Práctica N

Considere los siguientes archivos fuente

```
/**main.h*/
#ifndef MAIN_H
#include <iostream>
#include <vector>
#include <stdbool.h>
using namespace std;
#define NELEM(x) ((sizeof(x))/(sizeof((x)[0])))
typedef unsigned int uint;
extern unsigned int bucket[256];
/** suma de elementos
*/
unsigned int selem(unsigned int [NELEM(bucket)],int);
void fill_string_vec(string,vector<string>&);
void print_vector_string(vector<string>);
bool is_included(std::string,vector<string>);
bool is_space(char c);
#endif // MAIN_H
```

```
/** funciones.cpp */
#include <iostream>
#include <vector>
#include "main.h"
using namespace std;

/** Suma los elementos del arreglo b y
 *  devuelve el resultado.
 */
uint selem(unsigned int b[256],int s)
{
  int i,suma=0;//,x,z;
  for(i=0; i<=(int)NELEM(bucket); i++)
  {
     suma=suma+b[i];
  }
  return suma;
```

```
}/*end selem()*/

/**Stub (inicialmente 2020.03.10)
 * Debe colocar en el vector string_vec las subsecuencias de
 * characters del string linea que no contienen characters ' '.
  pre:string_vec.size()=0
*/
void fill_string_vec(string linea,
                     vector<string>& string_vec)
{
  uint i,inic=-1,fin;
  std::string str_tmp;
  /**detect if string is fill of ' 's*/
  for(i=0;i<linea.size();i++)
  {
    if(linea[i]==' ')
        continue;
    inic=i;
    break;
  }
  if(inic<0)
    goto out;
  while(i<=linea.size()-1)
  {
    for(i=inic;i<linea.size();i++)
    {
      if(linea[i]==' ')
        continue;
      inic=i;
      break;
    }
    for(i=inic;i<linea.size();i++)
    {
      if((' '==linea[i])||('\0'==linea[i])||(i==(linea.size()-1))){
        fin=i;
        str_tmp=linea.substr(inic,fin-inic+1);

        //print_index_char_of_string(str_tmp);
```

```
        if(str_tmp.size()>=2){
          /**Quitar coma --si la hay--*/
          if(str_tmp[str_tmp.size()-2]==',')
            str_tmp=str_tmp.substr(0,str_tmp.size()-2);
          /**Quitar punto y coma --si lo hay--*/
          if(str_tmp[str_tmp.size()-2]==';')
            str_tmp=str_tmp.substr(0,str_tmp.size()-2);
          /**Quitar punto --si lo hay--*/
          if(str_tmp[str_tmp.size()-2]=='.')
            str_tmp=str_tmp.substr(0,str_tmp.size()-2);
        }
        if(!is_included(str_tmp,string_vec))
          string_vec.push_back(str_tmp);
        inic=fin+1;
        break;
      }
    }/*end for()*/
  }/*end while()*/

out:
    return;
}/*end fill_string_vec()*/


void print_vector_string(vector<string> v)
{
    unsigned int i;
    for(i=0;i<v.size();i++)
    {
        cout<<v[i]<<"\n";
    }
}/*end print_vector_string()*/

bool is_included(std::string word,vector<string> word_list)
{
  for(uint i=0;i<word_list.size();i++)
  {
    if(word==word_list[i])
```

```
      return true;
  }
  return false;
}

bool is_space(char c)
{
  return (' '==c);
}
```

y el archivo principal (archivo que contiene la función main)

```
/**main0.cpp*/
#include <iostream>
#include <stdio.h>

#define NDEBUG
#include <assert.h>
#include "main.h"

using namespace std;
unsigned int bucket[256];

int main(int argc,char *argv[])
{
    uint i;
    char str[]="Pater noster, qui es in caelis, \
 santificetur nomen Tuum, adveniat Regnum Tuum, \
 fiat voluntas tua, sicut in caelo et in terra. \
 Panem nostrum cotidianum da nobis hodie, \
 et dimitte nobis debita nostra, \
 sicut et nos dimittimus debitoribus nostris; \
 et ne nos inducas in tentationem,\
 sed libera nos a malo";
    printf("%s\n",str);

    for(i=0;i<NELEM(str);i++)
```

```
    {
        bucket[(int)str[i]]++;
    }
    printf("suma=%d\n",selem(bucket,NELEM(bucket)));

    printf("%-4s %-6s\n","Char","Amount");
    for(i='A';i<='Z';i++)
    {
        if(bucket[i])
            printf("%c %5d\n",i,bucket[i]);
    }
    for(i='a';i<='z';i++)
    {
        if(bucket[i])
            printf("%c %5d\n",i,bucket[i]);
    }

    return 0;
}/*end main()*/
```

Abstracción de operaciones

Nótese que el código en la función `main` del archivo `main0.cpp` esencialmente utiliza la técnica buckets para contar cuántas letras mayúsculas y minúsculas se utilizan en la cadena

```
    char str[]="Pater noster, qui es in caelis, \
 santificetur nomen Tuum, adveniat Regnum Tuum, \
 fiat voluntas tua, sicut in caelo et in terra. \
 Panem nostrum cotidianum da nobis hodie, \
 et dimitte nobis debita nostra, \
 sicut et nos dimittimus debitoribus nostris; \
 et ne nos inducas in tentationem,\
 sed libera nos a malo";
```

Como ejercicio práctico se le pide al lector reescribir la función `main` de manera que en lugar de contar cuántas letras mayúsculas y minúsculas tiene la

cadena `str`, se haga la cuenta en la función `main` de cuántas veces aparece cada palabra en la misma cadena. Para responder al ejercicio práctico el lector deberá utilizar la estructura que se describe a continuación en la forma en que se le indicará en este documento explícitamente.

Primera modificación a la función main

Modifique la función main del archivo `main0.cpp` para que quede como se indica a continuación y guarde los cambios en el archivo `main1.cpp`

```
/**main1.cpp*/
#include <iostream>
#include <stdio.h>

#define NDEBUG
#include <assert.h>
#include "main.h"

using namespace std;
unsigned int bucket[256];

int main(int argc,char *argv[])
{
    uint i;
    char str[]="Pater noster, qui es in caelis, \
 santificetur nomen Tuum, adveniat Regnum Tuum, \
 fiat voluntas tua, sicut in caelo et in terra. \
 Panem nostrum cotidianum da nobis hodie, \
 et dimitte nobis debita nostra, \
 sicut et nos dimittimus debitoribus nostris; \
 et ne nos inducas in tentationem,\
 sed libera nos a malo";
    printf("%s\n",str);

    for(i=0;i<NELEM(str);i++)
    {
        bucket[(int)str[i]]++;
    }
```

```
    printf("suma=%d\n",selem(bucket,NELEM(bucket)));

    printf("%-4s %-6s\n","Char","Amount");
    for(i='A';i<='Z';i++)
    {
        if(bucket[i])
            printf("%c %5d\n",i,bucket[i]);
    }
    for(i='a';i<='z';i++)
    {
        if(bucket[i])
            printf("%c %5d\n",i,bucket[i]);
    }

    vector<string> vs;          /*vector string*/
    string stringsa=string(str);
    fill_string_vec(stringsa,vs);
    printf("/****************************************/\n");
    print_vector_string(vs);

    return 0;
}/*end main()*/
```

Una estructura string y uint

En esta subsección se describe una estructura que estará formada por un `string` y un `unsigned int`. Se pide al lector utilizar el tipo de dato `string` y el tipo de dato `uint` definido en el archivo `main.h` como

```
typedef unsigned int uint;
```

Agregue al archivo `main.h` la declaración de la estructura `StringYuint` cuyo diagrama UML se incluye a continuación.

StringYuint
palabra : string N : uint I : uint count : static uint
StringYuint(w : string) :

El lector deberá instanciar una estructura `StringYuint` por cada palabra distinta que esté contenida en la cadena `str`; el atributo de clase `count` deberá inicializarse a cero en alcance de archivo. En el constructor de la estructura se incrementará el valor del atributo de clase `count` y se guardará su valor en el atributo de objeto I. El atributo `palabra` contendrá una de las palabras de la cadena `str` y el atributo `N` contendrá la cantidad de veces que aparece esa palabra en la cadena `str`. Nótese que las distintas palabras contenidas en la cadena `str` ya han sido colocadas en el vector de strings `vs` del archivo `main1.cpp`.

Instanciación de las estructuras `StringYuint`

Una vez que el lector haya declarado la estructura `StringYuint` en el archivo `main.h` y agregado a ese mismo archivo de cabecera las líneas

```
extern uint *BUCKET;
uint mapa(string word,StringYuint *syu_pt,uint size);
void contar_palabras(string line,StringYuint *syu_pt,vector<string> vs);
```

se podrán instanciar las estructuras correspondientes a las distintas palabras como se pide a continuación en la función `main`: guarde los cambios en un archivo llamado `main2.cpp`.

```
/**main2.cpp*/
//...
uint *BUCKET;

int main(int argc,char *argv[])
{
//...
```

```
    vector<string> vs;         /*vector string*/
    string stringsa=string(str);
    fill_string_vec(stringsa,vs);
    printf("/*****************************************/\n");
    print_vector_string(vs);

    StringYuint *syu_Pt=new StringYuint[vs.size()];
    for(uint index=0;index<vs.size();index++)
    {
      syu_Pt[index]=StringYuint(vs[index]);
    }
    BUCKET=new uint[vs.size()+1];
    contar_palabras(stringsa,syu_Pt,vs);

    printf("/*****************************************/\n");
    for(uint index=0;index<vs.size();index++)
    {
      printf("%-15s %3d\n",
             &(syu_Pt[index].palabra[0]),
             BUCKET[syu_Pt[index].I]);
    }

    return 0;
}/*end main()*/

void contar_palabras(string line,
                     StringYuint *syu_pt,
                     vector<string> string_vec)
{

}
```