# 01 MFCプロジェクトの作成

**Keyword**
- MFC
- MFCアプリケーションウィザード

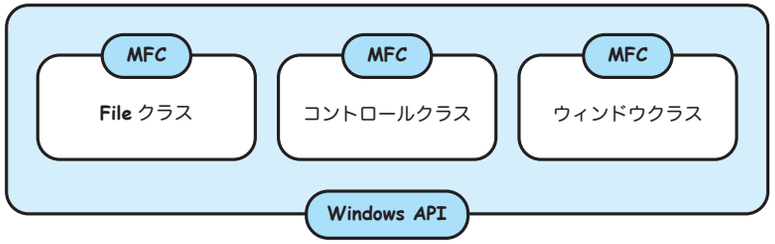
## MFCとは？

**MFC（Microsoft Foundation Class）**は、Visual C++のアンマネージコード用のクラスライブラリです。

Visual C++でアンマネージコードのアプリケーションを作成するには基本的にWindows APIを使用することになりますが、Windows APIはすべての関数が同列に見えるため、作成する目的にあったWindows APIを探すのが大変です。

MFCはオブジェクト指向で作成されており、Windows APIを目的毎にクラス単位でまとめることで必要最小限なプログラミング量でアプリケーションを作成できるように工夫されています（**図4-1**）。

**▼図4-1　Windows APIとMFCの関係**




本章は、MFCでアプリケーションを作成するための手順を学んでいきます。





## MFCアプリケーションのプロジェクトを作成する

**1章**でVisual Studioでプロジェクトを作成する方法を学びました。これからMFCアプリケーション用プロジェクトの作成手順を見ていきましょう。

Visual Studioの［ファイル］メニューから［新規作成］→［プロジェクト］を選択し、「新しいプロジェクトの作成」画面を表示します。「新しいプロジェクトの作成」画面のフィルターに［C++］［Windows］［デスクトップ］を選択し、［MFCアプリ］を選択して、［次へ］ボタンを押します（**図4-2**）。

**▼図4-2　「新しいプロジェクトの作成」画面**


「新しいプロジェクトを構成します」画面で、［プロジェクト名］に「MFCApplication」、［場所］に保存する場所を入力し、［ソリューションとプロジェクトを同じディレクトリに配置する］をチェックしないで、［作成］ボタンを押します（**図4-3**）。

# 01 クラスウィザード

Keyword ☑コードウィザード ☑クラスウィザード

## コードウィザードとは？

**コードウィザード**とは、ソースコードの中にクラスやメンバー関数、メンバー変数などを対話形式で追加することができるVisual Studioの機能です。

MFCアプリケーションに機能を追加するにはアプリケーション作成者がコードを記述することが必要になりますが、コードウィザードを用いることでコードすべてを記述する必要がなくなります。

MFCアプリケーションにコードウィザードを利用してコードを追加すると、適切なコードが追加されるので、正しいコードの作成方法を学ぶことができます。

この章ではコードウィザードを使用して、どのように機能を追加していくのかを見ていきましょう。

## クラスを追加するには？

それではVisual Studioでクラスを追加する方法を見ていきましょう。この章では**4章**で作成したMFCApplication1プロジェクトを使用して説明します。

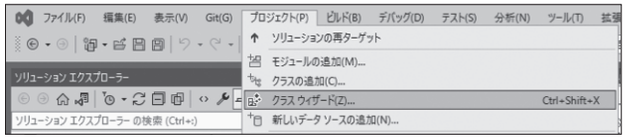
Visual Studio 2022では「**クラスウィザード**」を使用してクラスを追加することができますので、その方法を紹介します。





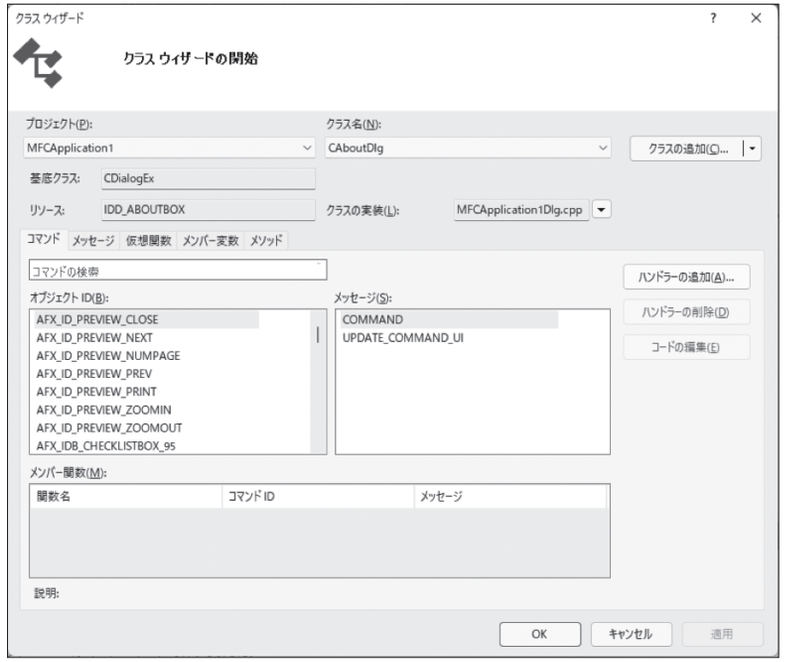
「クラスウィザード」を使用するには、Visual Studioの［プロジェクト］メニューから［クラスウィザード］を選択します（**図5-1**）。

**▼図5-1　クラスウィザードの使用**

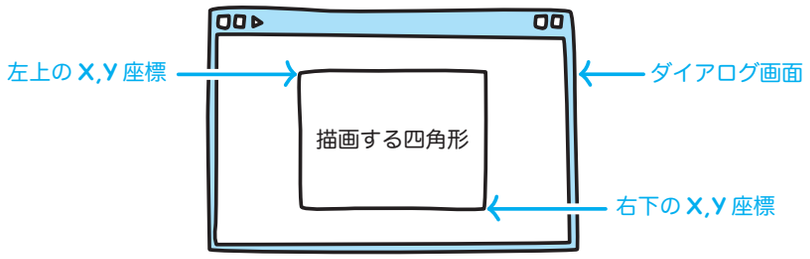



選択すると**図5-2**のような「クラスウィザード」画面が表示されます。

**▼図5-2　「クラスウィザード」画面**



▼図9-5　Rectangle()による座標イメージ



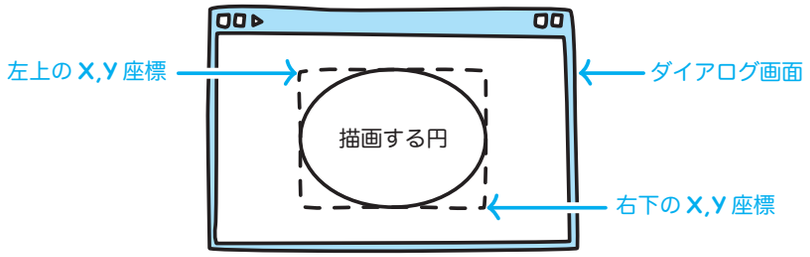

CPaintDCクラスで円を描画するには、**Ellipse関数**を使用します（**構文9-8**）。

**構　文**　**9-8　円を描画する**

```
インスタンス名.Ellipse(int 左上のX座標, 左上のint Y座標, int 右下のX座標, 左下のint Y座標);
```

構文9-8で指定する座標の描画イメージは、**図9-6**のようになります。

▼図9-6　Ellipse()による座標イメージ








# 四角形や円を描画してみよう

それでは四角形や円を描くプログラムを作成してみましょう。MFCダイアログベースのプロジェクト「SampleDrawRectAndCircle」を作成してください。

SampleDrawRectAndCircleDlg.cppを開いて、CSampleDrawRectAndCircleDlgクラスのOnPaint関数に**リスト9-3**のようにコードを追加してください。

▼リスト9-3　四角形と円の描画

```
01: void CSampleDrawRectAndCircleDlg::OnPaint()
02: {
03:     if (IsIconic())
04:     {
          中略
05:     }
06:     else
07:     {
08:         CPaintDC dc(this);
09:         CPen myPen;
10:         myPen.CreatePen(PS_SOLID, 5, RGB(255,0,0));
11:         CPen* pOldPen = dc.SelectObject(&myPen);
12:         dc.Rectangle(10,10,150,100);
13:         dc.Ellipse(180,10,300,100);
14:         dc.SelectObject(pOldPen);
15:         CDialogEx::OnPaint();
16:     }
17: }
```

8～14行目が追加したコードになります。8～11行目は前節で使用しているコードと同じですね。

12行目でRectangle関数を使用して四角形を描画しています。四角形の外枠は10～11行目で作成しているペンで描画されます。

13行目でEllipse関数を使用して円を描画しています。円の外枠は10～11行目で作成しているペンで描画されます。

# 02 スレッドを使ってみよう

**Keyword**　☑AfxBeginThread関数

## スレッドを作成するには？

● **AfxBeginThread 関数**

それではスレッドを作成する方法を学んでいきましょう。

MFC でワーカースレッドを作成するには**AfxBeginThread 関数**を使用します（**構文 12-1**）。

**構　文**　**12-1　ワーカースレッドを作成する**

```
CWinThread* AfxBeginThread(
            AFX_THREADPROC ワーカースレッドの制御関数,
            LPVOID ワーカースレッドの制御関数に渡す引数,
            int 優先度 = THREAD_PRIORITY_NORMAL,
            UINT スタックサイズ = 0,
            DWORD スレッドの制御追加フラグ = 0,
            LPSECURITY_ATTRIBUTES セキュリティ属性 = NULL);
```

**ポイント**

**第三～第六引数に記載されている「= THREAD_PRIORITY_NORMAL」等の記載は、「省略可能な引数で、省略した場合の初期値に右辺の値が入る」ことを示しています。**

「ワーカースレッドの制御関数」は**構文 12-2**のように宣言されたスタティック（静的）関数でなければなりません。





**構　文**　**12-2　ワーカースレッドの制御関数**

```
UINT 関数名(LPVOID 引数);
```

「ワーカースレッドの制御関数に渡す引数」は構文 12-2 に定義されている関数の引数に使用されます。

「優先度」にはスレッドが動作する優先順位を指定します。設定できる値には**表 12-1** のようなものがあります。

**▼表 12-1　スレッドの優先順位**

| 設定値 | 内容 |
|---|---|
| THREAD_PRIORITY_ABOVE_NORMAL | スレッド標準の相対優先順位値より1ポイント高い相対優先順位値を指定します |
| THREAD_PRIORITY_BELOW_NORMAL | スレッド標準の相対優先順位値より1ポイント低い相対優先順位値を指定します |
| THREAD_PRIORITY_HIGHEST | スレッド標準の相対優先順位値より2ポイント高い相対優先順位値を指定します |
| THREAD_PRIORITY_LOWEST | スレッド標準の相対優先順位値より2ポイント低い相対優先順位値を指定します |
| THREAD_PRIORITY_NORMAL | スレッド標準の相対優先順位値を指定します |

「スタックサイズ」にはスレッドのスタックサイズを設定することができます。**スタック**とはスレッド内部で管理用に使用するメモリー領域のことです。デフォルトではプロセスのスタックサイズと同じ値が設定されます。

「スレッドの制御追加フラグ」には**表 12-2** のようなスレッドの制御追加フラグを指定することができます。

**▼表 12-2　スレッドの制御追加フラグ**

| 設定値 | 内容 |
|---|---|
| CREATE_SUSPENDED | スレッド生成時には停止状態になります |
| 0 | スレッド生成時には起動状態になります |

# 04 同期処理

Keyword ☑同期クラス ☑アクセス制御クラス

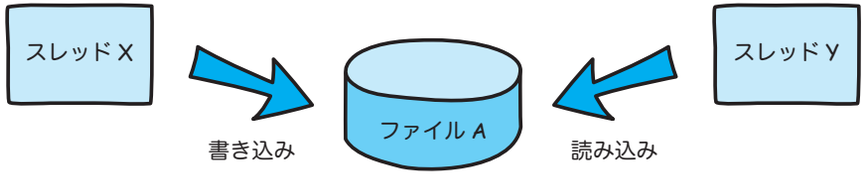
## スレッドの同期処理

前節まではスレッドを1つ作成して使用する方法を学んできました。スレッドを複数作成する場合には、**リソースの競合**に気をつける必要があります。

リソースの競合とは、1つのリソース（資源）に対して複数のスレッドやプロセスがアクセスする行為を指します。**図12-6**ではファイルAをスレッドXとスレッドYが共有していますが、スレッドXがファイルに書き込んでいる最中にスレッドYがファイルを読み込んだとしたら、どのようなことが起きるでしょうか？

ファイルはスレッドXが書き込み中の状態であるため、スレッドYが正しくファイルを読めるかどうかは保証することができません。

▼図12-6　リソースの競合




リソースの競合に対する問題は、1つの資源を共有するプロセスやスレッドが同期して同時にアクセスしないように**排他制御**を行うことにより解決できます。





MFCには同期処理を行うための同期クラスが提供されていますので、本節では同期クラスの使用方法について学んでいきましょう。

## MFCの同期クラス

MFCには**表12-8**のような同期クラスがあります。どの同期クラスを使用するのかは、その目的に応じて選択します。

▼表12-8　MFCの同期クラス

| クラス名 | 内容 |
|---|---|
| CSemaphore | 同時にリソースにアクセスできるスレッドの数を限定する機能を提供する同期クラス |
| CMutex | 複数のプロセスまたはスレッドのうちの1つだけがリソースに排他アクセスできる機能を提供する同期クラス |
| CCriticalSection | 複数のスレッドの1つだけがリソースに排他アクセスできる機能を提供する同期クラス |
| CEvent | イベントが発生したことを、あるスレッドが別のスレッドに通知できる機能を提供する同期クラス |

CSemaphore、CMutex、CCriticalSectionクラスは、複数のプロセスまたはスレッドが1つの共有リソースに対してアクセスを行う場合の排他制御を行うことを目的としており、プロセスまたはスレッドはこれらの同期クラスを通して間接的に同期を取ります。

CEventクラスはスレッド間でイベントを通知し合い、直接同期を取ります。

同期クラスは**アクセス制御クラス**の**CSingleLock**または**CMultiLock**を経由して使用します。共有するリソースが1つの場合はCSingleLock、1つ以上の場合はCMultiLockクラスを使用します。アクセス制御クラスを経由して同期クラスを使用すると、同期クラスの違いをほとんど意識することなく使用することができます。

同期クラスやアクセス制御クラスの使用方法について、学んでいきましょう。

# 共有メモリを使用するには？

共有メモリを使用するには、メモリを共有するプロセスのうち1つのプロセスが共有メモリを作成してから、複数のプロセスが作成された共有メモリにアクセスをします。

共有メモリを作成するには以下の手順で行います。

① ファイルマッピングオブジェクトを作成する
② ファイルマッピングオブジェクトをメモリに割り当てる（マップする）
③ 作成された共有メモリのハンドルを取得する
④ 必要であれば、共有メモリのハンドルから共有メモリのポインタを取得する
⑤ 共有メモリを読み書きする
⑥ 共有メモリを使い終わったら、共有メモリを解放する

共有メモリはファイルの一種として扱います。そのため共有メモリを作成する時には、**ファイルマッピングオブジェクト**を作成してからオブジェクトをメモリにマップします。

共有メモリ作成後に、共有メモリにアクセスするプロセスは共有メモリのハンドルを取得して、前節で説明したReadFile関数やWriteFile関数を使用して読み書きを行います。

ハンドルから共有メモリのポインタを取得して、直接メモリに読み書きを行うこともできます。これらの手順で使用するWindows APIの説明を行います。

**ポイント**

**共有メモリはファイルの一種として扱うため、メモリマップドファイルと言います。**





### ● ファイルマッピングオブジェクトを作成する

ファイルマッピングオブジェクトを作成するには、**CreateFileMapping関数**を使用します（**構文14-11**）。

**構　文**　**14-11　ファイルマッピングオブジェクトの作成**

```
HANDLE CreateFileMapping(
          HANDLE ファイルのハンドル,
          LPSECURITY_ATTRIBUTES セキュリティ記述子,
          DWORD 保護属性,
          DWORD 共有メモリサイズの上位DWORD,
          DWORD 共有メモリサイズの下位DWORD,
          LPCTSTR 共有メモリ名);
```

「ファイルのハンドル」にはファイルのハンドルを設定しますが、共有メモリを作成する場合は-1を設定します。

「セキュリティ記述子」にはファイルマッピングオブジェクトのセキュリティモードを設定することができます。設定しない場合はNULLを指定します。

「保護属性」にはファイルのアクセス権限を設定しますが、共有メモリを作成する場合はPAGE_READWRITE（読み書き権限）を指定します。

「サイズの上位DWORD」と「サイズの下位DWORD」には共有するメモリのサイズを設定します。例えば共有するメモリのサイズが2048バイトであれば、「サイズの上位DWORD」には0、「サイズの下位DWORD」には2048を設定します。

「共有メモリ名」には共有メモリの名称を設定します。

CreateFileMapping関数の処理が成功するとファイルマッピングオブジェクトのハンドルが返り、失敗するとNULLが返ります。

### ● ファイルマッピングオブジェクトをメモリにマップする

作成したファイルマッピングオブジェクトをメモリにマップするには、**MapViewOfFile関数**を使用します（**構文14-12**）。

# 01 ダイナミックリンクライブラリ

**Keyword**
- ☑ ダイナミックリンクライブラリ
- ☑ DLL

## ダイナミックリンクライブラリ（DLL）とは？

**ダイナミックリンクライブラリ（Dynamic Link Library：DLL）**とは、複数のアプリケーションが共通に利用することができるモジュールを集めた、実行可能なファイルです。DLLを使用すると、他のアプリケーションでモジュールを再利用することができるようになり開発効率をあげることができるなどの利点があります。

本章ではDLLの作成と使用方法について学んでいきましょう。

## ダイナミックリンクライブラリはどのように使うの？

ダイナミックリンクライブラリの拡張子は「.dll」であることがほとんどです。ダイナミックリンクライブラリを使用するアプリケーションは.dllファイル内に書かれている必要な関数だけを読み込み、実行します（図15-1）。

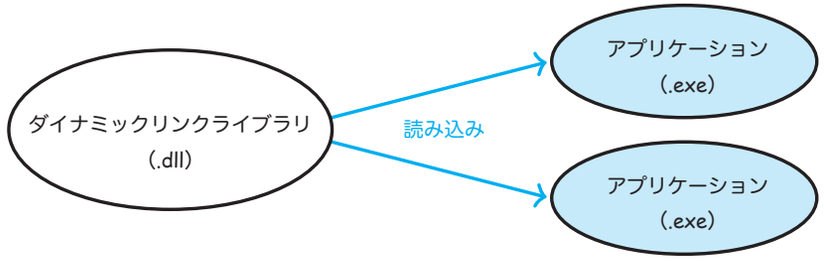
**ポイント**

.dll ファイルは単独で実行することはできません。





▼図15-1　ダイナミックリンクライブラリの利用




## MFC DLLプロジェクトの作成

それではMFCのDLLプロジェクトを作成する方法を見てみましょう。

Visual Studioの［ファイル］メニューから［新規作成］→［プロジェクト］を選択し、「新しいプロジェクトの作成」画面を表示します。「新しいプロジェクトの作成」画面のフィルターに［C++］［Windows］［ライブラリ］を選択し、[MFCダイナミックリンクライブラリ]を選択して[次へ]ボタンを押します（図15-2）。

▼図15-2　「新しいプロジェクトの作成」画面

# 04 暗黙的なリンクを使用しDLLを利用する

**Keyword** ☑インポートライブラリ

## DLLを利用したアプリケーションの作成

それでは前節で作成したIsbnDll.dllを利用して、入力されたISBNコードが正しいかをチェックするアプリケーションを作成していきましょう。

### ● ダイアログ画面の作成

これから**図15-15**に示すダイアログ画面を作成していきます。

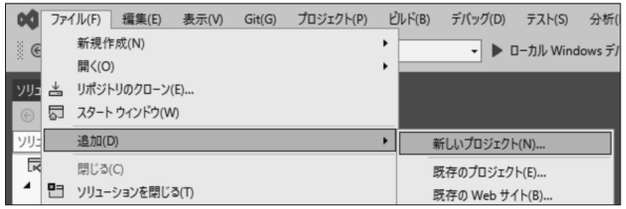
**▼図15-15　SampleIsbnダイアログ画面**


前節で使用したSampleDllソリューションに、図15-13のダイアログ画面を作成するために新しいプロジェクトを追加します。Visual Studioの［ファイル］メニューから［追加］→［新しいプロジェクト］を選択します（**図15-16**）。





**▼図15-16　新しいプロジェクトの作成**

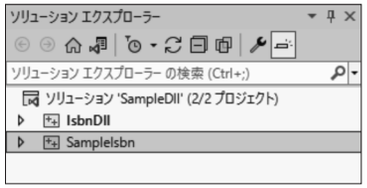



「新しいプロジェクトを追加」画面が表示されたら［MFCアプリ］を選択し、プロジェクト名に「SampleIsbn」を入力して作成します。

「MFCアプリケーション」画面が表示されたら、[アプリケーションの種類]では[ダイアログベース]を選択してください。

プロジェクトの作成が完了すると、ソリューションエクスプローラには2つのプロジェクトが表示されます（**図15-17**）。

**▼図15-17　SampleIsbnプロジェクトを追加した後のSampleDllソリューション**




プロジェクトの作成が完了したら、図15-15を参考にダイアログエディターでダイアログ画面を編集しましょう。

### ● スタティックテキストコントロールを追加する

スタティックテキストコントロールをダイアログ画面に1つ貼り付けます。Captionプロパティに「ISBN」を設定してください。

### ● エディットボックスコントロールを追加する

エディットボックスコントロールをダイアログ画面に1つ貼り付けます。作成したエディットボックスコントロールのIDはIDC_EDIT1として説明していきます。

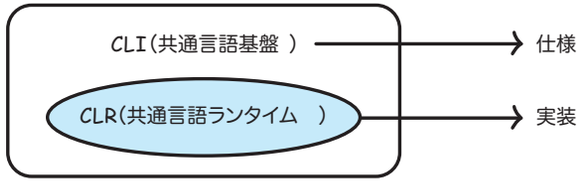
# 01 C++/CLIとは

Keyword ☑C++/CLI ☑名前空間 ☑値型 ☑参照型

## C++/CLIとは？

**C++/CLI(Common Language Infrastructure:共通言語基盤)**は、.NET上で動作するプログラムを作成するために、C++を拡張した言語です。
CLIとは.NETのプログラム実行環境の仕様を定義したもので、ECMA(ヨーロッパ電子計算機工業会)によって標準化されています。CLIをMicrosoftによって実装したものが、第1章で説明したCLR(共通言語ランタイム)です(図16-1)。

▼図16-1　CLIとCLR




CLIの概念の説明はややこしく感じられたと思いますが、要は「Visual C++でマネージコードを記述する場合には、C++/CLIというプログラミング言語で記述する必要がある」ということを覚えておいてください。





## Visual C++で作成する、NETアプリの制約事項

Visual C++で.NETに対応したマネージドコードのアプリを作成する場合、以下のような制約事項があります。

- Visual C++では直接実行可能なファイル(拡張子.exe)を作成することはできません。ダイナミックリンクライブラリ(拡張子.dll)は作成可能です。
- .NET対応のアプリはWindows、Linux、Mac OSなどの複数OSをターゲットとしていますが、Visual C++を使用する場合は、Windows OSのみをターゲットとすることになります。
- Visual C++で.NETに対応したWindowsフォームやWPF(Windows Platform Foundation)アプリを作成する場合、フォームデザイナーやXAMLデザイナーを使用することができません。

上記の制約事項はありますが、これからVisual C++ならではのプログラミング手法を説明していきますので、その方法を学んでいきましょう。基礎知識を学んだ後で、マネージコードを利用したアプリを作成していきます。

## 名前空間とは？

**名前空間(namespace)**とは、使用したいクラスやメソッドなどが他の同名クラスや同名メソッドなどにより名前が衝突しないように、一意となるような名前を設定する概念のことです。

図16-2は複数の同じ名前のクラスABCが作成され、利用する側に提供されていることを示した例です。クラスABCを利用する立場では名前だけで利用したいクラスを解決すると、コード中に利用しているクラスがどのクラスなのかを区別することが困難です。

# 02 Windows App SDK

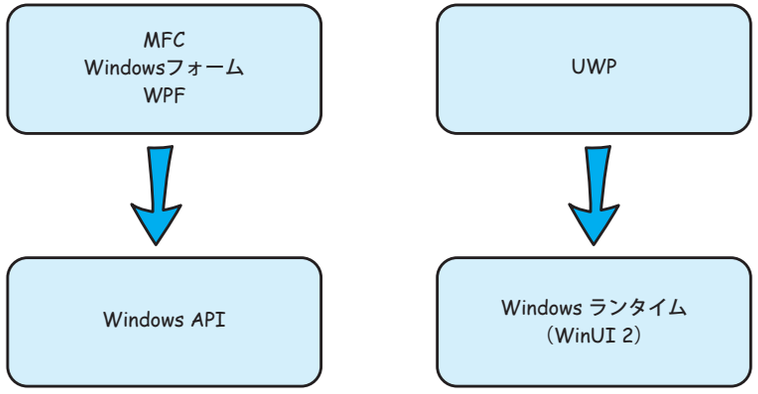
**Keyword**
- ☑ Windows App SDK
- ☑ WinUI3

## Windows App SDKとWinUI3

Windowsデスクトップアプリを開発するには、MFC、Windowsフォーム、WPFなど従来のWin32アプリを使用する方法と、Microsoftストアアプリ開発用の**UWP（Universal Windows Platform）**を使用する方法があります。

UWPアプリの開発にはWindowsランタイムを使用しますが、従来のWin32アプリから使用することができませんでした。また、Win32アプリの開発にはWindows 32 APIを使用しますが、UWPアプリからは使用することができませんでした（**図17-1**）。

**▼図17-1　Windows App SDK登場以前のWindowsデスクトップアプリ開発**

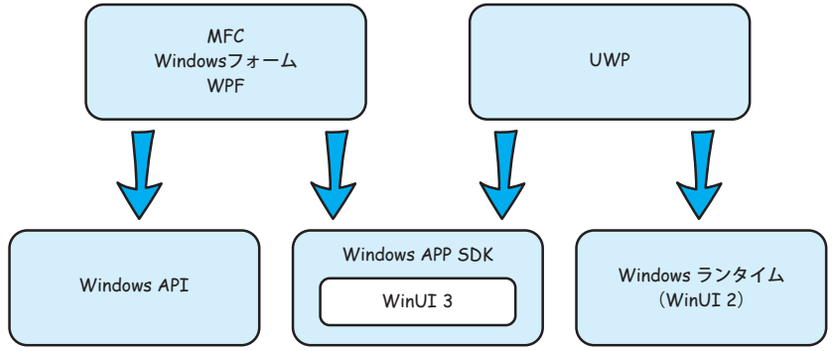







このことからWin32アプリとUWPでは開発のアプローチが異なっていましたが、どちらの技術を使用しても同じように開発できるようにすることを目的として、Microsoftは2021年に**Windows App SDK**を発表しました。Windows App SDKの中でUI（User Interface）関連のランタイムは**WinUI 3**と呼ばれています（**図17-2**）。

**▼図17-2　Windows App SDK登場後のWindowsデスクトップアプリ開発**




**ポイント**

**UWPのみが使用できるUI関連のWindowsランタイムはWinUI 2と呼ばれています。**

Windows App SDKは登場したばかりで、今後も機能の追加が計画されています。これからのWindowsデスクトップアプリ開発の主流はWindows App SDKに移っていくことになるでしょう。

## WinUI 3プロジェクトを作成してみよう

それではVisual C++でWinUI 3に対応したアプリを作成してみましょう。Visual Studioの［ファイル］メニューから［新規作成］→［プロジェクト］を選