Ⅰ　システム開発とは

到達目標

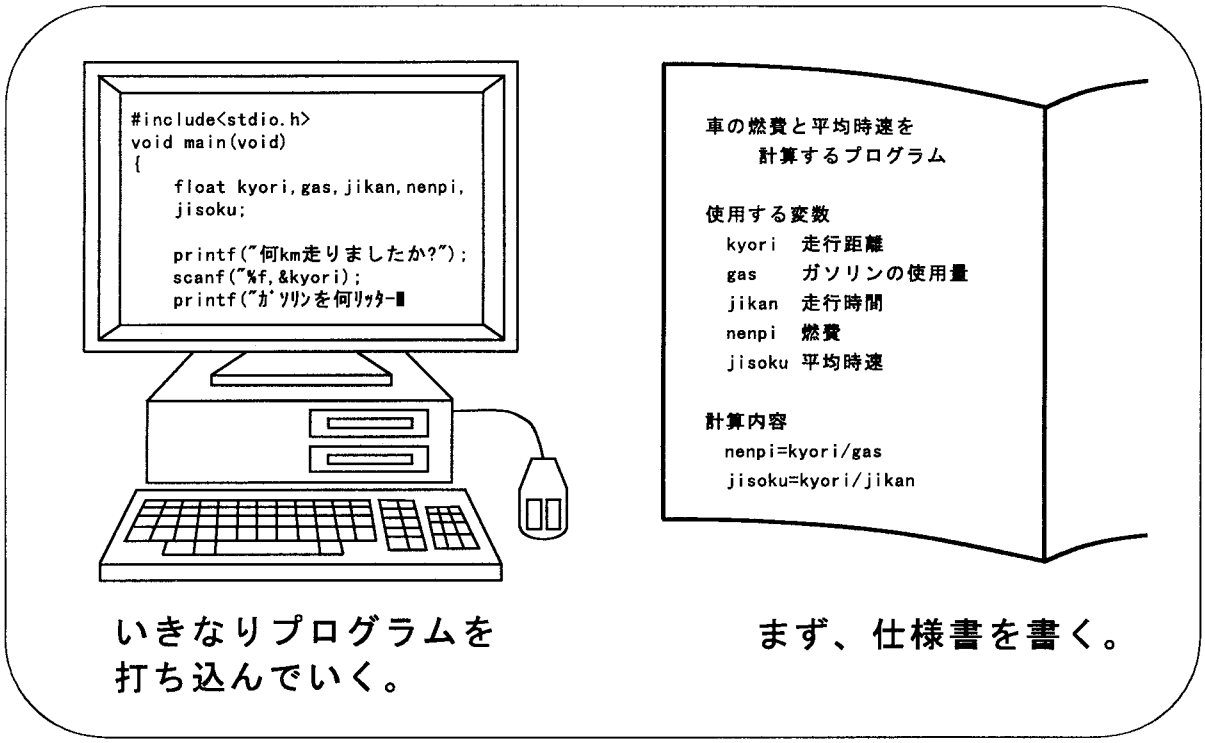
システム開発がなぜ必要なのかを理解させる。特に、個人でプログラムを作成する場合と、企業がシステムを開発する場合の相違点を認識させる。また、システムを開発するに当たって、その中心となるのはコンピュータではなく、あくまで「人」であるということを理解させる。

1　システム開発の必要性

今、あなたがパソコンやワークステーション上で何らかのプログラムを作成すると仮定しよう。全くの初心者ならば、書籍のプログラムや周りにいる人のプログラムを丸写しにするだけであろうが、プログラムというものが少し理解できてくると、アルゴリズムを変更するには至らないが入出力などの細かいところに自分なりに手を加えたり、また、もう少しレベルが上がると、規模は小さいが全くオリジナルのプログラムを作成するであろう。最近では、コンピュータグラフィクスやマルチメディアシステムが急速に発展・普及してきたことから、ハードウェアレベルだけでなく、ソフトウェアであるプログラミング言語内に、簡単にグラフィクスを表示したり、効果音を出したりするためのライブラリが揃ってきている。それらを利用すれば、見た目もなかなかの、ユーザインタフェースまでも考慮したプログラムが作成できるといっても過言ではない。特に、Windows 環境でのプログラミングは、ＧＵＩ（グラフィカルユーザインタフェース）を基本としていることから、数年前とは比べものにならないほど、簡単かつ短期間で「見た目の良い」ソフトウェアを作成することが可能となっている。しかしながら、システム開発という観点からみれば、プログラムがうまく動作したか否かや、「見た目の良い」プログラムであるかそうでないか、また、プログラムにバグがあるかないかという問題より、むしろ、その開発過程が大切となってくる。

あなたは、いきなりコンピュータに向かってプログラムを打ち込んでいくであろうか。それとも、どのような仕様・内容のプログラムを作成したいのか、どのような構造のプログラムにするのか、また、どのような変数をプログラムで使用するのかなどのプログラムの開発・設計に関することを、一旦、紙に書き出してからプログラムを作りはじめるであろうか(図Ⅰ－1)。

まず、いきなりコンピュータに向かってプログラムを打ち込んでいくタイプの人を「プログラミング（コーディング）優先派」と名付けよう。プログラミング優先派には、非常に短期間で、一気にプログラムを完成させなければならない必要性が生じる。それは、１度作成したプログラムを後で見直しても、よほど記憶力の良い者でない限り、１週間も経てばプログラムの内容を忘れてしまうからである。また、完成後に簡単な変更を加えるときでさえ、プログラムの



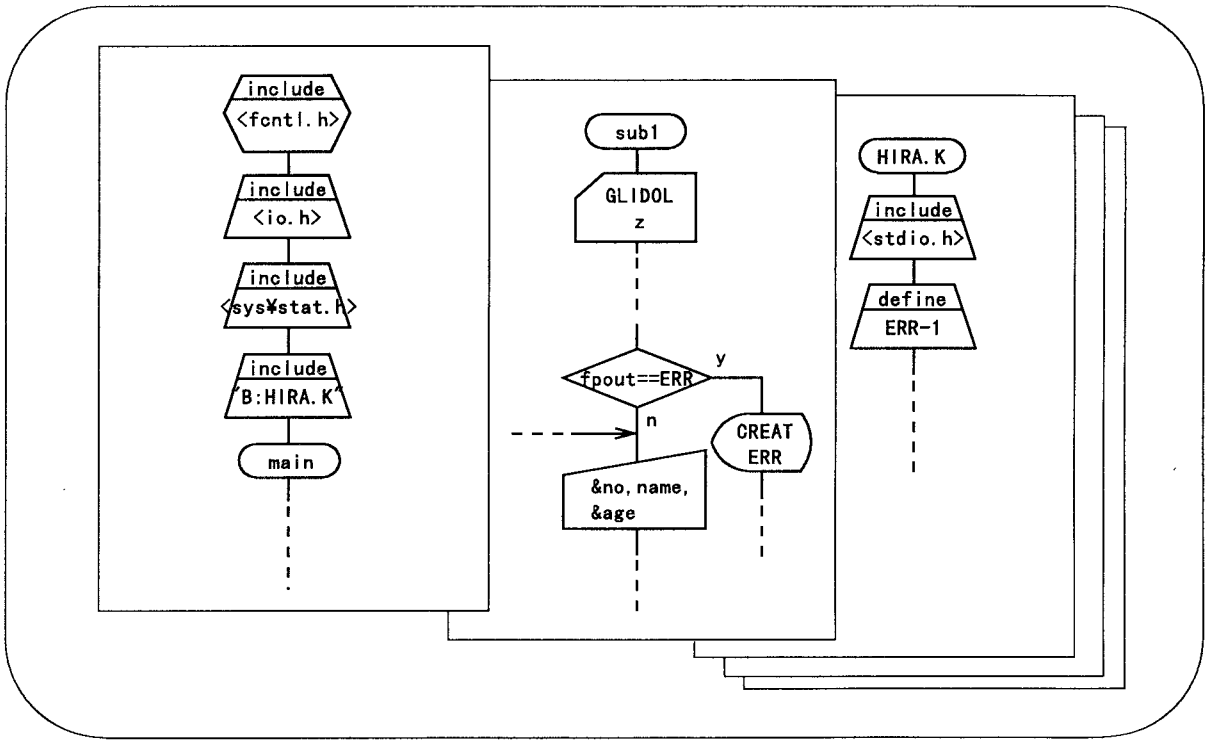
図Ⅰ-1　プログラムの開発・設計に関する個人差

内容を思い出さなければならないといった二度手間が生じることはいうまでもない。ただし、プログラムの規模が小さければ、曲がりなりにも動作するプログラムはできてしまうのである。特に、最近のプログラミング言語は、数年前とは比べものにならないヘルプ機能やデバッグ機能を備えていることから、何とか動作するプログラムが作れてしまうのである。また、ハードウェアの高速化により、多少長いプログラムを打ち込んでも、昔のようにコンパイルが終了するのに何分も待たなければならないということはなく、ものの数秒で終了するので、精神的ストレスなど皆無に等しい。しかも、自分で１回プログラムを作成すれば、それをコピーするだけで他の人が利用できるのだから、これでよいという考え方もある（現在では、汎用機を用いてその機種独自のプログラムを開発することはごく少数となっており、通常、Windows やＵＮＩＸ上でプログラムを開発するので、メーカーや機種が異なっていても基本的にプログラムは動作する。）。

それでは、次に、プログラムの仕様・内容や構造などのプログラム設計書を先に作成してからプログラミングに移るタイプの人を「設計優先派」と名付けてみる。設計優先派の人は、時間をかけてこつこつとプログラムを完成させていく方法が選択でき、また、「プログラミング優先派」と同様、一気にプログラムを完成させることもできる。当然のことながら、作りかけのプログラムの構造や各変数の意味は、設計書に記述してあるので、忘れたとしても思い出さなければならないといった二度手間が生じることはない。また、設計書があるので、ある程度規模の大きなプログラムも作成することができよう。

「設計優先派」の方が、いかにも優れているかのように述べたが、実のところ、個人の趣味

で小規模なプログラムを作成している範囲ではそれ程大差はない。実際、筆者らも学生のときは、１人で何千何万行にもわたるプログラムを書いていた（開発環境に恵まれていたので、計算のアルゴリズムや、一般にいう「エレガントなプログラム」の作成のことなど考えたこともなかったので、単に行数が多くなっただけである。）が、設計書を書いたことなど一度もなかった。書くのが面倒臭かったといえば格好良いが、実はそのような手法があることを知らなかったのである。一応は、講義で習ったフローチャートを用いて、プログラムの流れを分かり易くするために図示していくのだが、Ａ４用紙 10 枚も書けば、どこがどうなっているのか余計分からなくなる始末である（図Ｉ－２）。仕方がないので、山のようにあった変数の仕様だけをノートにまとめておいた記憶がある。今から思えば不思議であるが、それでもプログラムは作成でき、しかも立派に動くのである。



図Ｉ－２　フローチャートだけではすぐに限界がくる

それでは、対象を個人ではなく、企業に移してみよう。企業は営利目的で成り立っているものであり、その方針は個人や大学とは全く異なることはいうまでもない。一言でいえば、何をするにも「お客様」という相手の要求を満たさなければならないのである。いくら先端的で、また、バグのひとかけらも見当たらないソフトウェアが開発できたとしても、それが「お客様」のニーズと一致しなければ、商品価値など全くない。ソフトウェアであるプログラムは、ハードウェアである他の工業製品と同じく、要求分析・設計・製造・検査・保守という工程に従って生産される。設計書のないプログラムを生産することは、企業ではまず行われない。そ

れは、プログラムにバグがあったり、また、動作が正常でないといったような苦情がユーザから寄せられたとき、プログラムを作成した人が不在であったり、転勤していたり、ときには退職していたりし、そのプログラムを理解できる人が誰もいないという状況に陥るのを未然に防ぐ目的からである。また、設計書がなければ、最悪の場合、プログラムを作成した本人でさえも、そのプログラムの内容が分からなくなる可能性がある。このようなことが起きれば、当然のことながらバージョン・アップや他のシステムへの移植など不可能となる。つまり、企業において「商品となるプログラム」を開発することは、もはや１人の力では不可能なのである。

では、多人数で開発すれば、「商品となるプログラム」が生まれるのかというと、そうは簡単にいかない。なぜかというと、ソフトウェアというもの自体が他の工業製品と違い、目に見えないものだからである。機械であれ電子回路であれ、目に見える工業製品は、その製作工程にレベルの違いはあるものの、ある一定水準の力を持った技術者が集まり設計・製作すると、それほど大差なく仕上がってくるのである。しかしながら、ソフトウェアに関しては、設計工程であれ製造工程であれ、全ての工程で目に見えるのは仕様書に書かれたドキュメント（文書）だけである。ドキュメントだけを頼りに、仕様を満足するソフトウェアを作成しろといわれると、いくら簡単なシステムでさえ、十人十色、千差万別のものとなるであろう。そこで、企業においては、このような人為的な差を吸収するために、完全な階層的分業体制、すなわち工程分けを行い、人員の適材適所を考えた上でソフトウェアを開発している。

このように、たった１人で規模の小さなプログラムを作成するときには、その工程分けはさほど問題ではなかったが、複数の人員がある程度規模の大きなプログラムを作成するようになると、ソフトウェア作成時の工程分け、すなわちシステム開発時の工程分けを明確に行うことが重要となってくる。また、工程分けを行った結果、異なった開発工程への引継ぎとして唯一の共通の仕様であるドキュメントが渡されることとなる。

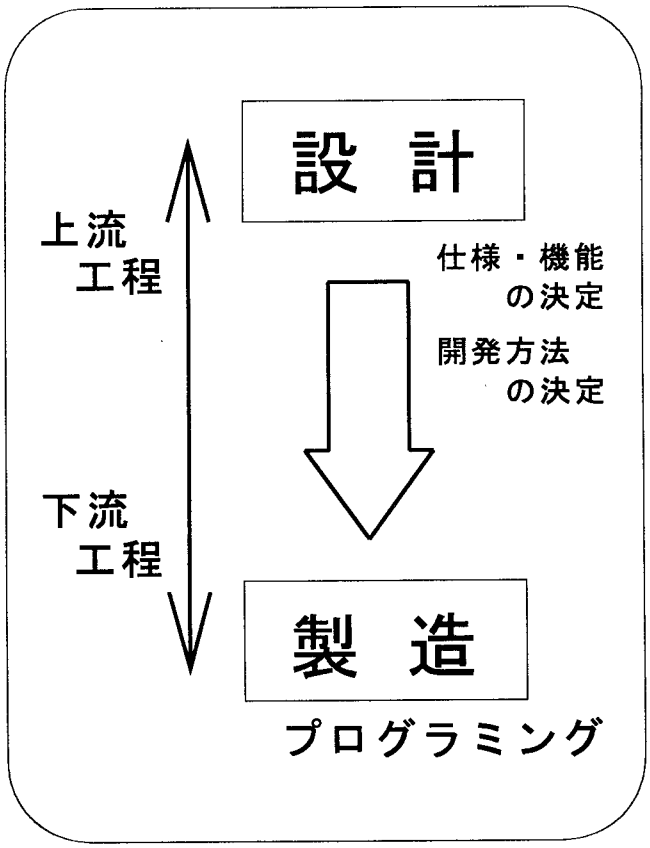
では、実際にどのような工程があり、どの人員をそこに配置し、また、誰がドキュメントを作成するであろうか。

２　システム開発における適材適所

システム開発に関する多くの書籍は、その工程に重きを置き、人の配置に関してはあまり考慮していないように見受けられるが、実際はこの「人員配置」がシステム開発では重要なポイントとなる。

「彼は選手としては今一だったが、監督としては素晴らしい」という言葉を、野球やサッカーの世界ではよく聞くが、これはシステム開発の世界でも同様で、「彼はプログラマーとしては今一だが、設計をさせたら右にでるものはいない」という者も存在する（各教育機関で情報処理関連の講義・実習を担当されている先生方はお分かりであろう。）。

システム開発の工程をごく簡単に図Ｉ－３のように大別してみる。上流工程が設計、下流工



図Ⅰ-3　簡略化したシステム開発工程

程が製造という開発工程になり、設計では仕様や機能および開発方法の決定、製造では実際のプログラミングを行うものとする。設計の工程は、ユーザ側からのニーズとシステムの分析結果以外に何もないところから出発しなければならず、そこから論理を展開・発展させて計画書・設計書を作成しなければならない。逆に、製造の工程では、計画書・設計書に添った事柄をプログラムに変換するだけである。どちらが難しいかは一目瞭然であるが、不思議とプログラミングより設計の方が得意になる者もでてくる。下流工程のセンスがないのに、上流工程のセンスが飛び抜けているのである。

言い表しにくいが、簡単に大別してみると、上流工程の技術は、良い悪いは別として物事を広く見ることのできる者、もしくは論理展開が一方向でない者が向いており、下流工程の技術は、教えられたことを忠実に実現できる者が向いているといえる。我々自身、毎年１人当たり数十名の学生を担当しているが、その中の 2、3 名の学生は、特別なことなど何も教えていないのに、正確に要求分析し、システムを提案することができるのである。

プログラムが書けないからといって、システム開発ができないという短絡的な発想は間違っており、あくまで適材適所を見いだすことをより上層の者は考えなければならないのである。逆に、プログラミングが得意だからといって、システム開発の能力が優れているともいえない。システム開発の工程が工学的にいくら正しくても、それをコントロールする人員の配置が悪け

れば、システムとしての機能は劣化するだけである。

また、ドキュメント作成技術についても同様である。意見はよく出るが、いざドキュメント化すると内容の違うものになっていたり、ひどい場合はドキュメントになっていない者がいる。逆に、意見はそれほど出さないが、要点をまとめた相手に読みやすいドキュメントを作成できる者がいる。先にも述べたが、システム開発工程において唯一明確な共通の仕様はドキュメントだけである。人員配置と同様、ドキュメントを作成する者に関しても、単にその工程のリーダーが行うと決めつけず、適材適所を考えた上で選出すべきである。