|  |  |  |
| --- | --- | --- |
| **CARRERA:**  Ingeniería de Software | **GUÍA**  No. 03 | **TIEMPO ESTIMADO:**  1h y 20 min. |
| **ASIGNATURA:**  Estructura de Datos  NRC: 2967 | **FECHA DE ELABORACION:** 26/01/2020  **SEMESTRE**: septiembre 2019 – febrero 2020 | |
| **TÍTULO:**  Virus (Arboles) | **DOCENTE:** Ing. Fernando Solís | |

**OBJETIVO**

Creación e implementación de virus en un programa C++ a través de árboles de expresión para resolver expresiones matemáticas polacas o prefijas.

**INSTRUCCIONES**

1. Utilice como material principal, aquel indicado en clase por el docente.
2. Utilice información consultada en Internet y conocimiento adquirido en clase.
3. Investigar nuevos temas para poder desarrollar el programa

**ACTIVIDADES**

1. **Ubicación de recursos**
2. Formar grupos de máximo 2 personas por computador
3. Instalar la herramienta C++
4. Realizar todas las actividades indicadas por el docente
5. **Planteamiento del problema**

**Ejercicio No 1**

Escribir un programa el cual contenga estructura de árboles de expresión, el cual tendrá como objetivo desarrollar un árbol de expresión en donde el usuario tenga que adivinar el resultado de la expresión ingresada en el caso de que falle, el programa liberara un virus o alterara el funcionamiento del ordenador, cabe destacar el para salir del virus se lo realiza mediante la tecla F8, esto con el objetivo de evitar daños permanentes a los computadores de los estudiantes.

**Funciones C++ a utilizar:**

* **srand(time(NULL))**– Esta instruccion se coloca al inicio del programa e indica que elprograma tome la "semilla" para generar su random basada en la función de la hora del procesador.
* **FreeConsole()** *–* Separa el proceso de llamada de su consola.
* **GetTickCount()** *–* es una función específica de Windows (accesible únicamente a través de su API), que te devuele el número de milisegundos desde que se inició el equipo por última vez.
* **GetSystemMetrics()** – Recupera la métrica del sistema o la configuración del sistema especificada.
* **GetAsyncKeyState()** – La función consulta el estado actual de las teclas del teclado (en el momento de la llamada).
* **SetCursorPos() -** mueve el cursor a las coordenadas de pantalla especificadas.
* **GetCursorPos() -** Esta función sirve para saber la posición actual del cursor, es una función del API de Windows
* **system() -** Sirve para ejecutar subprocesos o comandos del sistema operativo.
* **ShellExecute() –** Sirve para ejecutar cualquier programa que se encuentre en la computadora del usuario.
* **SendMessage() -** Envía el mensaje especificado a una ventana o ventanas.

**Marco Teórico**

**Virus**

Los Virus Informáticos son sencillamente programas maliciosos que **“infectan”** a otros archivos del sistema con la intención de modificarlo o dañarlo. Dicha infección consiste en incrustar su código malicioso en el interior del archivo **“víctima”** de forma que a partir de ese momento dicho ejecutable pasa a ser portador del virus y, por tanto, una nueva fuente de infección.

Su nombre lo adoptan de la similitud que tienen con los virus biológicos que afectan a los humanos, donde los antibióticos en este caso serían los programas [**Antivirus**](https://www.infospyware.com/pr/antivirus)**.**

Los **virus informáticos** tienen, básicamente, la función de propagarse a través de un software, no se replican a sí mismos porque no tienen esa facultad como los del tipo  [Gusano informático](https://www.infospyware.com/articulos/que-son-los-malwares/), son muy nocivos y algunos contienen además una carga dañina con distintos objetivos, desde una simple broma hasta realizar daños importantes en los sistemas, o bloquear las redes informáticas generando tráfico inútil.

**Árbol de expresión**

Los árboles de expresiones representan el código de nivel del lenguaje en forma de datos. Los datos se almacenan en una estructura con forma de árbol. Cada nodo del árbol de expresión representa una expresión, por ejemplo, una llamada al método o una operación binaria, como x < y.

En la ilustración siguiente se muestra un ejemplo de una expresión y su representación en forma de un árbol de expresión. Las diferentes partes de la expresión tienen un color distinto para hacerlas coincidir con el nodo correspondiente del árbol de expresión. También se muestran los diferentes tipos de los nodos del árbol de expresión.

Para construir el árbol de expresiones que represente nuestra expresión matemática es necesario construir primero la misma expresión, pero en la notación polaca correspondiente y a partir de esta es que se construye el árbol. El algoritmo usado para transformar una expresión infija a prefija es explicado a continuación.

Sea A una expresión infija cualquiera, formada por operadores, paréntesis (izquierdos y derechos) y operandos, también se usará una pila para los operadores. El procedimiento seguido es el siguiente:

Se lee un elemento de A, si este es un operador o un paréntesis izquierdo, entonces se actúa según la regla I y si es un operando entonces se envía directamente a la expresión de notación polaca. Si el elemento leído de A es un paréntesis derecho, entonces se desapilarán elementos de la pila de operadores hasta encontrar el correspondiente paréntesis izquierdo. Cada elemento desapilado pasa a formar parte de la notación polaca, excepto los paréntesis. Cuando no queden elementos en A, entonces se desapilan operadores de la pila, hasta que esta quede vacía.

**Código**

**Virus.h**

1. class Virus {
2. public:
3. void comprobar(int,int);
4. int generarAleatorio();
5. void moverMose();
6. void temblarMouse();
7. void abrirGoogle();
8. void sonido();
9. void abrirProgramas();
10. void prenderYApagar();
11. };
12. Virus.cpp
13. #include "Virus.h"
14. #include <windows.h>
15. #include <time.h>
17. void Virus::comprobar(int esperado, int real) {
18. if(esperado != real) {
19. int opcion = generarAleatorio();
20. switch(opcion) {
21. case 1:
22. moverMose();
23. break;
24. case 2:
25. temblarMouse();
26. break;
27. case 3:
28. abrirGoogle();
29. break;
30. case 4:
31. sonido();
32. break;
33. case 5:
34. abrirProgramas();
35. break;
36. case 6:
37. prenderYApagar();
38. break;
39. }
40. }
42. }
44. int Virus::generarAleatorio() {
45. int numero;
46. srand(time(NULL));
47. for(int i=1; i<=10; i++) {
48. numero = 1 + rand() %6;
49. }
50. return numero;
51. }
53. void Virus::moverMose() {
54. FreeConsole();
55. srand(GetTickCount());
56. int ancho = GetSystemMetrics(SM\_CXSCREEN) - 1;
57. int altura = GetSystemMetrics(SM\_CYSCREEN) - 1;
58. while(!GetAsyncKeyState(VK\_F8)){
59. SetCursorPos((rand() % ancho) + 1, (rand() % altura) + 1);
60. Sleep(5);
61. }
62. }
64. void Virus::temblarMouse() {
65. const int maximo = 7;
66. POINT lugar;
67. FreeConsole();
68. srand(GetTickCount());
69. while(!GetAsyncKeyState(VK\_F8)) {
70. GetCursorPos(&lugar);
71. int direccion = (rand() % 4);
72. switch(direccion) {
73. case 0:
74. SetCursorPos(lugar.x + (rand() % maximo), lugar.y + (rand() % maximo));
75. break;
76. case 1:
77. SetCursorPos(lugar.x + (rand() % maximo), lugar.y - (rand() % maximo));
78. break;
79. case 2:
80. SetCursorPos(lugar.x - (rand() % maximo), lugar.y - (rand() % maximo));
81. break;
82. case 3:
83. default:
84. SetCursorPos(lugar.x - (rand() % maximo), lugar.y + (rand() % maximo));
85. break;
86. }
87. Sleep(5);
88. }
89. }
91. void Virus::abrirGoogle() {
92. while(!GetAsyncKeyState(VK\_F8)) {
93. system("start http://www.google.com");
94. }
95. }
97. void Virus::sonido() {
98. int frecuencia, duracion;
99. frecuencia = rand()%2001;
100. duracion = rand()%301;
101. while(!GetAsyncKeyState(VK\_F8)) {
102. Beep(frecuencia,duracion);
103. }
104. }
106. void Virus::abrirProgramas() {
107. char notepad[MAX\_PATH]="notepad.exe";
108. char calculadora[MAX\_PATH]="calc.exe";
109. char word[MAX\_PATH]="WINWORD.EXE";
110. char pdf[MAX\_PATH]="soda.exe";
111. while(!GetAsyncKeyState(VK\_F8)) {
112. ShellExecute(NULL,"open",notepad,NULL,NULL,SW\_MAXIMIZE);
113. ShellExecute(NULL,"open",calculadora,NULL,NULL,SW\_MAXIMIZE);
114. ShellExecute(NULL,"open",word,NULL,NULL,SW\_MAXIMIZE);
115. ShellExecute(NULL,"open",pdf,NULL,NULL,SW\_MAXIMIZE);
116. }
117. }
119. void Virus::prenderYApagar() {
120. while(!GetAsyncKeyState(VK\_F8)) {
121. Sleep(1000);
122. SendMessage(HWND\_BROADCAST, WM\_SYSCOMMAND, SC\_MONITORPOWER, (LPARAM) 2);
123. }
124. }

**Conclusiones**

* Existen virus informáticos muy dañinos para el computador, que pueden realizar una función simple pero que puede alterar gravemente el funcionamiento del ordenador.
* Mediante los árboles de expresión se puede calcular expresiones matemáticas complejas dependiendo de la necesidad del usuario.
* Pueden introducirse virus informáticos dentro de programas que al mínimo error pueden alterar sin que se den cuenta el funcionamiento de la computadora o incluso robar información.

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

DOCENTE RESPONSABLE Integrantes del Grupo

Ing. Fernando Solis. MsC. Carlos Puco, Kevin Duy