Proyecto POO1

Teoría 2 - 2020.02

Universidad de Ingeniería y Tecnología

Integrantes:

- Pedro Franciso Arteta Flores - 202010198
- Ivana Castañeda Luperdi - 202010061
- Camila Xiomara Cornejo Moreno - 202010236

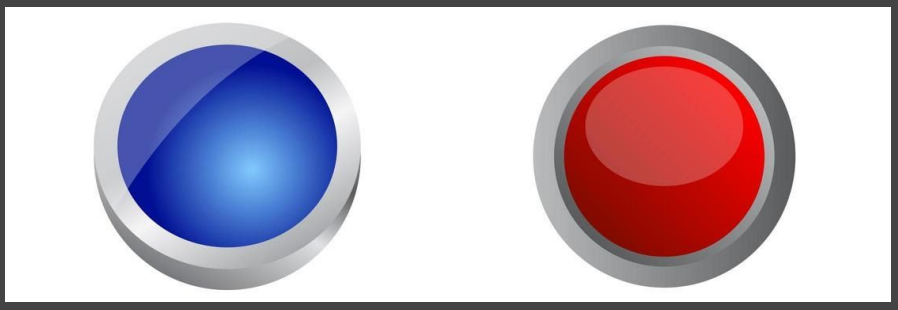
Porcentaje de participación:

- Pedro Arteta - 100%
- Ivana Castañeda - 100%
- Camila Cornejo - 100%

Contexto

El señor científico loco nos contacto para solicitar nuestros servicios. Él requiere que diseñemos un software que controle las claves de ingreso a su laboratorio, debido a que su sistema de seguridad anterior fue hackeado, con el siguiente requerimiento:

El visitante se encuentra al ingreso con dos botones de luz (rojo y azul) que debe presionar 15 veces. El problema está en que no conoce el orden correcto de los colores.



La clave ahora consiste en utilizar los colores rojo y azul, en 5 secuencias correctas de 3 colores cada una (en total 15 colores), que corresponden a 5 de las siguientes combinaciones:
•rojo azul rojo •azul rojo azul

Es decir, las combinaciones posibles serían:
(**a**) *azul rojo azul rojo azul rojo rojo azul rojo azul rojo azul rojo azul rojo.*
(**b**) *azul rojo azul rojo azul rojo rojo azul rojo azul rojo azul azul rojo azul.*

Mientras que la siguiente combinación sería incorrecta:
(**c**) *rojo azul azul rojo azul rojo rojo azul azul azul rojo azul azul rojo azul.*

Luego, la clave ingresada se transforma a un número binario, donde :

- rojo azul rojo corresponde a 1
- azul rojo azul corresponde a 0

Las combinaciones posibles anteriores serían entonces:
(a) 01101
(b) 01100

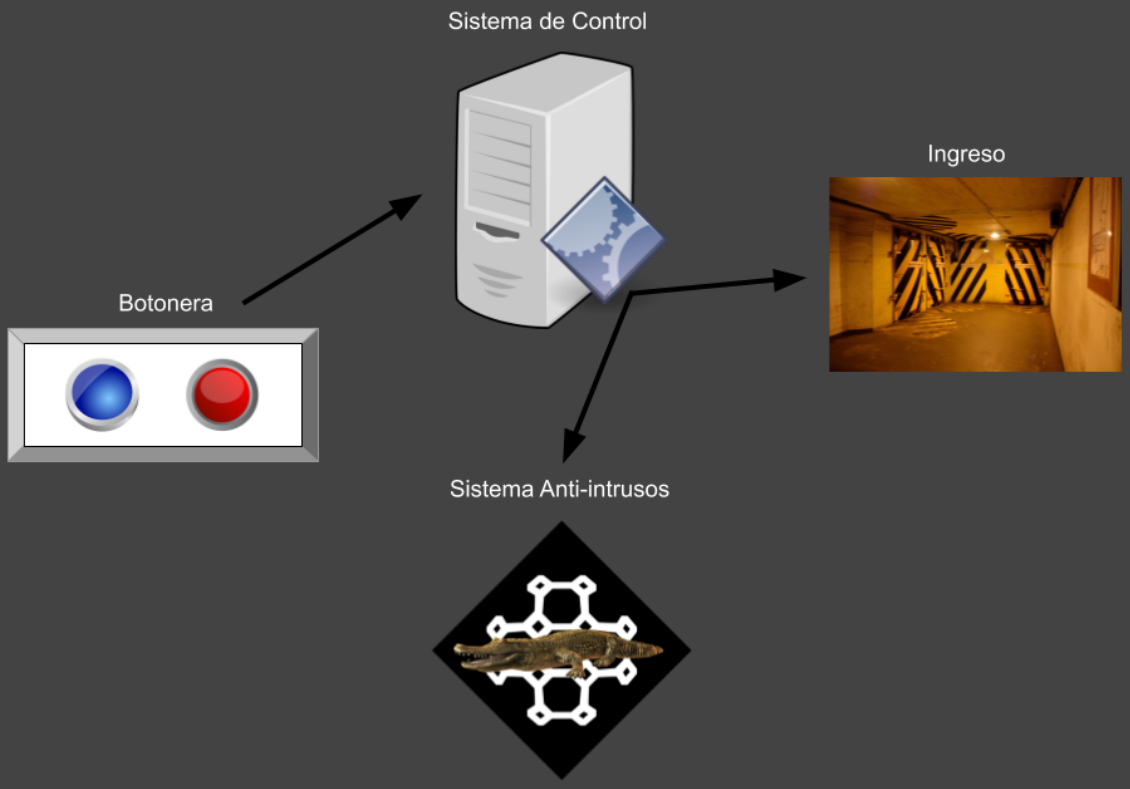
Finalmente, la clave se transforma a sistema decimal, donde debe cumplir con la siguiente condición: el número decimal debe ser un número primo.

En los ejemplos anteriores, tendríamos:
(a) 13→clave correcta
(b) 12→clave incorrecta

Tu tarea es desarrollar un código que acepte/rechace una cadena según las reglas mencionadas. Además, el científico reemplaza ahora a los tiburones por una trampa de lagartos hambrientos.

Observaciones

Según lo dicho anteriormente realizamos un posible diagrama que contiene las conexiones del sistema.



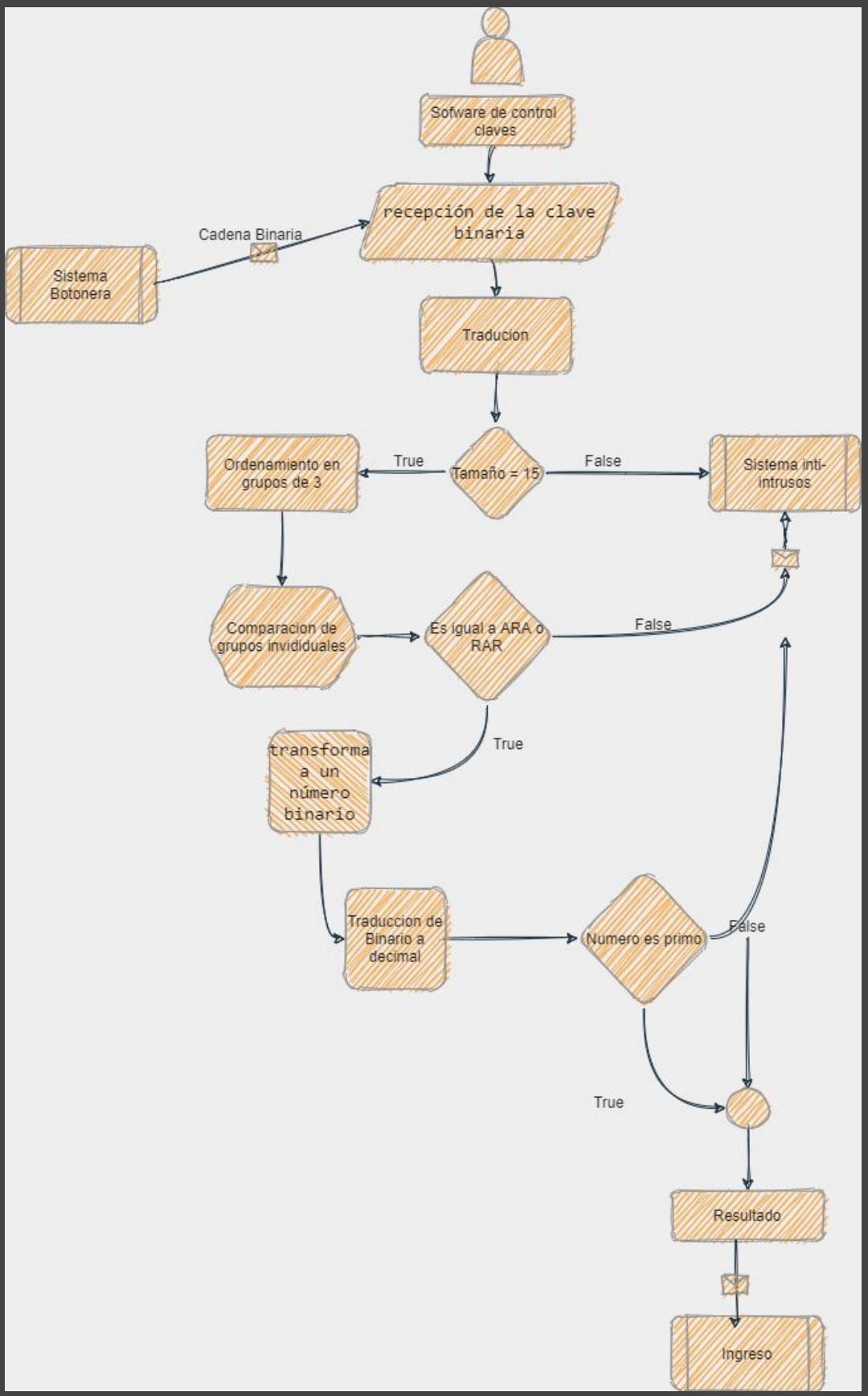
Nuestro software operará en el "sistema de control", recibiendo los datos enviados por la botonera, procesando estos para determinar si la clave es correcta o incorrecta. Si fuera correcta, se enviará un código de confirmación a "ingreso", en caso que fuera lo contrario enviará un código de negación, además activaría el sistema anti-intrusos.

El dispositivo “botonera” solo envía una secuencia binaria, donde 1 se interpreta como “Azul” y 0 como “Rojo”

Objetivos planteados

1. Desarrollar un software que acepte o rechace una clave ingresada.
    a. *Si en caso la clave fuera incorrecta, activar el sistema de seguridad.*
    b. *Debe ser seguro y eficiente para evitar posibles hackeos (internos como externos).*

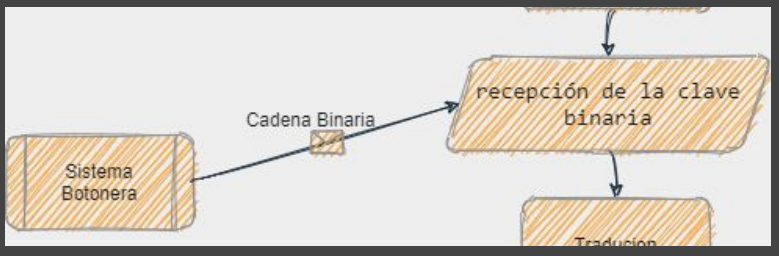
Diagrama de Flujo



Explicación paso a paso del diagrama de flujo

Lo primero que necesitamos para llevar a cabo el problema es entender lo que este nos pide, que en este caso sería determinar si la contraseña ingresada es correcta. Esto mediante ciertos pasos que irán construyendo el software que diseñamos para realizar esta acción.

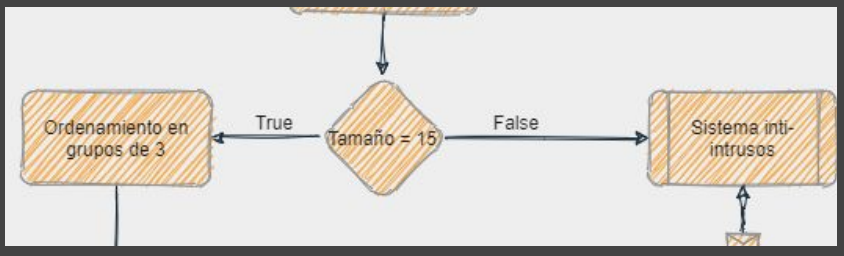
1. Para comenzar con el proceso, se recepciona la clave binario enviada por la "botonera".



2. Luego, este input es procesado a través del traductor el cual transforma la secuencia binaria de 0 y 1 en una secuencia de **A** y **R**, siendo esta más entendible para nosotros. Donde **A** representa la pulsación del botón **AZUL** y **R** la del **ROJO**.
   Por ejemplo:

   010010010010110 → RARRARRARRARAAR

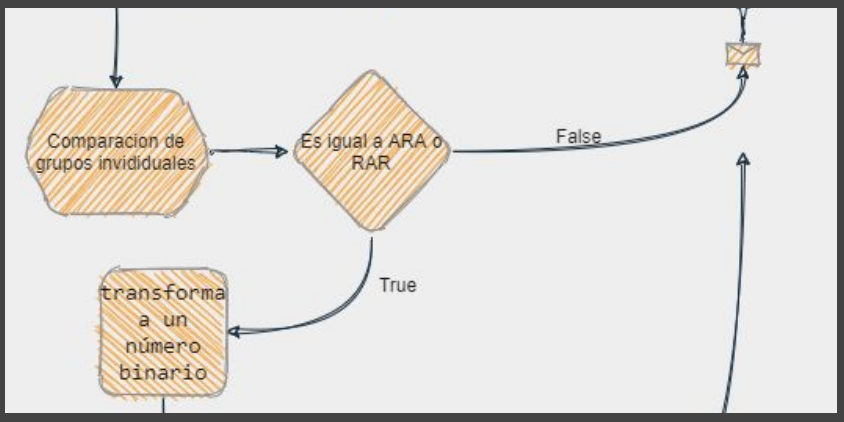
3. Posteriormente, evaluamos si esta clave cuenta con la cantidad de caracteres indicados (**15 dígitos**). En caso la secuencia cuente con una cantidad diferente a la indicada, ya sea menor o mayor, el programa rechazará la clave y se comunicará con el sistema *anti-intrusos* que la devolverá como un FALSE, interrumpiendo así la secuencia.



4. Una vez se haya verificado que la clave cuenta con los 15 dígitos, se procederá a agrupar la secuencia de **A** y **R** en conjuntos de 3 caracteres.
   Por ejemplo:

RARRARRARRARAAR → {RAR,RAR,RAR,RAR,AAR}

5. Ya habiendo establecido esto, se procederá a comparar cada uno de los grupos con las secuencias indicadas por el científico (**ARA** o **RAR**) para luego traducir estas a un número (**0** o **1**), posteriormente reagrupados los valores. En caso de que alguna de las secuencias fuera rechazada luego de la comparación, el programa rechazará asimismo la clave y se comunicará con el sistema *anti-intrusos* que la devolverá como FALSE, interrumpiendo así la secuencia.



Por ejemplo:

Conjunto → Verificación > resultado

RAR → True > 1

RAR → True > 1

RAR → True > 1

RAR → True > 1

AAR → False > sistema anti-intrusos

6. Nuestro siguiente paso será traducir el número en base binaria, anteriormente formado, a la base decimal.

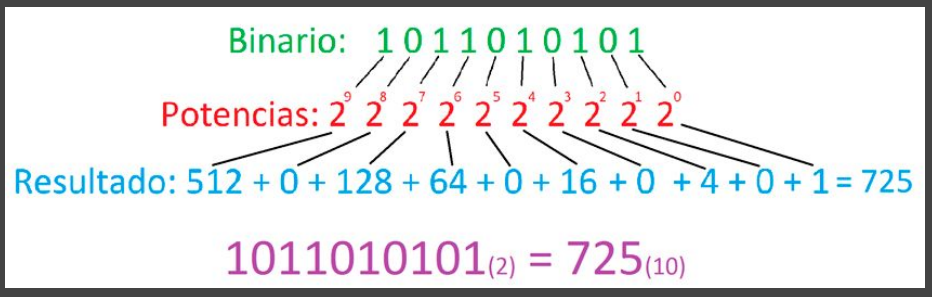


Imagen extraida del sitio web *"cual-es-mi-ip.online"*

7. Finalmente, se determinará si este número decimal es primo para enviar el resultado al sistema de ingresos. En caso de que el programa rechazara la clave se comunicará además con el sistema *anti-intrusos* que la devolverá como un FALSE, interrumpiendo así la secuencia.

Desarrollo del código

```
#include <iostream>
#include <string>
#include <math.h>
using namespace std;

int binario_decimal(int binario){
    int resultado=0;
    for(int i=0; i<5; i++){
        resultado += (binario%10)*pow(2,i);
        binario/=10;
    }
    return resultado;
}


bool software(string codigo_binario_ingreso){
  int binario=0;
  int decimal = 0;
  string secuencia[5];

  if (codigo_binario_ingreso.size() != 15)
    {
      return false;
    }

  // 0001001-> ARA
  for (int i=0; i < codigo_binario_ingreso.size(); i++){
        switch(codigo_binario_ingreso[i]) {
            case '1': codigo_binario_ingreso[i]='A';
                break;
            case '0': codigo_binario_ingreso[i]='R';
                break;
            default: return false;
        }
    }
  //arararararararararar -> {ara,rar,ara,rar,ara}
  for (int i=0; i <5; i++)
    {
      secuencia[i]= codigo_binario_ingreso.substr(i*3,3);
```

```
      if (secuencia[i]=="RAR"){
        binario= binario*10 +1;
      }
      else if (secuencia[i]=="ARA"){
        binario= binario*10;
      }
      else { return false; }
    }

  decimal = binario_decimal(binario);

  // primos
  if (decimal <= 1) return false;

  else if (decimal == 2) return true;

  else
    {
    for (int i=2; i<decimal; i++)
      {
        if (decimal % i == 0) return false;
      }
    return true;
    }

  return false;
}

int main() {
  cout << software("101101101010010");
}
```

Explicación paso a paso del código

1. La estructura principal del software recibe como parámetro una cadena de texto. Esto debido a que la clave a comparar puede estar formada por un número de **n** dígitos y determinamos que este tipo de variable es la mejor opción para el manejo de la información, gracias a su facilidad de modificación y los múltiples métodos que presenta.

```
string software(string codigo_binario_ingreso){
  ...
}
```

Esta información que fue enviada al software se procesa y devuelve posteriormente como TRUE o FALSE dependiendo del caso.

2. Luego de esto, se definen las variables, las cuales almacenarán los datos necesarios para el proceso.

```
int binario=0;
int decimal = 0;
string secuencia[5];
```

Estas variables son las siguientes:

- **Binario**: almacena un número de tipo entero, que guardará el binario obtenido de comparación y traducción de los grupos formados en paso X.

- **Decimal**: almacena la transformación del anterior número a la base decimal, siendo las variables que maneja de tipo entero.

- **Secuencia**: guarda los grupos de 3 dígitos formados en paso X, siendo de tipo entero debido a que la clave es de este mismo tipo.

3. Ya teniendo esto, se verifica si el código binario ingresado es de 15 dígitos o no. En caso de que no cumpla con esta cantidad, se retorna como respuesta un FALSE y se interrumpirá la secuencia. Por otro lado, si llega a cumplir con este requisito, el programa continuará con normalidad.

```
if (codigo_binario_ingreso.size() != 15)
{
   return "false";
}
```

El método size devuelve como parámetro el número de elementos de un contenedor.

4. El siguiente paso será iterar al binario ingresado para que los 1 y 0 que contiene se conviertan en una secuencia más entendible para nosotros como los es **A** y **R**. Estas letras representan respectivamente la pulsación de los botones **AZUL** y **ROJO**.

   En caso el código binario contenga algo diferente de 0 o 1, el sistema retornará como respuesta un FALSE y se interrumpirá la secuencia.

```
for (int i=0; i < codigo_binario_ingreso.size(); i++){
   switch(codigo_binario_ingreso[i]) {
       case '1': codigo_binario_ingreso[i]='A';
           break;
       case '0': codigo_binario_ingreso[i]='R';
           break;
       default: return "false";
   }
}
```

*La estructura repetitiva usada es **for** debido a que iteramos la secuencia binaria por índice, a causa de una mejor legibilidad. Además usamos la estructura de control **switch** debido a que nos permite verificar el valor de una variable sin tener que usar operadores adicionales.*

5. A continuación se itera esta cadena de **A** y **R**, creando subcadenas de 3 caracteres. El programa compara la subcadena con las dadas por el científico (**RAR** o **ARA**); en caso sea **RAR**, tomará la variable binario y la multiplicará por 10, esto nos permitirá desplazar la secuencia de números para obtener una espacio, dándonos 0 para luego sumarle 1. Si es **ARA**, el binario lo multiplicará por 10 dándonos 0 como resultado. En caso no se cumpla con ninguna de las condiciones, se retornará como respuesta un FALSE y se interrumpirá la secuencia.

```
for (int i=0; i <5; i++)
{
   secuencia[i]= codigo_binario_ingreso.substr(i*3,3);

   if (secuencia[i]=="RAR"){
       binario= binario*10 +1;
   }
   else if (secuencia[i]=="ARA"){
       binario= binario*10;
   }
   else { return "false"; }
}
```

*La estructura repetitiva usada es **for** debido a que iteramos en un rango de 0 a 4, a causa de poder crear subcadenas de 3 caracteres usando la función **substr**, para ese propósito le indicamos como primer parámetro la función matemática **F(x)=i*3**,con esta función obtenemos el valor de los índices de los primero elementos de cada subcadena; como segundo parámetros colocamos el número 3 debido a que indica la longitud a abarcar. Además usamos la estructura de*

*control **if-else** para comprar estas subcadenas con las dadas por el científico.*

6. Luego, el binario obtenido se convierte en un número decimal. A través de la obtención del dígito que se encuentre en la posición n y multiplicando por la potencia de 2 a la n. Posteriormente sumamos todos estos resultados.

```
int binario_decimal(int binario){
    int resultado=0;
    for(int i=0; i<5; i++){
        resultado += (binario%10)*pow(2,i);
        binario/=10;
    }
    return resultado;
}
```

7. Finalmente se evalúa el resultado para verificar si es primo. Esto siguiendo las siguientes condiciones:
    a. Si es 1 o menor a 1, retornará como respuesta un FALSE y se interrumpirá la secuencia.
    b. Si el resultado es 2, cumple con la condición retornando como respuesta un TRUE.
    c. En caso contrario, se iterará todos los valores que se encuentren entre 1 y el mismo número, buscando que el residuo de la división del número entre los número que se encuentren en el rango anteriormente mencionado den como resultado 0, y cumpliendo así con la condición retornando como respuesta un TRUE. Si en alguna iteración el residuo fuera diferente de 0 se retornará como respuesta un FALSE y se interrumpirá la secuencia.

```
if (decimal < 1) return "false";

else if (decimal == 2) return "true";

else
{
    for (int i=2; i<decimal; i++)
    {
        if (decimal % i == 0) return "false";
    }
    return "true";
}
```

Emulador

Con el fin de probar la efectividad de nuestro software, desarrollamos también un emulador que se asemeja al diagrama de conexiones del sistema, que nos entrega un informe del ingreso de clave.

```
1 Informe N
2 Razón
3 clave binaria Estado
4 Mensaje del sistema activado
```

Ejemplo de ingreso manual:

```
Informe0
(1) Ingrese clave:ARAARAARAARA
 Size
101101101101 0
Activacion del sistema anti-intrusos
Cierre de compuertas
Apertura de la poso con caimanes

Informe1
(1) Ingrese clave:ARAARAARAARAARA
 Primero <=1
101101101101101 0
Activacion del sistema anti-intrusos
Cierre de compuertas
Apertura de la poso con caimanes

Informe2
(1) Ingrese clave:
...
```

Gracias a este emulador logramos corregir errores y reducir el código.

Desarrollo del código

El main organiza todo los pasos de este algoritmo. Primero solicita el número de solicitudes de claves, permitiendo que como máximo se puede ingresar 4294967295 debido al tipo de variable (*unsigned int*). Esto para luego solicitar el tipo de ingreso de código que podrían ser 3. En caso se ingrese un numero diferente a los indicados, se tomará como caso base el ingreso de código binario acortado.

Luego, realizará la solicitud de claves al software, para dirigir su resultado a los sistemas respectivos. Todos estos datos se entregarán con el formato de informe ya mostrado.

***Archivo** main.cpp*

```
#include <iostream>
#include "botonera.h"
#include "puerta.h"
#include "software.h"
#include "sistema_seguridad.h"
#include <string>

using namespace std;

string selector(int opcion){
   switch (opcion) {
       case 1:return codigo_binario_manual();

       case 2:return codigo_binario_acortado();

       case 3:return codigo_binario();
   }
   return codigo_binario_acortado();
}

int main() {
   string binario="";
   bool confirmacion = true;
   int n_repeticiones = 0; int opcion=0;
   cout << "Ingrese el numero de repeticiones: "; cin >>
n_repeticiones;
   cout << "Selecione el tipo de ingreso \n" ;
   cout << "(1) Ingreso manual\n" ;
   cout << "(2) Ingreso aletorio acortado\n" ;
   cout << "(3) Ingreso aletorio\n" ;
   cout << " - - Ingrese numero de opcion\n" ;
   cin >> opcion;

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

       cout << "Informe" << i << endl;
       binario = selector(opcion);

       confirmacion = software(binario);
       cout <<binario<< " "<<confirmacion;
       cout << endl;

       activiacion_ingreso(confirmacion);
       activiacion_sistema_seguridad(confirmacion);
```

```
        cout << endl;
    }
}
```

Para esto, editamos el código del software para que imprima en consola la razón por la cual la clave no fue aceptada.

Archivo software.cpp

```
#include <iostream>
#include <string>
#include <math.h>
#include "software.h"
using namespace std;

int binario_decimal(int binario){
          int resultado=0;
          for(int i=0; i<5; i++){
              resultado += (binario%10)*pow(2,i);
              binario/=10;
          }
          return resultado;
      }

bool software(string codigo binario ingreso){
          int binario=0;
          int decimal = 0;
          string secuencia[5];

          if (codigo_binario_ingreso.size() != 15)
          {
              cout << "Size\n";
              return false;
          }

          for (int i=0; i < codigo_binario_ingreso.size();
      i++){
              switch(codigo_binario_ingreso[i]) {
                  case '1': codigo_binario_ingreso[i]='A';
                      break;
                  case '0': codigo_binario_ingreso[i]='R';
                      break;
                  default:
                      cout << "traductor A y R\n";
                      return false;
              }
          }

          for (int i=0; i <5; i++)
          {
              secuencia[i]=
```

```
    codigo_binario_ingreso.substr(i*3,3);

            if (secuencia[i]=="RAR"){
                binario= binario*10 +1;
            }
            else if (secuencia[i]=="ARA"){
                binario= binario*10;
            }
            else { cout << "Agrupamiento\n";return false;
    }
        }

        decimal=binario_decimal(binario);


        if (decimal <= 1) {cout << "Primero <=1 \n";
    return false; }

        else if (decimal == 2) return true;

        else
        {
            for (int i=2; i<decimal; i++)
            {
                if (decimal % i == 0) {cout << "Primero %
    != 0 \n"; return false; }
            }
            return true;
        }

    }
```

Archivo *software.h*

```
#ifndef EMULADOR_SOFTWARE_H
#define EMULADOR_SOFTWARE_H

#include <string>
using namespace std;

bool software(string codigo_binario_ingreso);
int binario_hexadecimal(int binario);

#endif //EMULADOR_SOFTWARE_H
```

Además de desarrollar los 3 sistemas restantes:

1. **BOTONERA**: se encarga de generar la clave de ingreso, contando con 3 modos:

a. *Ingreso manual*: en este tipo de ingrese se obtiene la secuencia que ingresa por consola y lo transforma en la serie binaria que envía la botonera

***Sección del archivo** botonera.cpp*

```
#include <iostream>
#include "botonera.h"
#include <cstdlib>
#include <string>
...
string codigo_binario_manual(){
   string codigo_binario_ingreso;
   cout << "(1) Ingrese clave: " ;
   cin >> codigo_binario_ingreso;
   for (int i=0; i <
codigo_binario_ingreso.size(); i++){
       switch(toupper(codigo_binario_ingreso[i]))
{
           case 'A':
codigo_binario_ingreso[i]='1';
               break;
           case 'R':
codigo_binario_ingreso[i]='0';
               break;
           default:
codigo_binario_ingreso[i]='2';
       }
   }
   return codigo_binario_ingreso;
}
```

b. *Ingreso aleatorio*: la clave es generada aleatoriamente, a través de la función `randint()` la cual nos genera un número aleatorio dependiendo del rango determinado. Con esto variamos el número de caracteres y también los mismos caracteres, creando así claves diferentes para cada caso.

***Sección del archivo** botonera.cpp*

```
#include <iostream>
#include "botonera.h"
#include <cstdlib>
#include <string>
...
int randint(int n) {
   return rand() % n;
}
```

```
string codigo_binario(){
    string codigo;
    int numero=0;
    int limite = rand()%20;
    for (int i=0; i<limite; i++){
        numero=randint();
        if (numero==0){
            codigo +="0";
        }
        else{
            codigo +="1";
        }
    }
    return codigo;
}
```

c. *Ingreso aleatorio reducido*: la clave es generada aleatoriamente, a través de la función `randint()`, pero el número de elementos ya está determinado y con esto variamos solo el número de secuencias repetidas. Debido a esto, se reduce también el número de claves posibles a generar pero, por otro lado, se aumenta la tasa de efectividad.

***Sección del archivo** botonera.cpp*

```
#include <iostream>
#include "botonera.h"
#include <cstdlib>
#include <string>
...
int randint(int n) {
    return rand() % n;
}

string codigo_binario_acortado(){
    string codigo;
    int numero=0;

    for (int i=0; i<2; i++){
        numero=randint(3);
        if (numero==0){
            codigo +="101";
        }
        else if (numero==1){
            codigo +="010";
        }
        else{
            codigo +="000";
        }
```

```
    }
    return codigo;
}
```

*La función **rand()** genera un número pseudo aleatorio de distribución no uniforme, entregando valores de 0 a (N-1) siendo N el número máximo.*

***Archivo** botonera.h*

```
#ifndef EMULADOR_BOTONERA_H
#define EMULADOR_BOTONERA_H
#include <string>
using namespace std;
int randint();
string codigo_binario();
string codigo_binario_manual();
string codigo_binario_acortado();

#endif //EMULADOR_BOTONERA_H
```

***Archivo** botonera.cpp*

```
#include <iostream>
#include "botonera.h"
#include <cstdlib>
#include <string>
using namespace std;

int randint(int n) {

   return rand() % n;
}

string codigo_binario(){
   string codigo;
   int numero=0;
   int limite = randint(20);
   for (int i=0; i<limite; i++){
       numero=randint(3);
       if (numero==0){
           codigo +="0";
       }
       else{
           codigo +="1";
       }
   }
   return codigo;
}

string codigo_binario_acortado(){
```

```
    string codigo;
    int numero=0;

    for (int i=0; i<2; i++){
        numero=randint(3);
        if (numero==0){
            codigo +="101";
        }
        else if (numero==1){
            codigo +="010";
        }
        else{
            codigo +="000";
        }
    }
    return codigo;
}

string codigo_binario_manual(){
    string codigo_binario_ingreso;
    cout << "(1) Ingrese clave: " ;
    cin >> codigo_binario_ingreso;
    for (int i=0; i < codigo_binario_ingreso.size();
i++){
        switch(toupper(codigo_binario_ingreso[i])) {
            case 'A': codigo_binario_ingreso[i]='1';
                break;
            case 'R': codigo_binario_ingreso[i]='0';
                break;
            default: codigo_binario_ingreso[i]='2';
        }
    }
    return codigo_binario_ingreso;
}
```

2. **PUERTA:** este sistema recepciona el resultado dado por el software, si el resultado es TRUE accionara la secuencia respectiva.

***Archivo** puerta.h*

```
#ifndef EMULADOR_PUERTA_H
#define EMULADOR_PUERTA_H

void activiacion_ingreso(bool mensaje);

#endif //EMULADOR_PUERTA_H
```

***Archivo** puerta.cpp*

```
#include "puerta.h"
#include <iostream>

using namespace std;

void activiacion_ingreso(bool mensaje){
   if (mensaje) cout << "Bienvenido cientifico" <<
endl;
}
```

3. **SISTEMA SEGURIDAD**: este sistema recepciona el resultado dado por el software, si el resultado es FALSE se negará y dará como resultado TRUE accionando así la secuencia respectiva.

***Archivo** sistema_seguridad.h*

```
#ifndef EMULADOR_SISTEMA_SEGURIDAD_H
#define EMULADOR_SISTEMA_SEGURIDAD_H

void activiacion_sistema_seguridad(bool mensaje);

#endif //EMULADOR_SISTEMA_SEGURIDAD_H
```

***Archivo** sistema_seguridad.cpp*

```
#include <iostream>
#include "sistema_seguridad.h"
using namespace std;

void activiacion_sistema_seguridad(bool mensaje){
   if (!(mensaje)){
        cout << "Activacion del sistema
anti-intrusos"<< endl;
        cout << "Cierre de compuertas"<< endl;
        cout << "Apertura de la poso con caimanes"<<
endl;
   }
}
```

Conclusiones

Para desarrollar el software tuvimos que tener en cuenta que al primer error que cometa la persona al ingresar el código, este debería ser capaz de activar un sistema de seguridad que impida hackeos internos o externos, y que además debía ser lo suficientemente seguro y eficiente para garantizar que estos hackeos no vuelvan a ocurrir. De esta manera fuimos desarrollando el código obteniendo como resultado final para este que se llegue a evaluar de manera exitosa mediante varias acciones que ejecuta el código si la clave de ingreso es correcta o si el

sistema de seguridad debe ser activado por la posibilidad de habernos encontrado con un hacker.