BIL 105E – Introduction to Scientific and Engineering Computing (C)

Spring 2015-2016

Homework 3

CRN:21834

Yunus Güngör

No:150150701

Date:01.05.2016

Introduction

This project's main aim is to understand pointers by string processes like, getting a substring, removing a part from a string, inserting another string into desired place or finding and replacing a part of the string.

Development Environment

This program has only 1 source code file written in C:

150150701.c

This program tested and compiled in following system:

gcc 4.8.5 20150623 on Red Hat 4.8.5-4 (ITU SSH Server)

gcc compiler has been used to compile the program by the command:

```
gcc 150150701.c -o hw3
```

Important Variables and Functions

user_menu: Prints menu and gets input about what to do next

set_ccs: Gets the string ccs

sub_string: Returns a substring of ccs between desired index numbers

remove_string: Returns a substring of ccs between desired index numbers and deletes that part from ccs

insert_string: inserts a string to desired place in ccs

replace_string: finds and replaces a substring from ccs with a desired strging. Returns how many strings found

findLeght: Returns the length of the given string

keepGoing: makes program repeat itself until 0 is choosen

length, lengthC, lengthF, lengthI, lengthR: lengths of certain strings

begin_index,end_index; indexes to perform operations on strings

ccs: main string to perform operations on

p,p1,p2,p3: some temporary pointer values to perform necessary tasks. This pointers used for another aim when their job is done.

lastCharacter: pointer of the strings last char. It sets to zero if necessary.

Program Flow

Pseudo code of the program:

```
int function main
{
      keepGoing=1
      initialize ccs as ‘\0’;
      while user did not enter 0 as input
      {
            user_menu()
            case 1:
                  set_ccs();
            case 2:
                  if ccs is entered before
                  {
                        Print "Enter the BEGIN INDEX and END INDEX
numbers:\n";
                        Get begin_index,end_index;
                        p=sub_string (ccs,begin_index,end_index);
                              if p is not null
                                    print “Substring” begin_index”,”
end_index”):”p;
                              else
                                    print ”An error happened!\n";
                  }
                  else
                  {
                        Print "Error:CCS must be entered to perform
this action!\n";
                  }
            case 3:
                  if ccs is entered before
                  {
                        Print “Enter the BEGIN INDEX and END INDEX
numbers:\n";
```

```
                    Get begin_index end_index;
                    p=remove_string(&ccs,begin_index,end_index);
                    if p is not null
                         print "Removed
String;”begin_index”,”end_index”:”p;
                    else
                         print "An error happened!\n";
               }
               else
               {
                    Print "Error:CCS must be entered to perform
this action!\n";
               }
          case 4:
               if ccs is entered before
               {
                    Print “Enter a new string for insertion:\n";
                    Get insert;
                    Adjust insert in memory for its new length;
                    print “Enter the BEGIN INDEX number where the
new string begins:\n";
                    get begin_index;
                    if insert_string(&ccs,insert,begin_index)
returns smaller than zero
                         print"An error happened!\n";
               }
               else
               {
                    Print "Error:CCS must be entered to perform
this action!\n";
               }
          case 5:
```

```
                if ccs is entered before
                {
                      Print “Find what:\n";
                      Get find;
                      Adjust find in memory for its new length;
                      print "Replace with what:\n";
                      get replace;
                      Adjust replace in memory for its new length;

      replacementCount=replace_string(&ccs,find,replace);
                      if replacementCount smaller than 0
                            print "An error happened!\n";
                      else
                            print replacementCount
                }
                else
                {
                      Print "Error:CCS must be entered to perform
this action!\n";
                }
           default:
                print "Seems like there is a non valid input, try
again maybe? \n";
           }
           Print CCS
}
      return 0;
}

int user_menu ()
{
```

```
      print "0: Exit the program\n1: Set Current Character
Sequence\n2: Get Substring\n3: Remove Substring\n4: Insert\n5:
Replace\nYour choise:\n";

      get choosen;

      return choosen;

}

set_ccs(ccs)

{

      Free ccs in memory;

      Print "Enter CCS:";

      Get ccs;

      Adjust ccs in memory for its new length;

      return findLeght(ccs);

}


sub_string (ccs, begin_index, end_index)

{

      lenght=findLeght(ccs);

      if begin_index smaller than 0 or end_index is bigger than
lenght or end_index smaller than 0 or end_index is smaller than
begin_index

            return NULL;

      else

      {

            copy from ccs between begin_index and end_index to p;

      }

      return p;

}

remove_string(ccs, begin_index, end_index)

{

      lenght=findLeght(*ccs);
```

```
     if begin_index smaller than 0 or end_index is bigger than
lenght or end_index smaller than 0 or end_index is smaller than
begin_index
          return NULL;
     else
     {
          size=end_index-begin_index+1;
          set p1 to starting point to remove;
          set p2 to ending point to remove;
          get size+1 byte from memory for substr;
          starting from p1 copy bytes as many as size to substr;
          add null to end of the string;
          starting from p2 copy lenght-end_index+1 bytes to p1;
          adjust p1 in memory to its new length;
     }
     return substr;
}

insert_string(ccs, insert, begin_index)
{
     lenghtC=findLeght(ccs);
     lenghtI=findLeght(insert);
     adjust ccs in memory for lenghtI+lenghtC;
     set p1 to start of the insertion;
     copy lenghtC-begin_index bytes from p1 to p1+lengthI;
     copy lenghtI bytes from insert to p1;
     return lenghtI+lenghtC;
}

replace_string(ccs, find, replace)
{
```

```
Initialize found=0;
Print "find:” find “replace:” replace;
lenghtC=findLeght(ccs);
lenghtR=findLeght(replace);
lenghtF=findLeght(find);
p1=ccs;
print "CCS before change:"p1;
while(p1 is not the end of string)
{
      p2=find;
      p3=p1;
      while(p1 equals p2 and p2 is not equal null)
      {
            move p1 one char forward;
            move p2 one char forward;
      }
      If p2 is not equal null
      {
            Adjust ccs in memory for lenghtC-lenghtF+lenghtR
bytes
            Adjust p1 to new ccs
            Adjust p3 to new ccs
      }
      slide the necessarry part;
      overwrite and change necessary part;
      add one to found;
      lenghtC=findLeght(ccs);
      }
      move p1 one char forward;
}
```

```
      return found;
}

findLeght(str)
{
      while str is not the end pf string
      {
            lenght++;
            move str one char forward;
      }
      return lenght;
}
```

Conclusion

This project helped me to understand pointers and memory allocation. Besides that, I also learned some exceptional cases about memory allocation and how to handle with those cases.