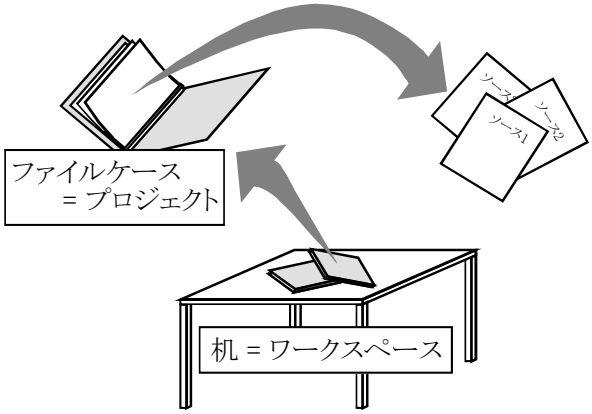
A.　Visual C++ Ver.6 の使い方

1.　Visual C++の構造

Visual C++ではファイルを図 A.1 のように3段階に分けて管理しています。Visual C++を立ち上げただけでは何もない状態になっています。そこで，プログラムを作るためにの中心になる作業場（ワークスペース）を作ることになります。このワークスペースの中にはいくつかのファイルケース（プロジェクト）を置くことができます。このプロジェクトが最終的にはそれぞれのプログラムになります。そして，ファイルケースの中にはプロジェクトを行うために必要な手順書（ソース）や資料（リソース）を入れる，ということです。

では実際に list1.1 のプログラムを作るまでの手順を順に説明していきます。



図A.1　ワークスペース・プロジェクト・ソースの関係

2.　ワークスペースを作る

初めに作業場となるワークスペースを作ってみましょう。[ファイル(F)]－[新規作成(N)...]を選ぶと，図 A.2 のようなダイアログが出てきます。そこで，図の丸で囲った[ワークスペース]というタグを選択します。「ブランクワークスペース」という項目が見えると思います。これは「プロジェクトも何もないワークスペースをこれから作成する」ということです。基本となる[位置](ディレクトリ)と[ワークスペース名]を入力して，OK を押すと新しい空のワークスペースが作成されます。例えば，位置を「F:¥USER¥」，ワークスペース名を「CHAP1」として，ワークスペースを作ってみましょう。（「CHAP1」と入力すると「F:¥USER¥」の後ろに，自動的に「CHAP1」が追加されます）

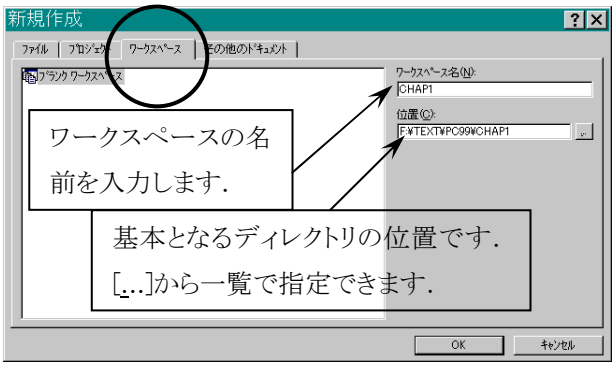


図 A.2　新規作成ダイアログ（ワークスペース）

図 A.3 は作成された新規ワークスペースです。「ワークスペース'CHAP1':0 プロジェクト」と書いてあるのは，今はまだプロジェクトがないということを示しています。では，この図で各部の名称と機能を確認しておきましょう。

2.1.　FileView

これはワークスペース内のファイル構成を表示します。ここに新しいプロジェクトを作成していくことになります。プロジェクトが作成されるとその中に「Source File」，「Header File」，「Resource File」というフォルダが作成され，その中に実際にプロジェクトを構成するファイルを入れていくことになります。

2.2.　ClassView

これはワークスペース内のクラス構成を表示します。C 言語では関係ないです。C++言語を使うようなアプリケーションを作成する時は，クラスの構成，関係などが樹形図で表示されます。

2.3.　ResourceView

これはワークスペース内のリソース構成を表示します。リソースとはアイコン，ダイアログ，メニュー，カーソル，ビットマップ，ツールバー，アクセラレータ（ショートカットキー），ストリングテーブルなど，Windows アプリケーションに必要な構成要素を示します。

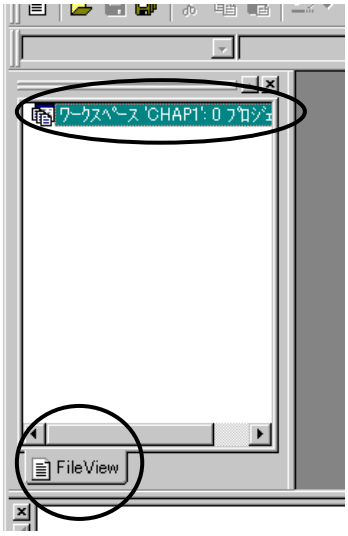


図 A.3　新規ワークスペース

3.　プロジェクトを作る

ワークスペースができたので，その中にプロジェクトを作ってみましょう。[ファイル(F)]－[新規作成(N)...]で新規作成ダイアログを出します。今度は，プロジェクトを作成するので，[プロジェクト]タグを選択します。たくさんの項目が並んでいますが，これがこれから作成するアプリケーションの性質を決定します。良く使うものは以下のものです。

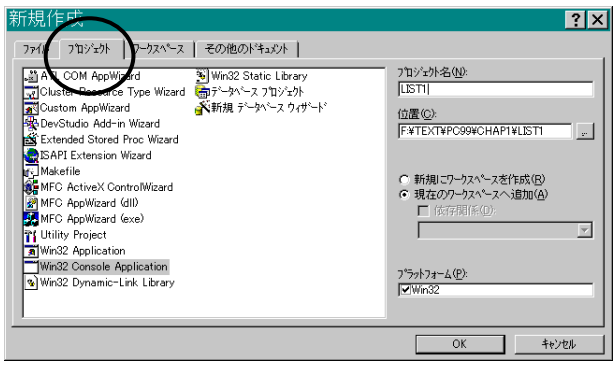


図 A.4　新規作成ダイアログ(プロジェクト)

Win32 Application － 一般的な Windows アプリケーションです。

Win32 Console Application － DOS 窓で動くアプリケーションです。

Win32 Dynamic-Link Library － DLL と呼ばれるライブラリ(関数の集合体)です。必要な時に動的に読み込まれ，要らなくなったらメモリから開放されます。読み込むアプリケーションと DLL のバージョンがあわないと正常動作しない時があります。

Win32 Static-Link Library － リンク時にプログラム中に取り込まれるライブラリです。プログラムと一体化しているため，バージョンの不一致などを気にする必要がなくなりますが，アプリケーションそのものが大きくなり，メモリの消費も大きくなります。

今回はコンソール用のアプリケーションを作成するので，[Win32 Console Application]を選択します。ここで[位置]を確認しておきましょう。ワークスペースを作成したディレクトリ(今回の場合なら「F:¥USER¥CHAP1¥」)になっていればOK です。なっていない場合は[...]で指定しておきましょう。次に，[プロジェクト名]を入力します。例えば，「LIST1」と入力します。すると，[位置]で指定したディレクトリに「LIST1」を追加したディレクトリ名が自動的に入力されます。あとは，せっかく作ったワークスペースに追加するわけですから，[現在のワークスペースに追加(A)]を選択して，[OK]を押します。すると，図 A.5 の「Win32 Console Application」というダイアログが出てきます。

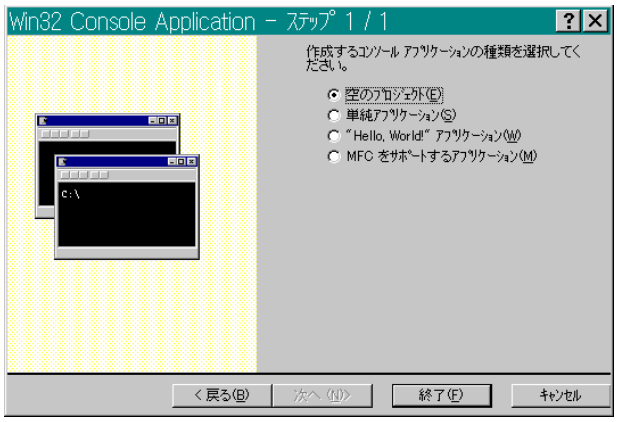


図 A.5　Win32 Console Application の種類

空のプロジェクト － プロジェクトのみを作成します。

単純アプリケーション － プロジェクトと共にソースのひな形を作ってくれます。

"Hello, World!"アプリケーション － "Hello, World!"と表示するアプリケーションのプロジェクトとソースを作ってくれます。基本構造を知るにはよいでしょう。それ以外の使い道はありません。

MFC をサポートするアプリケーション － MFC(Microsoft Foundation Class)を使うことのできるアプリケーションのプロジェクトとソースのひな形を作ってくれます。MFC の詳細は Help でも見てください。

今回は基本を知るためにも「空のプロジェクト」を選択しましょう。普段は「単純アプリケーション」の方が使いやすいです。「終了」を押すと，ワークスペースが図 A.6 のようにプロジェクト数が 1 になり，LIST1 ファイルというプロジェクトが作成されます。LIST1 ファイルの下の3つのフォルダが見えない時は，左にある「+」を押すと，見えるようになります。ここまできたら，あとはソースファイルを書くだけです。

図 A.6　新規プロジェクト

4.　ソースファイルを作る

さて，下準備が終わったので，実際にソースを書いてプログラムらしくしましょう。ソースファイルを作るには，ワークスペースの「Source Files」を選択して，[ファイル(F)]－[新規作成(N)...]です。すると今度は，図 A.7 のような新規作成ダイアログが現れます。ソースファイルを書くのですから，「C++ソースファイル」を選択して，[ファイル名]を入力します。(C じゃなく C++になっているのは気にしなくてもかまいません。気になる人は「LIST1.C」のようにファイル名に拡張子も付けて書いてください。)[OK]を押すと，「LIST1cpp」というファイルが作成され，右側のウィンドウが真っ白になります。ここにプログラムのソースを書いていくことになります。list1.1 を書くと図 A.8 のような感じになります。

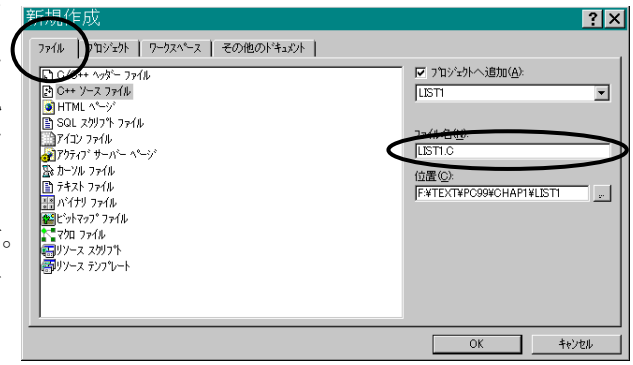


図 A.7　新規作成ダイアログ(ファイル)

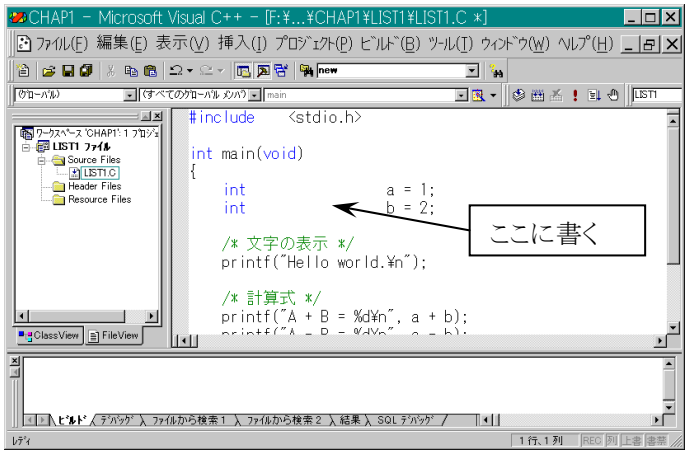


図 A.8　ソースを書いたところ

C/C++言語はスペースや改行を無視します。そのため，これらをうまく使うことで見やすいソースを書くことができます。見やすいソースはバグ(プログラムの問題点)を見つけやすくし，開発の効率を挙げるために役立ちます。一般的なスタイルを挙げると，

1) インデント(字下げ)する。

「{」から「}」で囲まれたブロックや関数の範囲を明確にすることで，有効範囲やループの始端・終端をはっきりさせます。

2) 処理の区切りに改行を入れる。

これもプログラム全体の流れをつかみやすくします。

3) 「,」の後ろ(場合によっては「(」,「)」の前後)にスペースを入れる。

式が繋がっているとどこからどこまでが変数かわかりづらくなります。適当にスペースを入れることで，それを解消します。

4) 長い行は適当なところで改行する。

ソースを書いている画面の横幅はそれほど大きくありません。横にはみ出してしまうと，スクロールした時にバグを見落としてしまう原因となります。

5) 一つの関数は 20～30 行程度に納め，適度に関数に分ける。

あまり長い関数だと初めの方で行った処理を忘れたりして，新たなバグを生む原因となります。うまく関数を利用して見通しの良いソースを作るようにしましょう。

6) コメントをうまく利用する

適度にコメントを入れるようにしましょう。後で見た時の理解の手助けになります。

5.　ビルドする

作成したプロジェクト内のソースをそれぞれコンパイルし，リンクすることを「ビルド」と言います。ビルドすることで，実行ファイルが作成されます。今回作成した list1.1 もビルドして，きちんと書けているか確認をしてみます。

ビルドするには図 A.9 のビルドメニューから，[ビルド(B)]－[ビルド(B) List1.exe]を選択します。すると，下の方の「アウトプットウィンドウ」に途中経過が表示されます。もし，

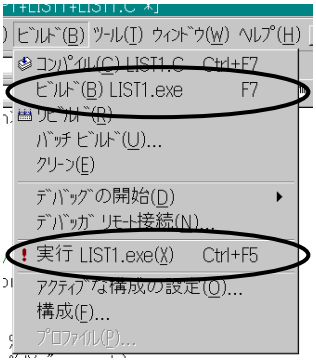


図 A.9　ビルドメニュー

うまくいくと図 A.10 のように表示されます。書き間違いやエラーがあるとアウトプットウィンドウにエラーあった行番号とエラーの内容が表示されます。エラーをダブルクリックすると，そのエラーのあった行にジャンプしてくれるので，修正して再度ビルドしてみましょう。

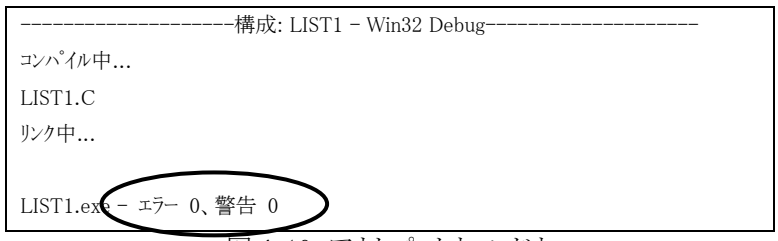


図 A.10　アウトプットウィンドウ

6.　実行する

さて，ビルドがうまくいったら実行してみましょう。[ビルド(B)]－[実行 LIST1.exe(E)]を選択すると，Win32 コンソールが開いて，図 A.11 のような実行結果が表示されます。予想通りの答えになっているでしょうか？予想通りの答えになっていない時はソースのどこかに間違いがあるということです。このように，実行段階で間違いを発見するとその原因がどこなのか見つけるのがとても困難になります。また，結果が予想のつかない場合などは，間違いそのものを見つけることが困難になります。できるだけ，見やすいソースを書くことがバグを減らす一番のポイントです。

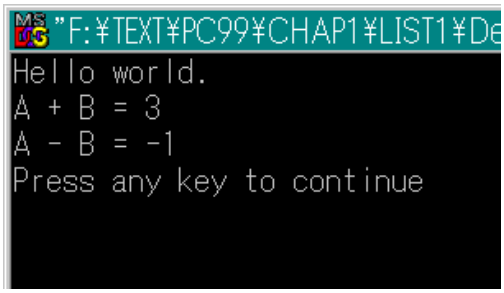


図 A.11　実行結果

7.　さらに

7.1.　Debug モードと Release モード

図 A.10 のアウトプットウィンドウの表示で Win32 Debug と出ています。これは Debug(バグ取り)モードでコンパイルしているという意味です。この Debug モードとは実行ファイルの中に Debug 用のコードを埋めこみ，実行途中の状態をモニタできるようにしているということです。大きなプログラムでバグがあった時などに有効です。しかし，プログラムサイズが大きくなるという欠点もあります。そこで，実際に実行するための実行ファイルを作るためのモードが Release モードです。Release モードに変更するには，[ビルド(B)]－[アクティブな構成の設定(O)...]を選択して，「プロジェクトの標準構成ダイアログ」から Release と書いてあるものを選択します。

7.2.　ワークプレースに複数のプロジェクトを共存させる

練習問題毎にワークプレースを作成するのは不便なので，1つのワークプレースに複数のプロジェクトを共存させましょう。方法は簡単です。すでにプロジェクトが 1 つあるワークプレースに，「3.プロジェクトを作る」の方法でプロジェクトを追加するだけです。

7.3.　デバッグ

ひとたび実行ファイルになってしまったら，バグを探すのは大変だということはすでに書きましたが，では実際にそのようなときはどうすればいいのでしょうか。そのような時のデバッグ法を2つほど紹介します。

1）値の表示

怪しいところの値を表示してみるという方法です。どのあたりでおかしくなっているか判っている時に便利です。

2）デバッガを使う

[ビルド(B)]－[デバッグの開始(D)]－[ステップイン(I)]からアプリケーションを実行すると，デバッグモードでの実行になります。1 行ごと実行したり，ブレークポイント(停止位置)まで実行したりしながら，値を見ることができます。