# 虚幻中的代码命名规则

帕斯卡命名法

1. 模板类以T作为前缀，比如TArray,TMap,TSet(仨容器)
2. UObject派生类都是以U做前缀
3. AActor派生类都以A做前缀
4. SWidget派生类都是以S做前缀
5. 全局对象使用G开头，如GEngine
6. 抽象接口以I前缀
7. 枚举以E开头
8. 布尔值变量以b前缀
9. 其他的大部分自定义类型数据（结构体，类）以F开头
10. typedef的以原型名称前缀为准，如typedef TArray FArrayOfMyTypes；

# UEC++中的基本数据类型

为了平台拓展性，UE对于基本C++中的数据类型做了深度定义，方便平台拓展，增加了UE的移植便捷性

禁止在UE中使用C++的基本数据类型。

* bool 代表布尔值（永远不要假设布尔值的大小），bool不会被编译
* TCHAR 代表字符型（永远不要假设TCHAR的大小）
* uint8 代表无符号字节（占一个字节）
* int8 代表有符号字节（一个字节）
* int16 有符号“短整型”（2个字节）
* uint16 无符号“短整型”（2个字节）
* uint32 无符号整型 （4个字节）
* int32 有符号整型（4个字节）
* uint64 无符号长整型（8个字节）
* int64 有符号长整型（8个字节）
* float 代表单精度浮点数（占4个字节）
* double代表双精度浮点型（8个字节）

# UE中的字符串对象

* FName，资源命名字符串，FName通过一个轻型系统使用字符串。在此系统中，特定字符串即使会被重复使用，在数据表中也之存储一份。FName不区分大小写（大小写不是比较依据）。
* FText表示一个显示字符串，用户的显示文本都需要由FText进行处理。支持文本格式化，不提供修改函数，无法进行内容修改。
* FString，可以被操作的字符串。开销大于其他类型字符串。

### FString

最常用的文本串对象，可以被修改和操作，重载了各种操作运算符。支持长度获取，大小写转换，格式化文本，数据转换等操作。效率低开销大，但使用方便。

构建方式

FString Name(TEXT("OK"));

FString N1 = FString(TEXT("OK"));

两种构建方式均可以使用

注意：构建字符串对象时，不能省略宏TEXT，因为我们需要借助与平台无关的字符串进行构建，应该使用UE字符串

##### 输出字符串

1. 窗口输出支持直接输出FString
2. UE\_LOG输出需要使用占位符进行格式化

FString N1 = FString(TEXT("OK1"));

UE\_LOG(LogTemp, Log, TEXT("%s"), \*N1);

**需要注意，运算符\*在这里不是解地址，是重载后的操作，因为N1是栈对象，\*运算符在这里是重载后的运算符**

##### 对比

查找字符串中是否包含某个串，可以使用**Contains函数，如果能查找到则返回true，反之返回false**

TestHUDString.Contains(TEXT("Test"), ESearchCase::CaseSensitive, ESearchDir::FromEnd);

参数：

·查找文本内容，切记使用TCHAR进行编码

·是否进行大小写忽略

·查找方向

##### 检索字符串中包含子串的位置

TestHUDString.Find(TEXT("test"), ESearchCase::CaseSensitive, ESearchDir::FromEnd, 10);

参数：

·查找的目标字符串

·是否大小写忽略

·查找方向

·查找起始位置，正向从头开始数，反向从尾开始数

·返回查找到的位置，均从首算起，查找不到返回-1

##### 裁切字符串

使用给定的分隔符（FString类型）进行数据拆分，只能左右拆分

FString left;

FString right;

m1.Split(TEXT("|"), &left, &right);//此函数返回true和false，如果包含分隔符，拆分成功返回true，反之返回false

##### **支持运算符重载 + +=构建新的字符串**

##### 检查给定的串是否为空

ma.IsEmpty()

##### 路径重载运算符/

用来组合路径

FString patch = F1 / F2;

##### 支持占位符格式化文本

与C++print占位符使用相同使用FString：：Printf进行字符串构建

FString::Printf(TEXT("%02d:%02d"), NumMinutes, NumSeconds)

%02d 意思就是最少补全2位数，如果不够用0补，例如数据是5，打印05，数据是12，打印12，数据是123，打印123

# FName

资源命名字符串，FName 通过一个轻型系统使用字符串。在此系统中，特定字符串即使会被重复使用，在数据表中也只存储一次。FNames 不区分大小写（大小写不是他比较的依据）。它们为不可变，无法被操作。FNames 的存储系统和静态特性决定了通过键进行 FNames 的查找和访问速度较快。FName 子系统的另一个功能是使用散列表为 FName 转换提供快速字符串。构建FName

###### 构建

FName name = FName（TEXT(“”)）;

或是

FName name（TEXT（“”））；

###### 对比

·==运算符重载比较，返回true或是false，不针对字符串本身进行比较，而是在对比索引中的数值，节约CPU开销

·函数FName::Compare比较，直接进行编码比较，相等返回0，不等返回非0

# FText

FText表示一个显示字符串，用户的显式文本都需要由FText进行处理。支持格式化文本（格式化数字，日期，时间），不提供修改函数，无法进行内容修改。FText类拥有本地化内置支持，可以处理本地化操作。

##### 构建

由于FText与本地化文本挂钩，所以FText的构建方式比较特殊，我们可以借助宏进行构建

1. 键位TEXT宏 LOCTEXT(“KEY”, “TEXT”);

键位宏需要注意，必须提供宏空间命名，使用宏LOCTEXT\_NAMESPACE 进行定义格式如下

在CPP最上端（Include之下）加入宏#define LOCTEXT\_NAMESPACE "TEXT\_ZQ"

在CPP末端放入宏#undef LOCTEXT\_NAMESPACE

在构建FText的地方使用 FText text = LOCTEXT("Key", "Text");

注意：如果此声明操作空间名称可以重复，在其他CPP中如需使用此空间名依照上面操作再次声明即可

1. 命名空间TEXT宏

NSLOCTEXT(“NAMESPACE”, “KEY”, “TEXT”)

声明时提供空间名和KEY

1. 空白text

FText::GetEmpty();

FText text = FText();