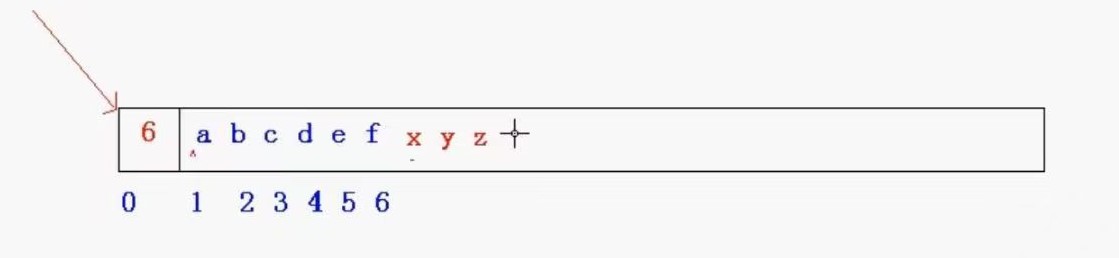
**字符串定长顺序存储表示**

****

此时 x y z为无效数据，因为0下标位置显示整个串有6个字符。采用该种表示方式则不用考虑“\0”的问题。

但是通常我们仍然采用\0 这种方式来进行

//C语言中保存的字符串是保存的字符串的首地址 //char \*p="hello" (C语言版) //C++里会有string类

#include <stdio.h>

#include <string.h>

#define MAXSTRLEN 3

#define u\_char unsigned char

typedef u\_char SString[MAXSTRLEN + 1]; //定义一个数组类型

void InitString(SString &S) {

S[0] = '\0'; //进行空串的初始化

}

void PrintString(SString S) {

printf("%s\n", S);

}

void StrAssign(SString &S,const char\* str) { //用一个字符串对定长字符串进行赋值

int len = strlen(str);

for (int i = 0; i < len; i++) {

S[i] = str[i];

}

S[len] = '\0';

}

int StrLength(SString S) { //求长度

int len = 0;

while (\*S != '\0') {

len++;

S++;

}

return len;

}

void StrCopy(SString &T, SString &S) { //拷贝函数

int len = StrLength(S);

for (int i = 0; i < len; i++) {

T[i] = S[i];

}

T[len] = '\0';

}

bool StrEmpty(SString S) { //判空函数

return S[0] == '\0';

}

int StrCompare(SString S, SString T) { //比较函数

int result=0; //作为局部变量，一定要初始化

while (\*S != '\0' || \*T != '\0') { //\*S,\*T 字符数组类型的指针 表示指向的字符

result = \*S - \*T; //实际上是ASCLL码相减

if (result != 0) {

break;

}

S++;

T++;

}

if (result > 0) {

result = 1;

}

else if (result < 0) {

result = -1;

}

return result;

}

void StrConcat(SString& T, SString S1, SString S2) { //链接函数

int len1 = StrLength(S1);

int len2 = StrLength(S2);

if (len1 + len2 < MAXSTRLEN) { //两者的和小于最大容量，可以相加

for (int i = 0; i < len1; i++) {

T[i] = S1[i];

}

for (int j = 0; j < len2; j++) {

T[len1 + j] = S2[j];

}

T[len1 + len2] = '\0';

}

else if (len1 < MAXSTRLEN) { //两者的和超了，但是S1不超

for (int i = 0; i < len1; i++) {

T[i] = S1[i];

}

for (int j = 0; j < MAXSTRLEN - len1; j++) {

T[len1 + j] = S2[j];

}

T[MAXSTRLEN] = '\0';

}

else { //S1就直接超了

for (int i = 0; i < MAXSTRLEN; i++) {

T[i] = S1[i];

}

T[MAXSTRLEN] = '\0';

}

}

//学C++string类之后 可以直接用string类的函数 这里相当于是对本质原理进行实现

void SubString(SString &sub,SString S ,int pos,int len) { //求子串函数

int s\_len = StrLength(S);

if (pos<0 || pos>= s\_len || len<0 || len>s\_len) {

return;

}

int j = pos;

for (int i = 0; i < len; i++) {

sub[i] = S[j + i];

}

sub[len] = '\0';

}

void StrInsert(SString &S, int pos, SString T) { //插入函数

int s\_len = StrLength(S);

int t\_len = StrLength(T);

if (s\_len + t\_len <= MAXSTRLEN) {

//先腾出空间，移动数据

t\_len = StrLength(T);

/\*for (int i = s\_len - 1; i >= pos; i--) {

S[i + t\_len] = S[i];

}

int j = pos;

for (int i = 0; i < t\_len; ++i) {

S[j + i] = T[i];

}

S[s\_len + t\_len] = '\0';\*/

}

else if (s\_len < MAXSTRLEN) {

t\_len = MAXSTRLEN - s\_len; //重新修改t\_len

/\*for (int i = s\_len - 1; i >= pos; i--) {

S[i + t\_len] = S[i];

}

int j = pos;

for (int i = 0; i < t\_len; ++i) {

S[j + i] = T[i];

}

S[s\_len + t\_len] = '\0';\*/

}

for (int i = s\_len - 1; i >= pos; i--) {

S[i + t\_len] = S[i];

}

int j = pos;

for (int i = 0; i < t\_len; ++i) {

S[j + i] = T[i];

}

S[s\_len + t\_len] = '\0';

}

void StrDelete(SString S,int pos,int len) { //删除函数

int s\_len = StrLength(S);

for (int i = pos; i < s\_len; i++) {

S[i] = S[i + len];

}

S[s\_len - len] = '\0';

}

void StrClear(SString S) { //清除函数

S[0] = '\0';

}

void main() {

SString S;

SString T;

InitString(S);

InitString(T);

//const char\* str = "abcdef";

//StrAssign(S, str);

//PrintString(S);

//StrCopy(T, S);

//PrintString(T);

StrAssign(S, "abc");

StrAssign(T, "xyz");

int res = StrCompare(S, T);

printf("%d \n", res);

SString Y;

InitString(Y);

StrConcat(Y, S, T);

PrintString(Y);

}