# C++Builder Unicode対応

C++Builderの RAD Studio 2009 バージョン以降で、完全にUnicodeをサポートして内部の文字列の扱いが変更になっている。特に、それ以前のバージョン（私の場合は C++Builder 5のソースを動かそうとした場合に、色々と留意する点があるので備忘録を残す。

## 文字列の扱い変更対応

UnicodeString型が追加され、従来の String型，Char型，PChar型の定義が以下のように変更されている。（参考1.資料参照）

| 型 | 意味 | 備考 |
|---|---|---|
| String | UnicodeString型でtypedef | UTF-16文字列 |
| Char | wchar_tでtypedef | UTF-16文字 |
| PChar | wchar_t *でtypedef | |

### 準備

ソースファイルは全て UTF-8 のBOM付きに変換して保存しておく。

### 文字列及び文字の型移行

従来のAnsiStringやchar型も引き続き使えるが、将来の事も考えて以下のようにソースを修正。

| 移行前 | 移行後 | 備考 |
|---|---|---|
| AnsiString | String | あえてUnicodeStringにしない |
| char | Char | |
| char * | PChar | |

### 文字列定数及び文字定数の修正

文字列と文字定数の前には、以下のように L 文字を付加する。

```
String str = L"文字列";
Char c = L'あ';
```

### C標準関数の修正

fopenなどのC標準関数は、char型を要求しているのでコンパイルエラーとなる。以下のように、_wを付加した関数名に変更する。

```
String fname = L"test.txt";
```

```
fp = fopen(fname.c_str(), "rt");
       ↓
fp = _wfopen(fname.c_str(), L"rt");
```

- 「fopen_wfopen」ヘルプページ

その他対応の関数名は「マップ関数」ヘルプページを参照し下記のように変更。

| 移行前 | 移行後 | リンク | 備考 |
|---|---|---|---|
| fgetc | fgetwc | fgetc fgetwc | 終了時は WEOF が戻る |
| sprintf | swprintf | sprintf swprintf | |
| toupper | towupper | toupper _mbctoupper towupper | |
| tolower | towlower | tolower _mbctolower towlower | |

# その他注意点

## 添え字番号の開始は1

String型の文字列は文字配列のようにアクセスできるが、添え字の開始番号は 0 でなくて 1～ なので注意。（0でアクセスすると例外トラップ）

```
String text = L"あいうえお";
Char c = text[1];          /* 'あ'が取り出される */
```

# 所感

最初は「かなり思い切った仕様変更だな」と戸惑ったが、ソース修正は必要だが上記要領で案外スムーズに動いた。

# 参考


1. Delphiにおける一般的なUnicodeへの移行テクニック
2. RAD Studio における Unicode
3. Delphi 2009ではじめるUnicodeアプリケーション - 既存コード移行のポイント -
4. 旧バージョンからの移行
5. C++Builder2009におけるStringの変更による既存コードの修正
6. C++Builder文字列変換
7. Cannot convert 'const char*' to 'WCHAR*' in argument passing
8. System.UnicodeString のメソッド
9. UTF-8を使う
10. Read a file character by character/UTF8

From:
https://yokoweb.net/dokuwiki/ - **新石器Wiki**

Permanent link:
**https://yokoweb.net/dokuwiki/programing/cpp_builder/cpp_builder-unicode**

Last update: **2026/07/28 13:16**