4/28/2023

由 4 个 "1" 和 4 个 "0" 组成的 8 位二进制补码，能表示的最小整数是

最大和最小分别是01111000 10000111,那么10000111的原码是11111001为-121

设栈的初始状态为空，当字符序列 "a3\_" 作为栈的输入时，输出长度为 3 的且可以用作 C 语言标识符的字符串序列有（）个

首先，栈的顺序是先进后出

字符序列为a3\_ 1)a入栈，再出栈，然后3入栈，再出栈，—入栈，再出栈 序列是a3\_

2)a入栈，再出栈,然后3,—入栈，再出栈，序列是a\_3

3)a入栈，3入栈，再出栈，a出栈，

—入栈，再出栈

序列是3a\_

4)

a入栈，3入栈，再出栈,

—入栈,序列是3\_a

5)

a入栈，3入栈,\_入栈，序列是\_3a

其次，C语言的标识符不能以数字开头，去除3a\_和3\_a 答案为3

从Java语言的定义，ABCD都是错误的，4个都不是正确的字符常量。可以查阅《JLS8》中的描述：

A character literal is expressed as a character or an escape

sequence,enclosed in ASCII single

quotes.一个字符常量表示为一个字符或一个转义序列，被一对ASCII单引号关闭。

哪个是不正确的字符常量？

A

”\n”

B

”1”

C

”a”

D

”\101”

CharacterLiteral:

' SingleCharacter '

' EscapeSequence '

SingleCharacter:

InputCharacter but not ' or \

而转义序列包括：

EscapeSequence:

\ b (backspace BS, Unicode \u0008)

\ t (horizontal tab HT, Unicode \u0009)

\ n (linefeed LF, Unicode \u000a)

\ f (form feed FF, Unicode \u000c)

\ r (carriage return CR, Unicode \u000d)

\ " (double quote ", Unicode \u0022)

\ ' (single quote ', Unicode \u0027)

\ \ (backslash \, Unicode \u005c)

OctalEscape (octal value, Unicode \u0000 to \u00ff)

...

四个答案都采用的是双引号关闭，所以是字符串常量而不是字符常量。

忽略这个错误，那么这题就没有答案，因为无论是采用特殊字符的转移序列'\n'还是采用八进制转义'\101'，都是满足Java语言规范中规定的字符常量格式。

已知串 S= "babab ", 其 Next 数值序列为 01123

Next数值序列需要计算字符串前缀和后缀最长匹配相同的长度加一而得。

第1位一定是0

看S第一位“b” ，没有前缀和后缀 所以一般第2位一定是1（0+1）；

看S.substring（0,2）“ba” 前缀b后缀a，相等长度为0，第3位是1（0+1）；

看S.substring（0,3）“bab” 前缀b=后缀b，相等长度为1，第4位是2（1+1）；

看S.substring（0,4）“baba” 前缀ba=后缀ba，相等长度为2，第5位是3（2+1）；

则S的Next数值序列为 01123

void testArgs(char\* s) {

printf("%d\n", sizeof(s));

}

int main() {

char s[] = "I want to go on vacation";

printf("%d\n", sizeof(s));

testArgs(s);

return 0;

}

题目主要考察C语言当中字符串长度统计和参数传递的问题。

从main函数可以知道sizeof统计的是字符串s占用空间的大小（且包括结束符'\0'），因此第一次输出的大小为25，当字符串s是通过参数传递到函数testArgs时，则函数会由指针s来指向传递的字符串这一片内存，因此第二次sizeof的时候，是对指向该字符串的指针变量长度的统计。

由于不同类型的指针变量长度都是一样的（不同机器可能存在差异），都是4字节长度

Java中，用new创建的对象在堆区；函数中的临时变量在栈区。但是，由于字符串使用过于频繁，在内存方法区中有一个字符串常量池，用来快速创建和访问字符串

public static void main(String[] args) {

String s1="6";

Double t1=6.3;

System.out.println(s1+t1);

}

A

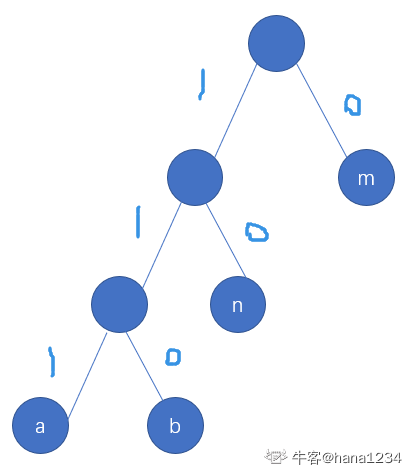
"66.3"

字符串与数字的相加；在运算时，会自动将Number型变量转化为字符串，并添加到结果之后

对字符串 "mabnmnm" 的二进制进行哈夫曼编码有多少位

字符串 "mabnmnm"每个字符的频率分别为：m 3/7，a 1/7，b 1/7，n 2/7。

建哈夫曼树，如图。



a：111，b：110，n：10，m：0。

字符串 "mabnmnm" 的哈夫曼编码为 0 111 110 10 0 10 0。

一共13位

字符串 "ababaaababaa" 的 next 数组为

Graphical user interface, text, application

Description automatically generated