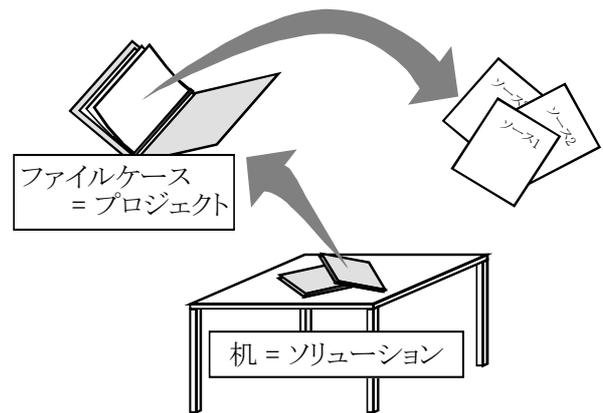


B. Visual Studio .NET の使い方

1. Visual Studio .NET の構造

Visual Studio .NET 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 Studio .NET 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の中心になる作業場(ソリューション)を作ることになります。このソリューション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B.1 ソリューション・プロジェクト・ソースの関係

2. ソリューションを作る

初めに作業場となるソリューションを作ってみましょう。[ファイル(F)]-[新規作成(N)]-[空のソリューション(B)...]を選ぶと、図 B.2 のようなダイアログが出てきます。基本となる[位置](ディレクトリ)と[プロジェクト名]にソリューション名を入力して、OK を押すと新しい空のソリューションが作成されます。例えば、位置を「F:\USER¥」、ソリューション名を「CHAP1」として、ソリューション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 B.3 は作成された新規ソリューションです。「ソリューション' 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。



ソリューションの名前を入力します。

基本となるディレクトリの位置です。[参照]から一覧で指定できます。

図 B.2 新規作成ダイアログ (ソリューション)

2.1. ソリューションエクスプローラ

これはソリューション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「参照設定」、「ソースファイル」、「ヘッダーファイル」、「リソースファイル」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. クラスビュー

これはソリューション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. リソースビュー

これはソリューション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)、string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。

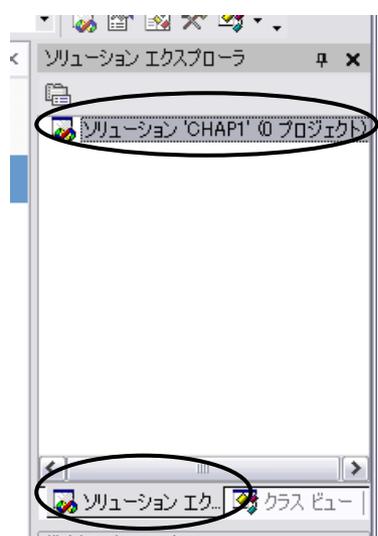


図 B.3 新規ソリューション

3. プロジェクトを作る

ソリューションができたので、その中にプロジェクトを作ってみましょう。ソリューションエクスプローラの CHAP1 ソリューションでマウスの右ボタンをクリックし、[追加(D)]-[新しいプロジェクト(N)...]で新しいプロジェクトの追加ダイアログを出します。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。

Win32 プロジェクト - 一般的な Windows アプリケーションです。

Win32 コンソールプロジェクト - コマンドプロンプトで動くアプリケーションです。

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 コンソールプロジェクト]を選択します。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。[OK]を押すと、アプリケーションウィザードダイアログが出てきます。ここで少し設定をしておきます。ダイアログ左側の[アプリケーションの設定]を押すと図 B.5 のような表示になります。アプリケーションの種類に注意しましょう。上の2つ、Windows アプリケーションとコンソールアプリケーションは先ほどの説明の通りです。その下の2つは、

DLL (Dynamic Link Library) - 実行時の必要ときに動的に読み込まれるライブラリ(関数の集合体)です。必要に応じて入れ替えが可能で、複数のアプリケーションで共通の機能を共有することができます。

スタティックライブラリ - 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。

追加のオプションは[空のプロジェクト(E)]をチェックしないと、プロジェクトと共にソースのひな形を作ってくれますが、今回は基本を知るためにも「空のプロジェクト(E)」をチェックしましょう。プロジェクトが図 B.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。実際にこれから作成するファイルはこれらのフォルダに分類されます。必要に応じてフォルダを追加することもできます。ここまできたら、あとはソースファイルを書くだけです。

3.1. プロジェクトのプロパティ

プロジェクトのプロパティを用いて作成するアプリケーションに関する様々な設定をすることができます。ソリューションエクスプローラ内の LIST1 プロジェクトを選択し、マウスの右ボタンメニューからプロパティを選択します。Visual Studio.NET は 64bit システム対応となっていますが、今回の授業では特に必要ないので、[構成プロパティ]-[C/C++]-[全般]-[64ビット移植への対応]を[いいえ]に変更しておきましょう。



図 B.4 新しいプロジェクトの追加ダイアログ



図 B.5 Win32 Console Application の種類



図 B.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、ソースファイルフォルダでマウスの右ボタンクリックをして、[追加(D)]-[新しい項目の追加(W)...]です。すると今度は、図 B.7 のような新しい項目の追加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++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括弧も付けて書いてください。)[OK]を押すと、「LIST1.cpp」というファイルが作成され、LIST1 という真っ白いウィンドウが現れます。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 B.8 のような感じになります。

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、開発の効率を上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。入力には Tab キーを使います。

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てたりして、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 B.9 のビルドメニューから、[ビルド(B)]-[LIST1 のビルド(U)]を

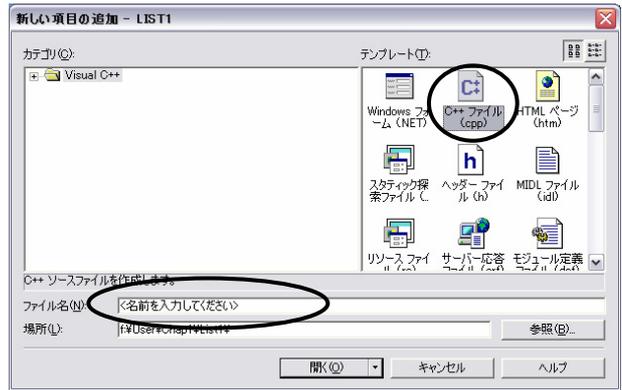


図 B.7 新規作成ダイアログ(ファイル)

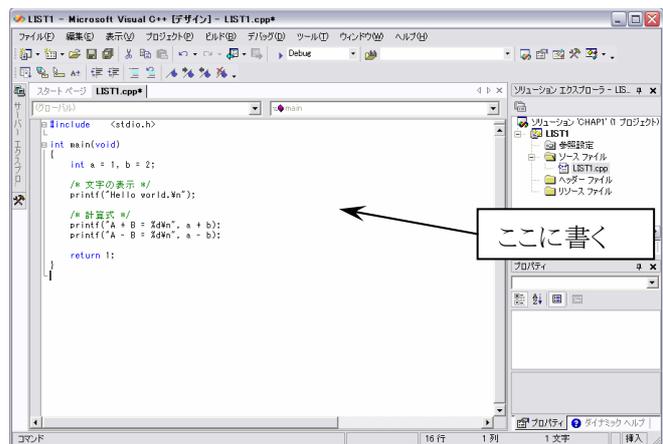


図 B.8 ソースを書いたところ



図 B.9 ビルドメニュー

選択します。すると、下の方の「アウトプットウインドウ」に途中経過が表示されます。もし、うまくいくと図 B.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イ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。

```

----- ビルド開始 : プロジェクト : LIST1, 構成 : Debug Win32 -----
コンパイルしています...
LIST1.cpp
リンクしています...

ビルドログは "file:///f:%User%Chap1%List1%Debug%BuildLog.htm" に保存されました。
LIST1 - エラー 0、警告 0

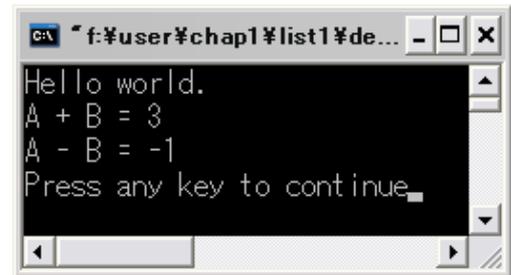
----- 終了 -----
ビルド 1 正常終了、0 失敗、0 スキップ

```

図 B.10 アウトプットウインドウ

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。[デバッグ(D)]-[デバッグなしで開始(G)]を選択すると、Win32 コンソールが開いて、図 B.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てい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。



```

C:\ \ f:%user%chap1%list1%de...
Hello world.
A + B = 3
A - B = -1
Press any key to continue.

```

図 B.11 実行結果

7. さらに

7.1. Debug モードと Release モード

図 B.10 のアウトプットウインドウの表示で Debug Win32 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)]-[構成マネージャ(O)...]を選択して、「アクティブソリューション構成(A)」から Release と書いてあるものを選択します。

7.2. ソリューションに複数のプロジェクトを共存させる

練習問題毎にソリューションを作成するのは不便なので、1つのソリューション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 つあるソリューションに、「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

[デバッグ(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。