NSString(一)

/\*

\* C样式字符串是由char字符组成的，NSString对象由unichar字符组成。Unichar字符是以Unicode（至少2个字节）多字节字符为标准的。

unichar也是可以转成utf-8格式的

C字符串由char字符组成，NSString由unichar字符组成

\* %@可以使用任何对象上，因为它使用的是-（NSString\*）description方法（继承自NSObject）

\* UTF8STring : 把NSString对象转换为C字符串，可用%s打印

\* 可以用length方法来计算字符串中的字符数长度，返回类型为NSUInterger无符号整数值

\* NSString类涉及了恒不变的字符串，这意味着不能更改字符串内的任何字符（不可变字符串）

\* NSString\* -- “指向常量的指针”，相当于const char\*

\* 创建一个字符串：

NSString\* str = @"Hello World"; //字符串常量

NSLog(@"%@",str);

\* @"常量字符串"是一个NSConstantString对象；

常量字符串空间是在内存中以不同的方式与其它对象分配

常量字符串没有引用计数机制（oxffffffff）

\*/

#import <Foundation/Foundation.h>

int main(int argc, const char \* argv[])

{

@autoreleasepool {

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

NSString\* str = @"hElloWorld";

const char\* strUtf8 = [str UTF8String];

NSLog(@"UTF8:%s",strUtf8); //utf8

NSLog(@"%@",[str lowercaseString]); //小写

NSLog(@"%@",[str uppercaseString]); //大写

NSLog(@"%@",[str capitalizedString]); //首字母大小

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//stringWithString : 创建一个新的另一个内容不可变的字符串

// str1 str2 分别指向字符串常量

NSString\* str1 = @"HelloWorld";

NSString\* str2 = [NSString stringWithString:str1];

NSLog(@"\n%p:%@ %p:%@ length:%d",str1,str1,str2,str2,[str2 length]);

NSString\* str3 = str1;

NSLog(@"\n%p:%@ %p:%@ length:%d",str1,str1,str3,str3,[str3 length]);

// stringByAppendingString : 可以连接两个字符串,创建一个新的字符串对象

NSString\* str4 = [str1 stringByAppendingString:str2];

NSLog(@"\nstringByAppendingString : 可以连接两个字符串,创建一个新的字符串对象 %@",str4);

// stringWithFormat : 用指定的格式，创建一个字符串

NSString\* str5 = [NSString stringWithFormat:@"Now is:%@",[NSDate date]]; //返回系统时间

NSLog(@"\nstringWithFormat : 用指定的格式，创建一个字符串 %@",str5);

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Compare Strings 比较字符串

\* isEqualToString : 比较两个串是否相等，按照字符串的编码比较

\* caseInsensitiveCompare : 忽略大小写

compare比较的结果是

NSOrderedSame

NSOrderedAscending

NSOrderedDescending

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*/

if ([str1 isEqualToString:str5]) {

NSLog(@"Equal!");

} //判断是否相等

else

NSLog(@"Not Equal!");

NSString\* ss1 = @"helloworld";

NSString\* ss2 = @"HELLOWORLD";

switch ([ss1 compare:ss2]) {

case NSOrderedSame:

NSLog(@" = "); //判断两个字符串大小

break;

case NSOrderedAscending:

NSLog(@" < ");

break;

case NSOrderedDescending:

NSLog(@" > ");

break;

default:

break;

}

switch ([ss1 caseInsensitiveCompare:ss2]) {

case NSOrderedSame:

NSLog(@" = "); //判断两个字符串大小,忽略大小写

break;

case NSOrderedAscending:

NSLog(@" < ");

break;

case NSOrderedDescending:

NSLog(@" > ");

break;

default:

break;

}

//NSCaseInsensitiveSearch : 不区分大小写

//NSLiteralSearch : 进行完全比较，区分大小写

//NSNumericSearch : 比较字符串的字符个数，而不是字符值

NSString\* c1 = @"this is a String!";

NSString\* c2 = @"This is a String!";

BOOL result = ([c1 compare:c2 options:NSLiteralSearch | NSNumericSearch]==NSOrderedSame);

NSLog(@"result = %@",result?@"YES":@"NO");

}

return 0;

}

NSSting（二）

#import <Foundation/Foundation.h>

int main(int argc, const char \* argv[])

{

@autoreleasepool {

//substringToIndex : 返回一个子串，从首字符到index-1

NSString\* str = @"HelloWorld";

NSString\* sub1 = [str substringToIndex:5];

NSLog(@"sub1 = %@",sub1);

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* NSRange

通过指定一系列的子字符串，范围包含起始索引号加字符数。

NSString类的一些方法来创建范围规范使用NSRange的特殊的数据类型;

\* 在<Foundation/NSRange.h>中定义是一种有两个数字，位置和长度的结构：

typedef struct\_NSRange{

unsigned int location; //起始下标

unsigned int length; //范围长度

}

\* substringFromIndex : 返回子字符串的指定索引位置的字符开头的接收器，穿过字符串的末尾

\* substringWithRange : 返回指定范围中的一个字符的字符串

\* NSMakeRange : 从其参数创建一个范围

\* rangeOfString : 在一个字符串中找另一个字符串，如果找到了则返回指定字符串的range

如果没有找到，则将返回location member设置为NSNotFound

＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝

\*/

// NSString\* str = @"HelloWorld";

//substringFromIndex

NSString\* sub2 = [str substringFromIndex:5];

NSLog(@"sub2 = %@",sub2);

//substringWithRange : 按照所给的位置，长度，任意地从字符串中截取子串

//NSMakeRange

NSString\* sub3 = [str substringWithRange:NSMakeRange(5,5)];

NSLog(@"sub3 = %@",sub3);

NSRange range = [str rangeOfString:@"oW"];

if (range.location==NSNotFound) {

NSLog(@"Not found!"); //查找rangeOfString

}

else{

NSLog(@"Found! Index:%lu length:%lu",range.location,range.length);

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// 创建字符串常量，创建空字符串，给予赋值

NSString\* astring1 = @"hello world!";

NSLog(@"astring1 = %@",astring1);

NSString\* astring2 = [[NSString alloc] initWithString:@"hello world!"];

NSLog(@"asting2 = %@",astring2);

NSString\* astring3 = @"";

NSLog(@"astring1 = %@",astring3);

//initWithCString : 用标准创建字符串方法

char\* Cstring = "This is a String!";

NSString\* astring4 = [[NSString alloc] initWithCString:Cstring];

NSLog(@"astring4 = %@",astring4);

//创建一个临时字符串

const char\* cs = "This is a temporary string!";

NSString\* astring5 = [NSString stringWithCString:cs];

NSLog(@"astring5 = %@",astring5);

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

可变字符串

\* NSMutableString : 用于创建可更改其字符的字符串对象，此类是NSString类的子类 ，所以可以使用所有NSStringr方法

\* NSMutableString\* mstr = [NSMutableString stringWithString:otherStr];

当stringWithString : 方法发送到NSMutableString类，则返回一个可变的字符串对象

\* appendString : 在已有的字符串结尾追加

\* insertString:atIndex : 在指定位置，插入一个字符串

\* deleteCharacterInRange : 把制定范围内的字符子串删除

\* setString : 直接修改字符的内容

\* replaceCharactersInRange:withString : 把制定范围内的字符子串，替换为withString指定的字符串

\* repalceOccurencesOfString:withString:options:range : 搜索和替换一个字符串（在指定范围内查找替换）

mstr replaceOccurencesOfString:searchStr

withString:replaceStr

options:nil //不区分大小写

range:NSMakeRange(0,[mstr length]); //options:参数类型NSStringCompareOptions

\*/

NSString\* mstr = @"HelloWorld!";

NSMutableString\* mstr1 = [NSMutableString stringWithString:mstr];

[mstr1 appendString:mstr];

NSLog(@"mstr1 = %@",mstr1); //可变mstr1,追加

[mstr1 insertString:@"," atIndex:11];

NSLog(@"mstr2 = %@",mstr1); //指定位置插入字符串

[mstr1 deleteCharactersInRange:NSMakeRange(11,1)];

NSLog(@"mstr3 = %@",mstr1); //指定位置删除指定个数的字符

[mstr1 setString:mstr];

NSLog(@"mstr4 = %@",mstr1); //修改字符串内容

[mstr1 replaceCharactersInRange:NSMakeRange(5, 2)

withString:@"\_change\_"];

NSLog(@"mstr5 = %@",mstr1); //修改指定位置的指定个数的字符

[mstr1 replaceOccurrencesOfString:@"O" //查找'O'

withString:@"HHHH" //用"HHHH"替换

options:NSCaseInsensitiveSearch //不区分大小写

range:NSMakeRange(0,[mstr1 length])]; //全部遍历

NSLog(@"mstr6 = %@",mstr1);

}

return 0;

}