```
#include <stdlib.h>
#include <ctime>
#include <stdio.h>
#include <stdlib.h>
#include <sys/time.h>
#include <string>
#include <iostream>
using namespace std;

#define _NUMEROCASOS 1
#define _NUMEROVALORES 1

int progreso = 0;
int n = _NUMEROCASOS;
int incremento = _NUMEROCASOS /_NUMEROVALORES;

//Función para la barra de progreso
void barraProgreso(int i){

      int aux = (((i/incremento)*100)/_NUMEROVALORES)+1;

      if(aux > progreso){
            fprintf(stderr,"%d%%\r",aux);
            progreso = aux;
      }
      else progreso = aux;
      if(progreso == 100) fprintf(stderr,"\n");
}

//Función para sacar el tiempo.
long long mseconds(){
      struct timeval t;
      gettimeofday(&t,NULL);
      return t.tv_sec*100000+t.tv_usec;
}
```

```
bool esConsonante(char c){
      if(('a' == c)||('e' == c)||('i' == c)||('o' == c)||('u' == c)) return false;
      return true;
}

string iterativo(string cadena){
      //INICIALIZACION DE VARIABLES
      string cadenaActual = "";
      string cadenaMaxima = "";
      int buscando;//BUSCANDO REPRESENTA EL ESTADO. SI ESTÁ A 0 ESTARÁ BUSCANDO CONSONANTE Y SI ESTÁ A 1 VOCALES.
      //BUSCA LA PRIMIERA CONSONANTE
      int i = 0;
      while((i < cadena.length())&&(!esConsonante(cadena[i]))) i++;
      if(i < cadena.length()){
            //HA ENCONTRADO LA PRIMERA CONSONANTE Y EMPIEZA EL CICLO PARA BUSCAR LA CADENA  DESESADA.
            buscando = 1; //UNA VEZ QUE SE HA ENCONTRADO LA PRIMERA CONSONANTE SE EMPIEZA A BUSCAR UNA VOCAL.
            cadenaActual = cadena[i];
            for(int j = i+1; j < cadena.length();j++){
                  if(buscando == 0){ //SI ESTÁ BUSCANDO CONSONANTE.
                        if(esConsonante(cadena[j])){ //SI ES CONSONANTE.
                              cadenaActual = cadenaActual + cadena[j];
                              buscando = 1;
                        } else {
                              cadenaActual = "";
                        }
                  } else { //PARA QUE LEA UNA LETRA EN CADA ITERACION.
                        if(buscando == 1){//SI ESTÁ BUSCANDO UNA VOCAL.
                              if(!esConsonante(cadena[j])){//SI ES VOCAL.
                                    cadenaActual = cadenaActual + cadena[j];
                                    if(cadenaActual.length() > cadenaMaxima.length()) cadenaMaxima = cadenaActual;
                                    buscando = 0;
                              } else {
                                    cadenaActual = cadena[j];
                                    buscando = 1;
                              }
                        }
                  }
            }
            //CUANDO ACABA LA ITERACION DEVOLVEMOS EL RESULTADO.
            return cadenaMaxima;
      } else {
```

```
            //SI NO HAY CONSONANTES DEVUELVE LA CADENA VACIA.
            return "";
      }
}

bool esVC(string a){
      if((!esConsonante(a[0]))&&(esConsonante(a[1]))) return true;
      return false;
}

bool esCV(string a){
      if((esConsonante(a[0]))&&(!esConsonante(a[1]))) return true;
      return false;
}

string max(string cad1, string cad2){
      if(cad1.size() >= cad2.size()) return cad1;
      return cad2;
}

string max(string cad1, string cad2, string cad3){
      if((cad1.size() >= cad2.size())&&(cad1.size() >= cad3.size())) return cad1;
      if((cad2.size() > cad1.size())&&(cad2.size() > cad3.size())) return cad2;
      return cad3;
}
string combinar(string cadena, string izq, string der){
      string solucionI = "";
      string solucionD = "";
      string cadenaCentral = "";
      cadenaCentral = cadena.substr(((cadena.size()/2)-1),2);
      //CASOS BASE
      if(cadena.size() == 1){
            return "";
      }
      if(cadena.size() == 2){
            if (esCV(cadena)) return cadena;
            else return "";
      }
      if(cadena.size() == 3){
            if(esCV(cadenaCentral)) return cadenaCentral;
            else return der;
```

```
}
if(cadena.size() == 4){
      if(esCV(cadenaCentral)){
            return cadenaCentral;
      } else {
            return max(izq,der);
      }
}
if(cadena.size() == 5){
      if(esCV(cadenaCentral)){
            if(esCV(cadena.substr(3,2))) return cadenaCentral+cadena.substr(3,2);
            else return cadenaCentral;
      }else {
            string auxI = cadena.substr(0,2);
            string auxD = cadena.substr(2,2);
            if(esCV(auxI)&&esCV(auxD)) {
                  //LA CADENA FORMADA POR LAS CUATRO LETRAS CENTRALES ES CORRECTA.
                  return auxI+auxD;
            } else {
                  return max(der,izq);
            }
      }
}
if((cadena.size() == 6)||(cadena.size() == 7)){
      if (esCV(cadenaCentral)){
            if(esCV(cadena.substr(4,2))) cadenaCentral = cadenaCentral+cadena.substr(4,2);
            if(esCV(cadena.substr(0,2))) cadenaCentral = cadena.substr(0,2)+cadenaCentral;
            return cadenaCentral;
      }else{
            string auxI = cadena.substr(1,2);
            string auxD = cadena.substr(3,2);
            if(esCV(auxI)&&esCV(auxD)) {
                  //LA CADENA FORMADA POR LAS CUATRO LETRAS CENTRALES ES CORRECTA.
                  return max(auxI+auxD,der,izq);
            } else {
                  return max(der,izq);
            }
      }
} else {
  //SE OPTIENE LAS DOS LETRAS CENTRALES PARA POSTERIORMENTE PORDER IDENTIFICAR LOS CASOS.
  if(esCV(cadenaCentral)){//SI LA CADENA CENTRAL ESTA FORMADA POR UNA VOCAL SEGUIDA DE UNA CONSONANTE.
```

```
            //ITERACIÓN POR LA DERECHA DE LA CADENA HASTA QUE SE ACABE O HASTA QUE NO HAYA UNA COMBINACION VALIDA.
            //WHILE ...
            float y = 1;
            while((y <= cadena.size()-(cadena.size()/2))&&(esCV(cadena.substr(((cadena.size()/2)+y),2)))){
                  solucionD = solucionD + cadena.substr((cadena.size()/2)+y,2);
                  y+=2;
            }
//ITERACIÓN POR LA IZQUIERDA DE LA CADENA HASTA QUE SE ACABE O HASTA QUE NO HAYA UNA COMBINACION VALIDA.
            //WHILE ...
            float z = 1;
            while((((cadena.size()/2)-z-2) >= 0)&&(esCV(cadena.substr((cadena.size()/2)-z-2,2)))){
                  solucionI = cadena.substr((cadena.size()/2)-z-2,2) + solucionI;
                  z+=2;
            }
            return max(solucionI+cadenaCentral+solucionD,der,izq);
      } else {
            if (esVC(cadenaCentral)){//SI LA CADENA CENTRAL ESTA FORMADA POR UNA CONSONANTE SEGUIDA DE UNA VOCAL.
    //ENCONTRAR LA SIGUIENTE LETRA POR LA IZQUIERDA PARA VER SI SE PUEDE ACOPLAR AL PRINCIPIO DE LA CADENA CENTRAL.
                  string auxI = cadena.substr(((cadena.size()/2)-2),2);
                //ENCONTRAR LA SIGUIENTE LETRA POR LA DERECHA PARA VER SI SE PUEDE ACOPLAR POR EL FINAL DE LA CADENA.
                  string auxD = cadena.substr(((cadena.size()/2)),2);
    //COMPROBAR SI SE PUEDEN AÑADIR LAS DOS CADENAS A LA CADENA CENTRAL, UNA POR DELANTE Y OTRA POR DETRAS.

                  if(esCV(auxI)&&esCV(auxD)) {
                        //LA CADENA FORMADA POR LAS CUATRO LETRAS CENTRALES ES CORRECTA.
                        cadenaCentral = auxI+auxD;
                  }
                  else {
                        return max(der,izq);
                  }
    //ITERACIÓN POR LA DERECHA DE LA CADENA HASTA QUE SE ACABE O HASTA QUE NO HAYA UNA COMBINACION VALIDA.
                  //WHILE ...
                  float y = 2;
                  while((y <= cadena.size()-(cadena.size()/2))&&(esCV(cadena.substr(((cadena.size()/2)+y),2)))){
                        solucionD = solucionD + cadena.substr(((cadena.size()/2)+y),2);
                        y+=2;
                  }
    //ITERACIÓN POR LA IZQUIERDA DE LA CADENA HASTA QUE SE ACABE O HASTA QUE NO HAYA UNA COMBINACION VALIDA.
                  //wHILE ...
                  float z = 2;
                  while((((cadena.size()/2)-z-2) >= 0)&&(esCV(cadena.substr((cadena.size()/2)-z-2,2)))){
```

```
                        solucionI = cadena.substr((cadena.size()/2)-z-2,2) + solucionI;
                        z+=2;
                    }
                    return max(solucionI+cadenaCentral+solucionD,der,izq);
                }
            }
            return max(der,izq);
    }
}

string maxConsVocal(string cadena,int n){
      if(n <= 1){
            return iterativo(cadena);
      }
      string cad1 = cadena.substr(0,n/2);
      string cad2 = cadena.substr(n/2,n-(n/2));
      return combinar(cadena,maxConsVocal(cad1,(n/2)),maxConsVocal(cad2,(n-(n/2))));
}

string generarCadena(int n){
      string solucion;
      int num,prob;
      char c;
      string vocales = "aeiou";
      for(int i = 0; i < n; i++){
            prob=1+rand()%(10-1);
            if(prob > 4){
                  num = rand()%5;
                  solucion += vocales[num];
            }
            else {
                  //generar numero aleatorio entre 97 - 122 (rango de minusculas)
                  num=97+rand()%(122-97);
                  //combertir el numero a char
                  c = (char)num;
                  //añadir el caracter a la cadena
                  solucion += c;
            }
      }
      return solucion;
}
```

```
////////////////////////////////////////////////////////////
////////////        PROGRAMA PRINCIPAL        ///////////////
////////////////////////////////////////////////////////////

int main (void)
{
      srand(time(NULL));
      //Variables.
      int m,base;
      long long int val;
      base = 0;
      long long int tiempoIni,tiempoFin;
      ////////////////////////
      /*
  //Se pintan los indices.
  printf("n\t");
  for(int i = 1;i <= n; i+=incremento) printf("%d\t",i);
    printf("\nTiempo\t");
  ////////////////////////
   fprintf(stderr,"Se van a genererar %d datos\n", _NUMEROCASOS);
  for(int i = 1; i <= n; i+=incremento){
          tiempoIni = mseconds();
       */


      ////////////////////////////////////////////////


      string cadena = generarCadena(500);
      cerr << "La cadena es " << cadena << endl;
      cerr << "Cadena final (iterativa): " << iterativo(cadena) << endl;
      cerr << "Cadena final (recursivo): " << maxConsVocal(cadena,cadena.length()) << endl;

      ////////////////////////////////////////////////
      /*
          tiempoFin = mseconds();
          barraProgreso(i);
          val=(tiempoFin-tiempoIni);
          /*
          if(val<0){
```

```
            fprintf(stderr, "error\n");
          }

    //fprintf(stderr, "%lld %lld %d\n", tiempoIni, tiempoFin, abs(val));

    if(val>0)printf("%lld\t",val);
       */


}
```