EUROPEAN UNIVERSITY OF LEFKE

Faculty of Engineering

Department of Computer Engineering



COMP218

OBJECT-ORIENTED PROGRAMMING

**PROGRAMMING ASSIGNMENT**

Prepared by **David O. Ladipo** (174574)

Submitted to Dr. Ferhun Yorgancıoğlu

**DString.h**

#ifndef DSTRING\_H

#define DSTRING\_H

#include <string.h>

using namespace std;

class DString

{

char \*value;

int len;

public:

DString();

DString(const char \*s);

DString(const DString &s);

~DString();

DString operator=( const DString &rhs );

friend DString operator+(DString &x, DString &y);

DString& operator+= (const DString &rhs);

friend int operator==(DString &x,DString &y);

friend int operator!=(DString &x,DString &y);

friend int operator<(DString &x,DString &y);

friend int operator<=(DString &x,DString &y);

friend int operator >(DString &x,DString &y);

friend int operator >=(DString &x,DString &y);

friend istream &operator >> (istream & ccin, DString & obj);

friend ostream &operator << (ostream & ccout, DString &obj);

int my\_strlen() const;

int my\_strcmp(const DString & rhs) const;

int my\_strncmp(const DString & rhs, size\_t num) const;

char my\_Strcpy (const DString & rhs)const;

char my\_Strncpy (const DString & rhs, size\_t num)const;

char my\_strcat(const DString & rhs) const;

char my\_strncat(const DString & rhs, size\_t num) const;

const char my\_strchr(int ch);

const char my\_strchrlast(int ch);

const char my\_StrStr(const DString & rhs);

const char my\_Strtok(const DString & rhs);

};

#endif // DSTRING\_H

**DString.cpp**

#include <iostream>

#include "DString.h"

#include <string.h>

using namespace std;

// DEFAULT CONSTRUCTOR

DString::DString()

{

len = 0;

value = new char[0];

value[0]= '\0';

}

// PARAMETERIZED CONSTRUCTOR

DString::DString(const char \*s){

len = strlen(s);

value = new char[len+1];

strcpy(value,s);

}

// COPY CONSTRUCTOR

DString::DString(const DString &s){

len = s.len;

value = new char[len+1];

strcpy(value,s.value);

}

// DESTRUCTOR

DString::~DString()

{

delete[] this -> value;

}

//................OPERATOR OVERLOADING................

// ASSIGNMENT OPERATOR

DString DString::operator=(const DString &rhs){

if (this == &rhs){

return \*this;

}

else{

delete[] value;

len = rhs.len;

value = new char[len+1];

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

value[i] = rhs.value[i];

value[len]='\0';

return \*this;

}

}

//ADDITION OPERATOR

DString operator+(DString &x, DString &y){

DString z;

z.len = x.len + y.len;

z.value= new char[z.len+1];

strcpy(z.value, x.value);

strcat(z.value, y.value);

return z;

}

//SHORT HAND ASSIGNMENT OPERATOR

DString& DString::operator+=(const DString &rhs){

if (this == &rhs){

return \*this;

}

delete[] value;

//value = new char[strlen(rhs.value)+1];

strcat(value, rhs.value);

return \*this;

}

// EQUAL TO OPERATOR

int operator==(DString &x,DString &y)

{

int rel =0;

if (strcmp(x.value, y.value)==0){

rel =1;

}

return rel;

}

//NOT EQUAL TO OPERATOR

int operator!=(DString &x,DString &y){

int rel =0;

if (strcmp(x.value, y.value)==0){

rel =1;

}

return rel;

}

// LESS THAN OPERATOR

int operator<(DString &x,DString &y){

int rel =0;

int result = 0;

rel = (strcmp(x.value, y.value));

if (rel < 0)

{

result =1;

}

return result;

}

// LESS THAN OR EQUAL TO OPERATOR

int operator<=(DString &x,DString &y){

int rel =0;

int result = 0;

rel = (strcmp(x.value, y.value));

if (rel < 0 || rel == 0)

{

result =1;

}

return result;

}

// GREATER THAN OPERATOR

int operator > (DString &x,DString &y){

int rel =0;

int result = 0;

rel = (strcmp(x.value, y.value));

if (rel > 0)

{

result =1;

}

return result;

}

// GREATER THAN OR EQUAL TO

int operator >= (DString &x,DString &y){

int rel =0;

int result = 0;

rel = (strcmp(x.value, y.value));

if (rel > 0 || rel ==0)

{

result =1;

}

return result;

}

//INSERTION OPERATOR

istream & operator >> (istream & ccin, DString & obj){

char val[20];

cout << "enter a string " << endl;

ccin >> val;

obj.len = strlen(val);

obj.value = new char [obj.len+1];

strcpy(obj.value, val);

return ccin;

}

//EXTRACTION OPERATOR

ostream & operator << (ostream & ccout, DString &obj){

cout << obj.value;

return ccout;

}

//\*\*\*\*\*\*\*\*\*CUSTOMIZED MEMBER FUNCTIONS\*\*\*\*\*\*\*\*\*\*\*

// STRLEN FUNCTION

int DString::my\_strlen() const

{

return len;

}

// STRCMP FUNCTION

int DString::my\_strcmp(const DString & rhs)const {

if (len < rhs.len)

    return 1;

  else if (len > rhs.len)

    return -1;

  return strcmp(value, rhs.value);

}

//STRNCMP FUNCTION

int DString::my\_strncmp(const DString & rhs, size\_t num)const{

char \*s1 = value;

char \*s2 = rhs.value;

unsigned char u1, u2;

while (num-- > 0)

{

u1 = (unsigned char) \*s1++;

u2 = (unsigned char) \*s2++;

if (u1 != u2)

return u1 - u2;

if (u1 == '\0')

return 0;

}

}

//STRCPY FUNCTION

char DString::my\_Strcpy(const DString & rhs)const {

char \* ptr = value;

char \* ptr2 = rhs.value;

while (\*ptr2 != '\0'){

\*ptr = \*ptr2;

ptr ++;

ptr2++;

}

\*ptr = '\0';

return \*ptr;

}

// STRNCPY FUNCTION

char DString::my\_Strncpy(const DString & rhs, size\_t num)const {

char \* ptr = value;

char \* ptr2 = rhs.value;

while (\*ptr2 && num--){

\*ptr = \*ptr2;

ptr ++;

ptr2++;

}

\*ptr = '\0';

return \*ptr;

}

// STRCAT FUNCTION

char DString::my\_strcat(const DString & rhs) const{

char \* ptr = value;

char \* ptr2 = rhs.value;

char\* strret = ptr;

if((NULL != ptr) && (NULL != ptr2)){

// Iterate till end of dest string

while(NULL != \*ptr)

{

ptr++;

}

//Copy src string starting from the end NULL of dest

while(NULL != \*ptr2)

{

\*ptr++ = \*ptr2++;

}

// put NULL termination

\*ptr = NULL;

}

return \*strret;

}

// STRCAT FUNCTION

char DString::my\_strncat(const DString & rhs, size\_t num)const{

char \* ptr = value;

char \* ptr2 = rhs.value;

char\* strret = ptr;

if((NULL != ptr) && (NULL != ptr2) ){

/\* Iterate till end of dest string \*/

while(NULL != \*ptr)

{

ptr++;

}

/\* Copy src string starting from the end NULL of dest \*/

while(NULL != \*ptr2 && num--)

{

\*ptr++ = \*ptr2++;

}

/\* put NULL termination \*/

\*ptr = NULL;

}

return \*strret;

}

//STRCHR FUNCTION

const char DString::my\_strchr(int ch){

int index = 0;

char \*p = value;

if (NULL == p)

return NULL;

for (int i=0; \*p!= '\0'; i++){

if(\*p == (char)ch){

index = i;

cout << "Found Character " << "\""<<(char)ch <<"\"""at index: " << index <<endl;

}

\*p++;

}

return NULL;

}

// STRRCHR FUNCTION

const char DString::my\_strchrlast(int ch){

int index = 0;

char \*p = value;

if (NULL == p)

return NULL;

for (int i=0; \*p!= '\0'; i++){

if(\*p == (char)ch){

index = i;

}

\*p++;

}

cout << "Found Last Character " << "\""<<(char)ch <<"\"""at index: " << index <<endl;

return NULL;

}

// STRSTR FUNCTION

const char DString::my\_StrStr(const DString & rhs){

if( const char \*p = strstr(value, rhs.value) )

std::cout << "Found: " << p << std::endl;

else

std::cout << "Not found!" << std::endl;

}

//STRTOK FUNCTION

const char DString::my\_Strtok(const DString & rhs){

char \* token;

token = strtok(value, rhs.value);

while(token != NULL){

cout << token << endl;

token = strtok(NULL, rhs.value);

}

}

**main.cpp**

#include <iostream>

#include "DString.h"

using namespace std;

int main()

{

cout << "\*\*\*\*\*\* OPERATORS TEST \*\*\*\*\*\*" <<endl <<endl;

cout<<"\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"<<endl;

cout << "TESTING ASSIGNMENT = OPERATOR........" <<endl;

cout<<"\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"<<endl;

DString str1;

DString str2("Wifey");

cout << "String str1: " << str1 <<endl;

cout << "String str2: " << str2 <<endl;

str1 = str2;

cout << "Assignment Operation Successful.. str1 gets stored in str2.. " << endl;

cout << "Str1: " <<str1<<endl;

cout << "Str2: " <<str2 <<endl<<endl;

cout<<"\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"<<endl;

cout << "TESTING ADDITION + OPERATOR........" <<endl;

cout<<"\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"<<endl;

DString str3 = "Good ";

DString str4("Morning");

DString str5;

str5 = str3 + str4;

cout << "Addition Operation Successful.. str3 + str4 is assigned to str5.." <<endl;

cout << "str3: " <<str3 <<endl;

cout<<"str4: "<<str4 <<endl;

cout <<"str5: "<<str5 <<endl<<endl;

cout<<"\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"<<endl;

cout << "TESTING SHORT HAND ASSIGNMENT += OPERATOR........" <<endl;

cout<<"\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"<<endl;

cout << "str4: "<<str4 <<endl;

cout << "str3: " <<str3<<endl;

cout << "After Short hand Assignment Operation...(str4+=str3)" <<endl;

str4+=str3;

cout << "str4: "<<str4 <<endl<<endl;

cout<<"\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"<<endl;

cout << "TESTING IS EQUAL TO == OPERATOR........" <<endl;

cout<<"\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"<<endl;

DString str6= "Good Evening";

DString str7("Good Evening");

cout <<"str6: "<< str6 <<endl;

cout <<"str7: " <<str7<<endl;

if (str6 == str7)

cout << "Both Strings are Equal" <<endl<<endl;

else

cout << "They are not Equal" <<endl <<endl;

cout<<"\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"<<endl;

cout << "TESTING IS NOT EQUAL TO != OPERATOR........" <<endl;

cout<<"\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"<<endl;

DString str8 = "up";

DString str9("Whatsup");

cout << "str8: " <<str8<<endl;

cout << "str9: " << str9<<endl;

if (str8 != str9)

cout << "They are Equal" <<endl<<endl;

else

cout << "Not Equal" <<endl <<endl;

cout<<"\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"<<endl;

cout << "TESTING IS GREATER THAN > OPERATOR........" <<endl;

cout<<"\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"<<endl;

DString str10("Everything is Everywhere");

DString str11("Everything");

cout <<"str10: " <<str10<<endl;

cout <<"str11: "<<str11<<endl;

if (str10 > str11)

cout<<"String:" <<str10<< "....is greater than.... " <<"String:" << str11 <<endl<<endl;

else

cout<<"String:" <<str10<< "....is greater NOT than.... " <<"String:" << str11 <<endl<<endl;

cout<<"\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"<<endl;

cout << "TESTING IS GREATER THAN OR EQUAL TO >= OPERATOR........" <<endl;

cout<<"\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"<<endl;

DString str12("Everything");

DString str13("Every");

cout <<"str12: " <<str12<<endl;

cout <<"str13: "<<str13<<endl;

if(str12 >= str13)

cout<<"String:" <<str12<< "....is greater than or equal to.... " <<"String:" << str13 <<endl<<endl;

else

cout<<"String:" <<str12<< "....is greater NOT than or equal to.... " <<"String:" << str13 <<endl<<endl;

cout<<"\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"<<endl;

cout << "TESTING LESS THAN < OPERATOR........" <<endl;

cout<<"\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"<<endl;

DString str14("Every");

DString str15("Everything");

cout <<"str14: " <<str14<<endl;

cout <<"str15: "<<str15<<endl;

if (str14 < str15)

cout<<"String:" <<str14<< "....is less than.... " <<"String:" << str15 <<endl<<endl;

else

cout<<"String:" <<str14<< "....is not less than.... " <<"String:" << str15 <<endl<<endl;

cout<<"\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"<<endl;

cout << "TESTING LESS THAN OR EQUAL TO <= OPERATOR........" <<endl;

cout<<"\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"<<endl;

DString str16("Every");

DString str17("Every");

cout <<"str16: " <<str16<<endl;

cout <<"str17: "<<str17<<endl;

if (str16 <= str17)

cout<<"String:" <<str16<< "....is less than or equal to.... " <<"String:" << str17 <<endl<<endl;

else

cout<<"String:" <<str16<< "....is not less than or equal to.... " <<"String:" << str17 <<endl<<endl<<endl;

cout << "\*\*\*\*\*\* CUSTOMIZED STRING FUNCTIONS TEST \*\*\*\*\*\*" <<endl <<endl;

cout<<"\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"<<endl;

cout << "TESTING CUSTOMIZED Strlen function........" <<endl;

cout<<"\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"<<endl;

DString s1("Everything will be fine");

cout << "s1: " << s1 <<endl;

cout <<"Length of String s1 is: " << s1.my\_strlen() <<endl<<endl;

cout<<"\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"<<endl;

cout << "TESTING CUSTOMIZED Strcmp function........" <<endl;

cout<<"\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"<<endl;

DString s2("World");

DString s3 = "World";

cout <<"String s2: "<<s2 <<endl;

cout <<"String s3: "<<s3 <<endl;

int ret = s2.my\_strcmp(s3);

if(ret == 1)

cout <<"String s2: "<< s2 << " is less than String s3: "<<s3 <<endl<<endl;

else if (ret == -1)

cout <<"String s2: "<< s2 << " is more than String s3:"<<s3 <<endl<<endl;

else

cout <<"String s3:" <<s2 << " is EQUAL to String s3:" <<s3 <<endl <<endl;

cout<<"\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"<<endl;

cout << "TESTING CUSTOMIZED Strncmp function........" <<endl;

cout<<"\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"<<endl;

DString s4("Hello");

DString s5= "World";

cout <<"String s4: "<<s4 <<endl;

cout <<"String s5: "<<s5 <<endl;

int rel = s4.my\_strncmp(s5, 2);

if(rel == 0)

cout <<"EQUAL "<<endl<<endl;

else

cout << "NOT EQUAL" <<endl <<endl;

cout<<"\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"<<endl;

cout << "TESTING CUSTOMIZED Strcpy function........" <<endl;

cout<<"\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"<<endl;

DString s6("Hello");

DString s7;

cout <<"String s6: "<<s6 <<endl;

cout <<"String s7: "<<s7 <<endl;

s7.my\_Strcpy(s6);

cout << "After copying s6 to s7" <<endl;

cout <<"String s6: "<<s6 <<endl;

cout <<"String s7: "<<s7 <<endl;

cout<<"\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"<<endl;

cout << "TESTING CUSTOMIZED Strncpy function........" <<endl;

cout<<"\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"<<endl;

DString s8("Hello");

DString s9;

cout <<"String s8: "<<s8 <<endl;

cout <<"String s9: "<<s9 <<endl;

s9.my\_Strncpy(s8, 2);

cout << "After copying 2 characters from s8 to s9" <<endl;

cout <<"String s8: "<<s8 <<endl;

cout <<"String s9: "<<s9 <<endl<<endl;

cout<<"\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"<<endl;

cout << "TESTING CUSTOMIZED Strcat function........" <<endl;

cout<<"\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"<<endl;

DString s10("Hello ");

DString s11 = "David";

cout <<"String s10: "<<s10 <<endl;

cout <<"String s11: "<<s11 <<endl;

s10.my\_strcat(s11);

cout <<"String s10 is now: "<<s10 <<endl<<endl;

cout<<"\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"<<endl;

cout << "TESTING CUSTOMIZED Strncat function........" <<endl;

cout<<"\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"<<endl;

DString s12("Hello ");

DString s13("Kate");

cout <<"String s12: "<<s12 <<endl;

cout <<"String s13: "<<s13 <<endl;

s12.my\_strncat(s13, 2);

cout <<"String s13 is now: "<<s12 <<endl<<endl;

cout<<"\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"<<endl;

cout << "TESTING CUSTOMIZED Strchr function........" <<endl;

cout<<"\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"<<endl;

DString s14("object oriented Programming");

cout <<"String s14: "<<s14 <<endl;

s14.my\_strchr('o');

cout <<endl;

cout<<"\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"<<endl;

cout << "TESTING CUSTOMIZED Strrchr function........" <<endl;

cout<<"\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"<<endl;

DString s15("i love Programming");

cout <<"String s15: "<<s15 <<endl<<endl;

s15.my\_strchrlast('o');

cout<<"\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"<<endl;

cout << "TESTING CUSTOMIZED Strstr function........" <<endl;

cout<<"\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"<<endl;

DString s16("i love to Program most times");

DString s17("most");

cout <<"String s16: "<<s16 <<endl;

cout <<"String s17: "<<s17 <<endl;

s16.my\_StrStr(s17);

cout <<endl;

cout<<"\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"<<endl;

cout << "TESTING CUSTOMIZED Strtok function........" <<endl;

cout<<"\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"<<endl;

DString s18("i-love -to -Program -most -times");

cout <<"String s18: "<<s18 <<endl;

cout << "Result after strtok Operation: " <<endl;

s18.my\_Strtok("-");

return 0;

}

**Output**







