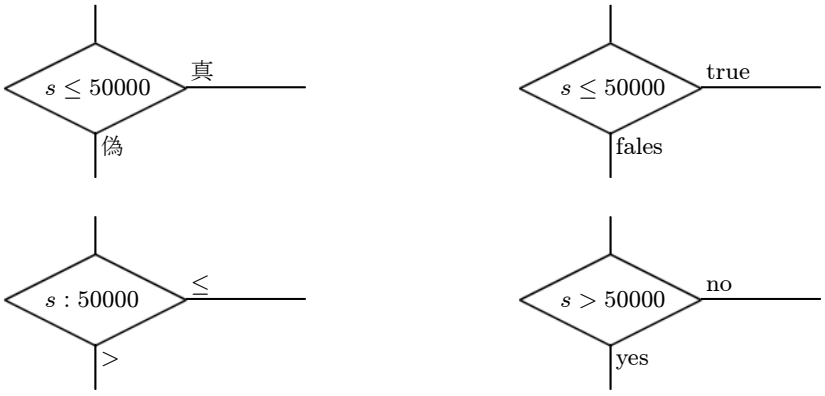


図 5.5: 5 の平方根を小数第 2 位まで求める流れ図

【注意】処理と準備の流れ図の中で使用されている「← (矢印)」は、右辺の値 (右辺) を左辺の変数 (左辺) に代入することを表します。また、判断の流れ図記号の中には条件 (数値の比較; $=, \neq, <, >, \leq, \geq$ など) または命題 (論理演算; 否定, 論理和, 論理積 など) が記述されますが、記述方法がいくつかあるので以下に紹介しておきます (図 5.5 の場合)。



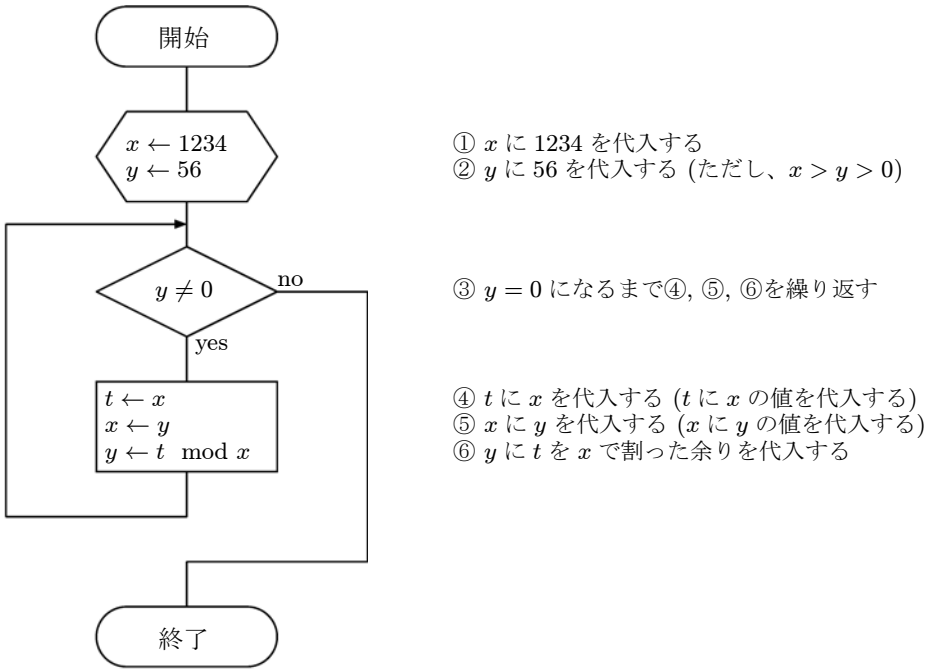


図 5.6: 正の整数 x, y の最大公約数を求める流れ図

現在、アルゴリズム (またはプログラム) の設計において主流となっているのは**構造化プログラミング** (structured programming) と呼ばれる方法です。構造化プログラミングが主流となる前は、特に制限がなかったため各設計者が独自にアルゴリズムの設計を行っており、第三者にも理解し辛いものでした[6]。このような背景と人間は誤解や過ちを起こしやすいという観点から、1966 年に C. Bohm と G. Jacopini によって**構造化定理**という理論が発表されました。この理論は、「1 つの入り口と 1 つの出口を持つように設計されていれば、三つの基本的な論理構造 (図 5.7 の① 順次, ②選択, ④繰り返し (前判定)) の組み合わせで、どんなアルゴリズムの理論も記述できる」というものです。構造化プログラミングはこの理論を取り入れたもので、現在使用されている多くのプログラム言語もこの理論に従っています。

また、構造化プログラミングに関していえば、前記の流れ図記号を用いて流れ図を記述しても構いませんが、階層化された構造を、さらに分かりやすく記述することのできる**構造化チャート**もあります。現在使用されている代表的な構造化チャートには **NS チャート** (Nassi Schneiderman chart) を改良した **PSD** (Program Structure Diagrams) や、日立製作所から発表された **PAD** (Problem Analysis Diagrams) などがあります。

[6]特に問題とされたのが、goto 文の多用によるプログラムの複雑化です。なお、このような状態をスパゲティが複雑に絡み合っている様子に比喩されて**スパゲティ構造**と呼ばれています。

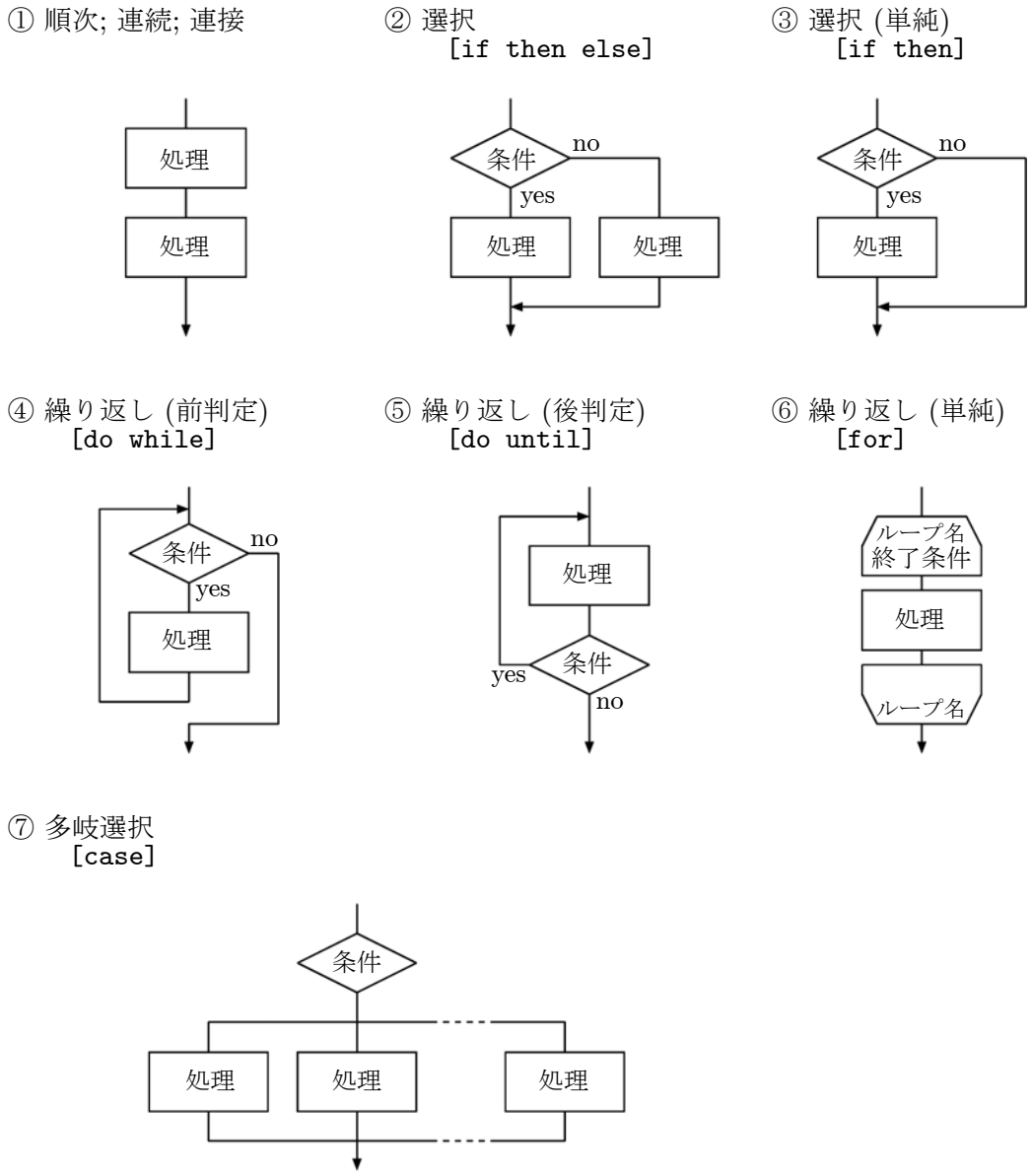


図 5.7: 構造化プログラミングの論理構造

問題 1　5.1.2 節の後半部分を参考に、効率よく平方根の値を求めるアルゴリズムの流れ図を描きなさい。

問題 2　最小公倍数を求めるアルゴリズムの流れ図を描きなさい。

問題 3　ハノイの塔のアルゴリズムの流れ図を描きなさい。

問題 4　図 5.5 の流れ図 (後判定) を繰り返し (前判定) を用いた流れ図に描き直しなさい。

問題 5　図 5.6 の流れ図 (前判定) を繰り返し (後判定) を用いた流れ図に描き直しなさい。