**МИНОБРНАУКИ РОССИИ**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ  
ВЫСШЕГО ОБРАЗОВАНИЯ  
**«БЕЛГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ  
ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ им. В.Г.ШУХОВА»  
(БГТУ им. В.Г.Шухова)**

Кафедра программного обеспечения вычислительной техники и автоматизированных систем

Лабораторная работа №6  
дисциплина: Объектно-ориентированное программирование

тема: «**Потоки в C++»**

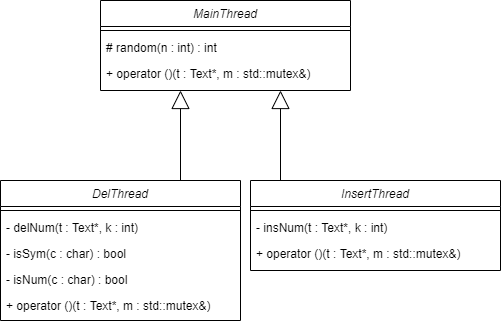
Выполнил: ст. группы ВТ-22  
Егоров Владислав Юрьевич  
Проверил: Буханов Д. Г.

Белгород 2020

**Цель работы:** Получение теоретических знаний в области разработки классов, получение практических навыков реализаций классов и отношений между ними.

**Задание 4.**

Один поток удаляет случайное число из текста, а другой поток заносит в текст символьное представление случайных чисел. Произвести синхронный вывод при каждой итерации. Показать выполнение работы программы в синхронном и асинхронном режимах.



***var4.h***

1. #ifndef VAR4\_H
2. #define VAR4\_H
3. class Text
4. **{**
5. public:
6. std::string text;
7. int nCount;
8. // осуществляет ввод текста.
9. void inputText();
10. // осуществляет подсчет чисел.
11. **void countNums();**
12. };
13. class MainThread
14. {
15. **protected:**
16. // возвращает псевдослучайное число в интервале [0..n).
17. int random(int n);
18. public:
19. virtual void operator ()(Text\*, std::mutex&) = 0;
20. };
21. class DelThread : protected MainThread
22. {
23. private:
24. // удаляет число с номером k в тексте t.
25. void delNum(Text\* t, int k);
26. // Возвращает значение «истина», если символ c не является
27. // числом, иначе – «ложь».
28. **bool isSym(char** c**);**
29. // Возвращает значение «истина», если символ c является
30. // числом, иначе – «ложь».
31. bool isNum(char c);
32. public:
33. void operator ()(Text\*, std::mutex&);
34. };
35. class InsertThread : protected MainThread
36. {
37. private:
38. // Вставляет символьное представление числа k в текст t.
39. void insNum(Text\* t, int k);
40. **public:**
41. void operator ()(Text\*, std::mutex&);
42. };
43. #endif // !VAR4\_H

***var4.cpp***

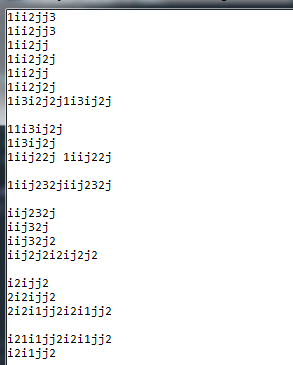
1. #include <iostream>
2. #include <thread>
3. #include <mutex>
4. #include <time.h>
5. **#include <windows.h>**
6. #include <mutex>
7. #include <random>
8. #include <string>
10. **#include "var4.h"**
12. const int SLEEP\_THREAD = 100;
13. const int SLEEP\_PRINT = 500;
15. **void Text::inputText()**
16. {
17. std::getline(std::cin, text);
19. countNums();
20. **}**
21. void Text::countNums()
22. {
23. int n = 0;
24. int len = text.length();
26. for (int i = 0; i < len; i++)
27. {
28. if (isdigit(text[i]))
29. n++;
30. **}**
32. nCount = n;
33. }
35. **int MainThread::random(int n)**
36. {
37. if (n == 0)
38. return 0;
40. **return rand() % n;**
41. }
43. void DelThread::delNum(Text\* t, int k)
44. {
45. **int i = 0;**
46. int len = (\*t).text.length();
48. while (i < len && this->isSym((\*t).text[i]))
49. i++;
51. while (k && i < len)
52. {
53. if(this->isSym((\*t).text[i]))
54. i++;
55. **else if (this->isNum((\*t).text[i]))**
56. {
57. k--;
58. if(k)
59. i++;
60. **}**
61. }
63. if(!k)
64. {
65. **(\*t).text.erase(i, 1);**
66. t->nCount--;
67. }
68. }
69. bool DelThread::isSym(char c)
70. **{**
71. return !((c >= '0') && (c <= '9'));
72. }
73. bool DelThread::isNum(char c)
74. {
75. **return ((c >= '0') && (c <= '9'));**
76. }
77. void DelThread::operator ()(Text\* t, std::mutex& m)
78. {
79. while (1)
80. **{**
81. m.lock();
83. int n = t->nCount;
84. int k = random(n) + 1;
85. **this->delNum(t, k);**
87. std::cout << t->text << std::endl;
89. m.unlock();
90. **Sleep(SLEEP\_THREAD);**
91. }
92. }
94. void InsertThread::insNum(Text\* t, int k)
95. **{**
96. int len = (\*t).text.length();
97. int randPlace = this->random(len);
98. std::string sn = std::to\_string(k);
100. **(\*t).text.insert(randPlace, sn);**
101. t->countNums();
102. }
103. void InsertThread::operator()(Text\* t, std::mutex& m)
104. {
105. **while (1)**
106. {
107. m.lock();
109. int n = t->nCount;
110. **int k = random(n) + 1;**
111. this->insNum(t, k);
113. std::cout << t->text << std::endl;
115. **m.unlock();**
116. Sleep(SLEEP\_THREAD);
117. }
118. }

***main.cpp***

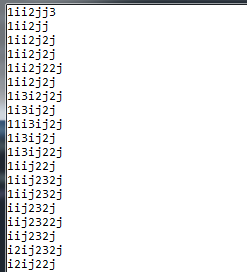
1. #include <iostream>
2. #include <thread>
3. #include <time.h>
4. #include <windows.h>
5. **#include <mutex>**
6. #include <random>
7. #include <string>
8. #include <conio.h>
10. **#include "var4.h"**
12. void out(Text\* t, std::mutex& m)
13. {
14. while (1)
15. **{**
16. m.lock();
18. std::cout << t->text << std::endl;
20. **m.unlock();**
22. Sleep(500);
23. }
24. }
26. int main()
27. {
28. Text t;
29. t.inputText();
31. DelThread del;
32. InsertThread ins;
34. std::mutex m;
36. std::thread t1(del, &t, std::ref(m));
37. std::thread t2(ins, &t, std::ref(m));
38. std::thread t3(out, &t, std::ref(m));
40. **t1.join();**
41. t2.join();
42. t3.join();
44. system("pause");
45. **return 0;**
46. }

***Результаты (строка “1ii2jj3”)***

1. Синхронный режим



1. Асинхронный режим



***Задание 7.*** Один поток создает бегущую строку (из случайных символов), а другой вставляет или удаляет между ними случайные знаки препинания. Произвести синхронный вывод при каждой итерации. Показать выполнение работы программы в синхронном и асинхронном режимах.

***var7.cpp***

1. #include <iostream>
2. #include <thread>
3. #include <time.h>
4. #include <windows.h>
5. **#include <mutex>**
6. #include <random>
7. #include <string>
8. #include <tchar.h>
10. **int nSymbs = 9;**
11. char symbols[] = { '.', ':', ';', '!', '?', '"', '-', '(', ')' };
12. **int** curr = 0;
14. // Возвращает значение "истина", если с - знак препинания,
15. // иначе - "ложь".
16. **int isSign(char c);**
17. // Вставляет случайный знак препинания в строку s.
18. void insertRandSymb(std::string\*);
19. // Удаляет случайный знак препинания из строки s.
20. void delRandSymb(std::string\*);
22. // Поток бегущей строки.
23. DWORD WINAPI threadRun(LPVOID);
24. // Поток вставки или удаления знаков препинания.
25. DWORD WINAPI threadChars(LPVOID);
27. int main()
28. {
29. std::string str = "Source string";
31. **HANDLE thread1 = CreateThread(NULL, 0, threadRun, &str, 0, NULL);**
32. HANDLE thread2 = CreateThread(NULL, 0, threadChars, &str, 0, NULL);
34. WaitForSingleObject(thread1, INFINITE);
35. WaitForSingleObject(thread2, INFINITE);
37. CloseHandle(thread1);
38. CloseHandle(thread2);
40. return 0;
41. **}**
43. int isSign(char c)
44. {
45. int i = 0;
47. while (i < nSymbs && symbols[i] != c)
48. i++;
50. return i < nSymbs;
51. **}**
52. void insertRandSymb(std::string\* strPtr)
53. {
54. strPtr->insert(rand() % strPtr->length(), 1, symbols[rand() % nSymbs]);
55. curr++;
56. }
57. **void delRandSymb(std::string\* strPtr)**
58. {
59. **if** (curr < 3)
60. return;
62. int rIndex = rand() % strPtr->length();
63. int f = 1;
65. **while (!isSign(strPtr->at(rIndex)))**
66. rIndex = rand() % strPtr->length();
68. strPtr->erase(rIndex, 1);
69. curr--;
70. }
72. DWORD WINAPI threadRun(LPVOID str)
73. {
74. short a = 0;
75. short b = 0;
76. **HANDLE hConsole = GetStdHandle(STD\_OUTPUT\_HANDLE);**
77. system("cls");
78. std::string\* strPtr = ((std::string\*)str);
79. std::string s = "";
80. COORD x = { a, b };
81. **SetConsoleCursorPosition(hConsole, x);**
82. int bufW = 100;
84. while (1)
85. {
86. strPtr->push\_back((char)(rand() % 25 + 47));
87. **for (a = 0; a < bufW; a++)**
88. {
89. HANDLE m = OpenMutex(SYNCHRONIZE, FALSE, \_T("mutex"));
91. system("cls");
92. **x.X = a;**
93. int len = strPtr->length();
94. SetConsoleCursorPosition(hConsole, x);
95. if (len + a > bufW)
96. {
97. **s.clear();**
98. for (int i = 0; i < bufW - a; i++)
99. s.push\_back(strPtr->at(i));
100. std::cout << s;
101. s.clear();
102. **SetConsoleCursorPosition(hConsole, COORD{ 0, b });**
103. for (int i = bufW - a; i < len; i++)
104. s.push\_back(strPtr->at(i));
105. std::cout << s;
106. SetConsoleCursorPosition(hConsole, x);
107. **}**
108. else
109. std::cout << \*strPtr;
111. ReleaseMutex(m);
112. **Sleep(50);**
113. }
114. }
115. return 0;
116. }
117. **DWORD WINAPI threadChars(LPVOID str)**
118. {
119. srand(time(NULL));
120. while (1)
121. {
122. **HANDLE m = OpenMutex(SYNCHRONIZE, FALSE, \_T("mutex"));**
124. if (rand() % 2)
125. insertRandSymb((std::string\*)str);
126. else
127. **delRandSymb((std::string\*)str);**
129. ReleaseMutex(m);
130. Sleep(100);
131. }
132. **return 0;**
133. }

***Результаты (исходная строка «Source string»)***

