|  |  |  |
| --- | --- | --- |
| **CARRERA:**  Ingeniería de Software | **GUÍA**  No. 01 | **TIEMPO ESTIMADO:**  1 mes |
| **ASIGNATURA:**  Estructura de datos  NRC: 2967 | **FECHA DE ELABORACION:** 23-10-2019  **SEMESTRE**: septiembre 2019 – febrero 2020 | |
| **TÍTULO:**  Backtracking - Combinaciones | **DOCENTE:** Ing. Fernando Solís | |

**OBJETIVO**

Utilizar Backtracking para realizar un proyecto el que tenga códigos u algoritmos más eficientes.

**INSTRUCCIONES**

**Backtracking**

Enumerar sistemáticamente las alternativas que existen en cada momento para dar con la solución a un problema.  
Se prueba una alternativa, guardando memoria del resto de alternativas.  
Si no damos con la solución, podemos dar marcha atrás (backtracking) y probar otra alternativa.

Esta técnica tan trivial resulta especialmente útil para los problemas de búsqueda en estructuras de datos. Especialmente si no existe a priori ningún criterio de búsqueda mejor o peor. El inconveniente es que hay que explorar sistemáticamente todas las alternativas, lo que resulta muy lento en general.

Como hemos visto, un algoritmo de backtracking tiene ciertos requerimientos: un espacio de búsqueda, esto es un conjunto de posibilidades o "estados" del problema a resolver. En general, un estado puede estar definido por el elemento de la estructura de datos que queremos tratar en cada instante. Por ejemplo, en un árbol, cada uno de sus nodos, un estado inicial, el punto de partida del problema, un conjunto de estados finales. Estos se definen mediante alguna característica, criterio o condición. Por ejemplo, todos los nodos de un árbol que almacenen un número entero igual a 10 ,una memoria de estados. Generalmente una pila donde guardamos los estados de búsqueda, es decir las alternativas posibles en cada momento.

**ACTIVIDADES**

1. **Ubicación de recursos**
2. El grupo de 2 personas
3. Herramienta para C++, llamada Code::Blocks
4. **Planteamiento del problema**

Elaborar un algoritmo mediante el uso de Backtracking que genere las posibles combinaciones que pueden realizarse del factorial de un numero

1. **Entregable (s)**

**Clase Combinación.h**

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Module: Convinacion.h

\* Author: Luis Carvajal, Elian Llorente

\* Modified: miércoles, 6 de noviembre de 2019 1:04:26

\* Purpose: Declaration of the class Convinacion

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#if !defined(\_\_UML\_Class\_Diagram\_2\_Convinacion\_h)

#define \_\_UML\_Class\_Diagram\_2\_Convinacion\_h

class Convinacion

{

public:

void backtraking(int\* str, int k);

void archivo();

void mostrar(int\* str, int k);

protected:

private:

};

#endif

**Clase Combinación.cpp**

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Module: Convinacion.cpp

\* Author: Luis Carvajal, Elian Llorente

\* Modified: miércoles, 6 de noviembre de 2019 1:04:26

\* Purpose: Implementation of the class Convinacion

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#include "Convinacion.h"

#include<fstream>

#include<iostream>

using namespace std;

int n=0,ev,count;

ofstream g("out.txt");

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

// Name: Convinacion::backtraking(int\* str, int k)

// Purpose: Implementation of Convinacion::backtraking()

// Parameters:

// - str

// - k

// Return: void

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

void Convinacion::backtraking(int\* st, int k)

{

if(k==n+1){

for(int i = 1; i < k; i++)

g << \*(st+i) << " ";

count++;

g<<" Solucion "<<count<<endl;

}else{

for(int i = 1; i <= n; i++)

{

\*(st+k) = i;

ev=1;

for(int j = 1; j < k; j++)

if(\*(st+j) == \*(st+k))

ev=0;

if(ev)

backtraking(st, k+1);

}

}

}

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

// Name: Convinacion::archivo()

// Purpose: Implementation of Convinacion::archivo()

// Return: void

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

void Convinacion::archivo(){

FILE \*file1 , \*file2;

int data1 =0;

file1 = fopen ( "out.txt", "r" );

file2 = fopen ( "out2.txt" , "w" );

while ( (data1 = fgetc ( file1 )) != EOF ) {

fputc ( data1, file2 );

}

fclose ( file1 );

fclose ( file2 );

}

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

// Name: Convinacion::mostrar(int\* str, int k)

// Purpose: Implementation of Convinacion::mostrar()

// Parameters:

// - str

// - k

// Return: void

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

void Convinacion::mostrar(int\* st, int k)

{

cout<<"\n\n\n\nNumeros que se combinaran: "<<endl;

for(int i = 1; i <= n; i++)

{

\*(st+k) = i;

cout<<\*(st+k)<<" ";

}

cout<<endl;

}

**Clase Main.cpp**

#include"Convinacion.cpp"

#include <iostream>

#include <stdio.h>

#include <string.h>

#include <conio.h>

#include <stdlib.h>

#include <pthread.h>

#include <windows.h>

void \*moverMarquesina(void \*marq);

void gotoxy(short x, short y);

using namespace std;

int main(){

pthread\_t thread1;

pthread\_t thread2;

pthread\_create(&thread1,NULL,moverMarquesina,NULL);

main:

int \*st,s;

int opt;

Convinacion c;

system("cls");

cout<<"\n\n\n\n\n\n||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||";

cout<<"\n|||| ||||\n";

cout<< "|||| UNIVERSIDAD DE LAS FUERZAS ARMADAS ESPE ||||";

cout<<"\n|||| ESTRUCTURA DE DATOS ||||";

cout<<"\n|||| INTEGRANTES: CARVAJAL LUIS - ELIAN LLORENTE ||||";

cout<<"\n|||| NRC: 2967 ||||";

cout<<"\n||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||\n\n";

cout<<"ELIGA UNA OPCION\n\n";

cout<<"1. APLICATIVO\n2. SOLUCIONES\n3. AYUDA\n4. SALIR \n";

cin>>opt;

switch(opt)

{

case 1:

system("cls");

st=(int\*)calloc(n,sizeof(int\*)\*n);

cout<<"\n\n\n\nIngrese un numero para generar las combinaciones:\n\n\n\n";

cin>>n;

c.backtraking(st,1);

c.mostrar(st,1);

system("pause");

system("cls");

goto main;

case 2:

system("cls");

c.archivo();

system("out2.txt");

system("pause");

goto main;

case 3:

system("cls");

system("Ayuda.chm");

goto main;

case 4:

exit(0);

default:

cout<<"Ingrese una opcion correcta"<<endl;

system("pause");

goto main;

}

}

void gotoxy(short x, short y) {

COORD pos = {x, y};

SetConsoleCursorPosition(GetStdHandle(STD\_OUTPUT\_HANDLE), pos);

}

void\* moverMarquesina(void \*marq){

int a, b, c, n=0,letra=39,pos=1,cont=39,aux,cont1=39,auxg;

char t[50] ="BIENVENIDOS A LA ESTRUCTURA DE DATOS",auxt[39]=" ";

do{

for (a=0;a<42;a++){

aux=pos;

for(b=39;b>cont;b--){

gotoxy(pos,1);

cout<<t[b];

pos--;

}

aux++;

pos=aux;

cont--;

Sleep (75);

if(a==40){

break;

}

}

for(a=3;a<70;a++){

gotoxy(a-1,1);

cout<<" ";

gotoxy(a,1);

cout<<t;

Sleep (75);

}

pos=70;

auxg=69;

for (a=0;a<41;a++){

gotoxy(auxg,1);

cout<<" ";

aux=pos;

for(b=0;b<=cont1;b++){

gotoxy(pos,1);

cout<<t[b];

pos++;

}

cont1--;

aux++;

pos=aux;

auxg++;

Sleep (75);

}

cont1=39;

letra=39;

pos=1;

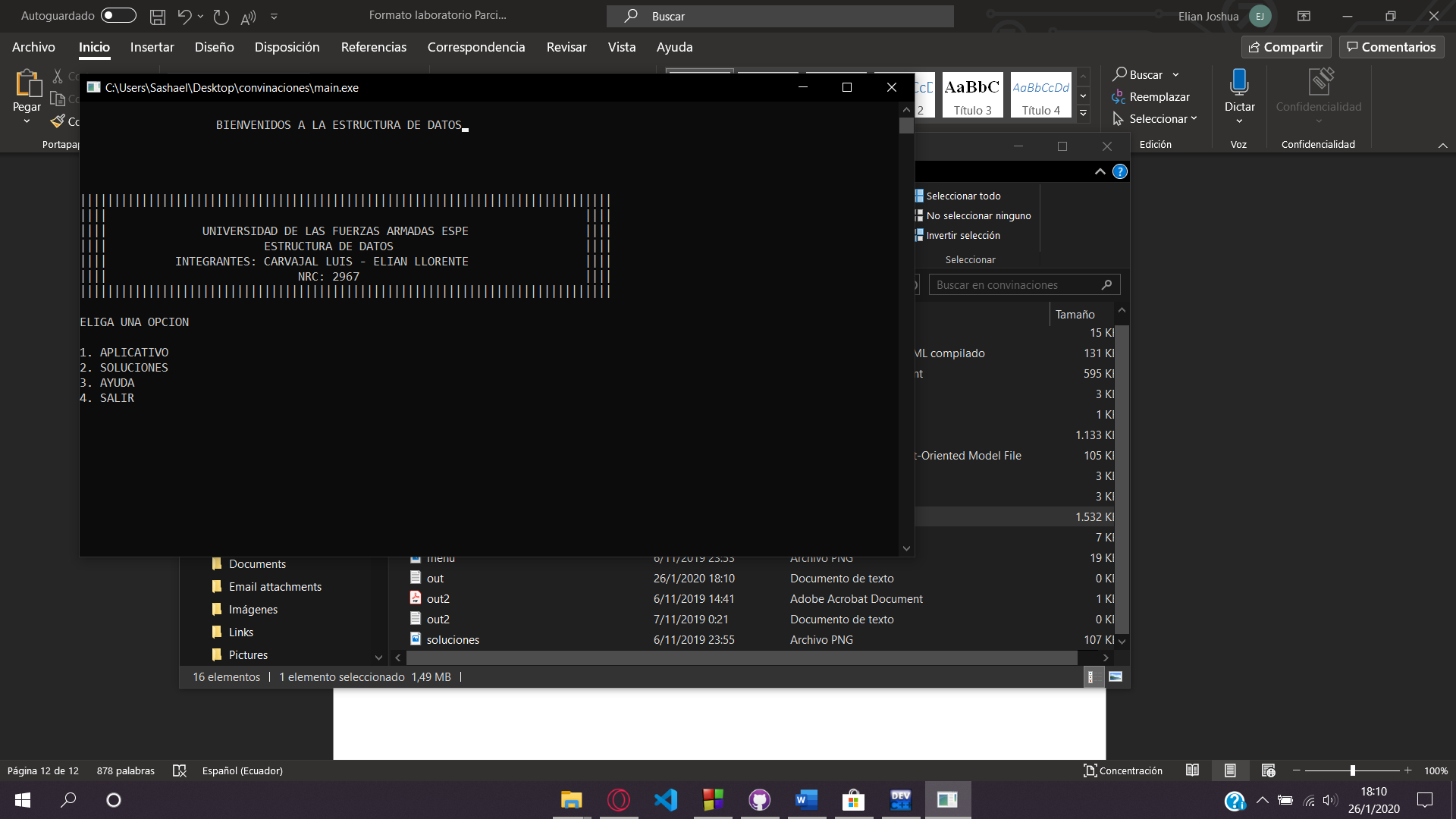
cont=39;

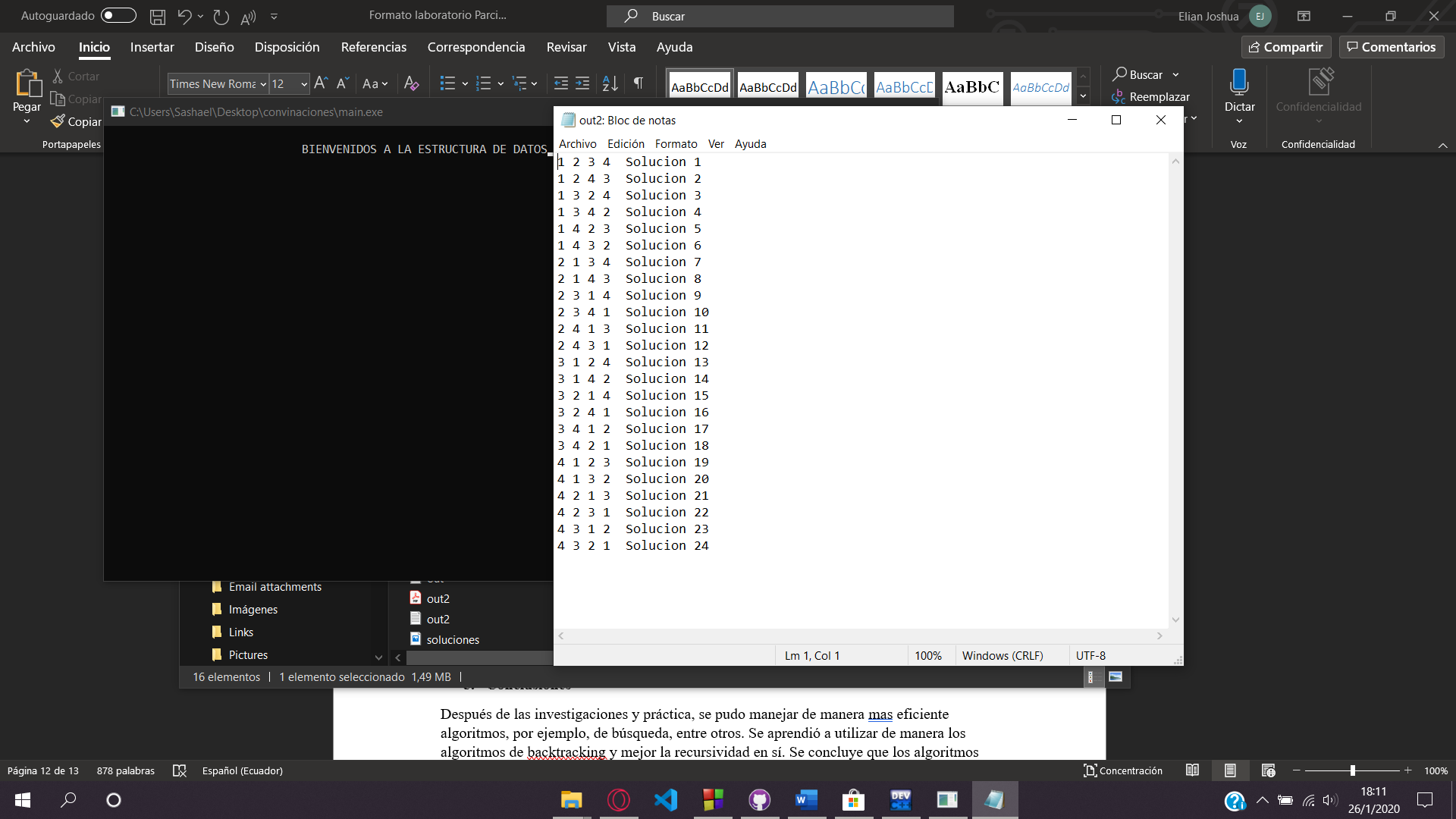
c=0;

}while (c=1);

}

1. **Ejecución**





1. **Conclusiones**

Finalmente se pudo manejar de manera más eficiente algoritmos. Se aprendió a utilizar algoritmos de backtracking y entender de mejor manera la recursividad. Se concluye que los algoritmos que usan backtracking son más rápidos que los normales, según sea el ejercicio.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

DOCENTE RESPONSABLE COORDINADOR DE ÁREA

Ing. Fernando Solis. MsC. PhD. Rodrigo Fonseca.