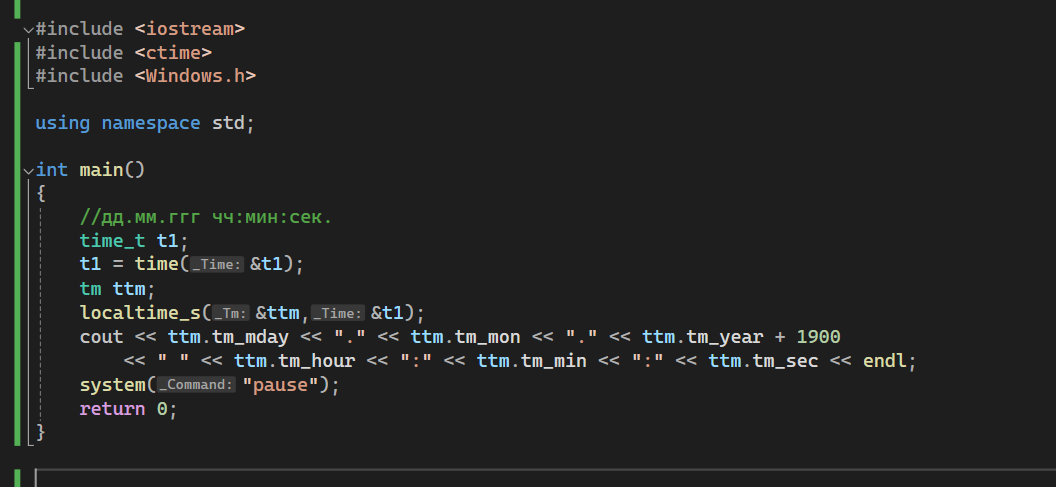
Лабораторная работа 07

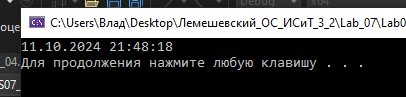
Компьютерное время

OC, ПОИТ-3

**Задание 01. Windows**

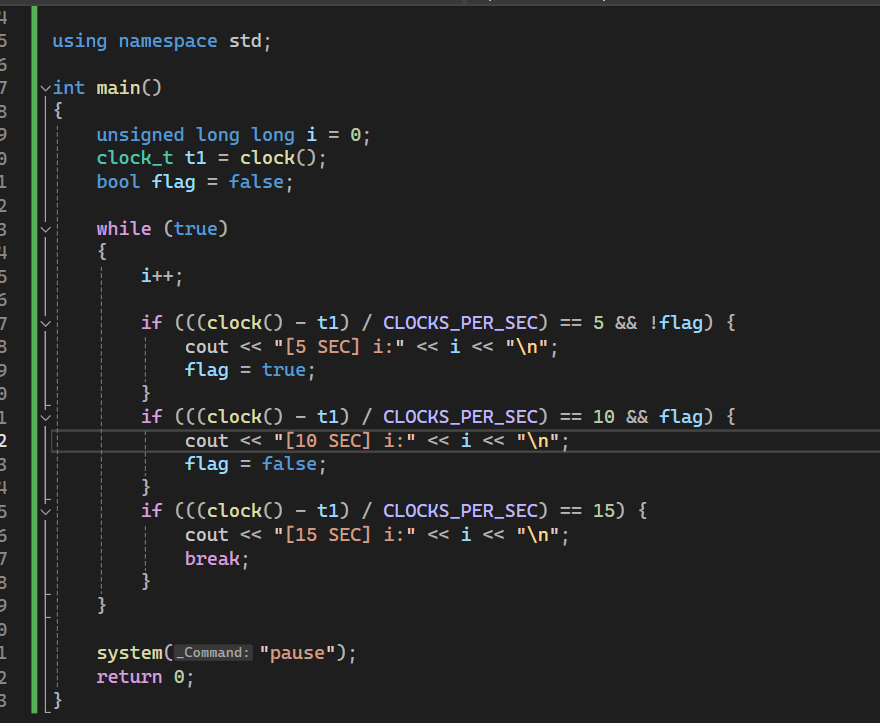
1. Разработайте приложение **OS07\_01**.
2. Приложение **OS07\_01** выводит на консоль текущую локальную дату и время в формате ***дд.мм.гггг чч:мин:сек***.

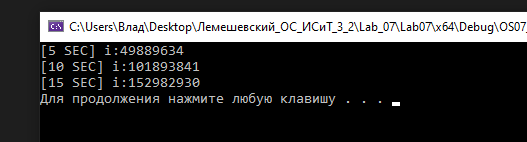
****

****

**Задание 02. Windows**

1. Разработайте приложение **OS07\_02,** выполняющее бесконечный цикл.
2. В теле цикла подсчитывается количество итераций.
3. Выведите на консоль значения счетчика итераций через 5 сек. и 10 сек.
4. Корректно завершите работу цикла и приложения через 15 сек., выведите итоговое значение счетчика итераций.

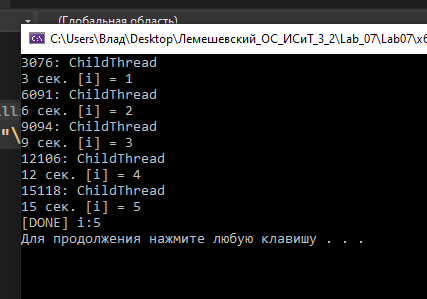
****

****

**Задание 03. Windows**

1. **Указание: самостоятельно освойте и примените периодический ожидающий таймер**
2. Разработайте приложение **OS07\_03,** выполняющее бесконечный цикл.
3. В теле цикла с задержкой подсчитывается количество итераций.
4. Выведите на консоль значения счетчика итераций каждые 3 сек.
5. Корректно завершите работу цикла и приложения через 15 сек., выведите итоговое значение счетчика итераций.

|  |
| --- |
| 1. #include <Windows.h> 2. #include <iostream> 3. #include <ctime> 4. #define SECOND 10000000 5. using namespace std; 6. HANDLE htimer = NULL; 7. DWORD WINAPI ChildThread() 8. { 9. WaitForSingleObject(htimer, INFINITE); 10. cout << clock() << ": ChildThread" << "\n"; 11. return 0; 12. } 13. int main() 14. { 15. SetConsoleCP(1251); 16. SetConsoleOutputCP(1251); 17. htimer = CreateWaitableTimer(NULL, FALSE, NULL); 18. long long it = -3 \* SECOND; 19. unsigned long long i = 0; 20. short n = 0; 21. clock\_t t1 = clock(); 22. while (true) 23. { 24. i++; 25. if (!SetWaitableTimer(htimer, (LARGE\_INTEGER\*)&it, 0, NULL, NULL, FALSE)) throw "[ERROR]: SetWaitableTimer"; 26. DWORD ChildId = 0; 27. HANDLE hChild = CreateThread(NULL, 0, (LPTHREAD\_START\_ROUTINE)ChildThread, NULL, NULL, &ChildId); 28. WaitForSingleObject(hChild, INFINITE); 29. cout << ((clock() - t1) / CLOCKS\_PER\_SEC) << " сек. [i] = " << i << "\n"; 30. if (((clock() - t1) / CLOCKS\_PER\_SEC) == 15) { 31. cout << "[DONE] i:" << i << "\n"; 32. break; 33. } 34. } 35. system("pause"); 36. return 0; 37. } |

****

**Задание 04. Windows**

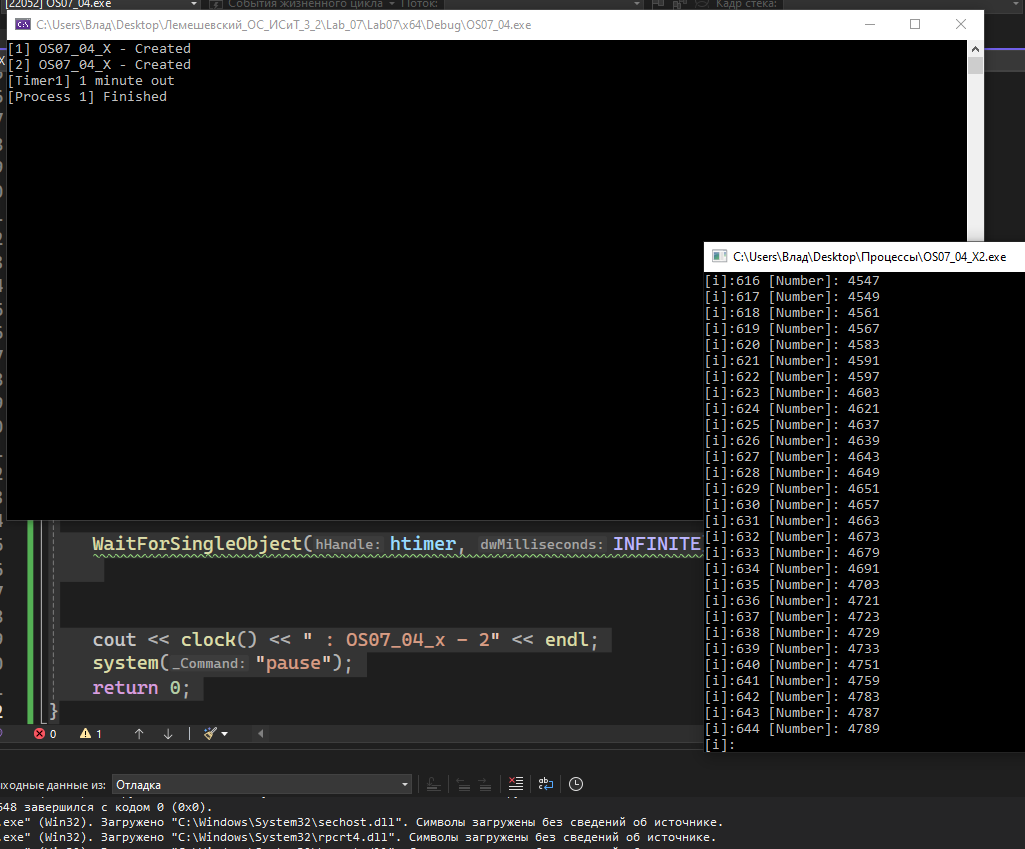
1. Разработайте приложение **OS07\_04,** запускающее два одинаковых дочерних процесса **OS07\_04\_X**.

|  |
| --- |
| 1. #include <Windows.h> 2. #include <iostream> 3. #include <ctime> 4. #define SECOND 10000000 5. using namespace std; 6. int main() { 7. long long it1 = -60 \* SECOND; 8. long long it2 = -120 \* SECOND; 9. HANDLE hTimer1 = CreateWaitableTimer(NULL, FALSE, L"vladTimer1"); 10. if (!hTimer1) { 11. cerr << "Failed to create Timer1.\n"; 12. return 1; 13. } 14. if (!SetWaitableTimer(hTimer1, (LARGE\_INTEGER\*)&it1, 0, NULL, NULL, FALSE)) { 15. cerr << "Failed to set Timer1.\n"; 16. return 1; 17. } 18. HANDLE hTimer2 = CreateWaitableTimer(NULL, FALSE, L"vladTimer2"); 19. if (!hTimer2) { 20. cerr << "Failed to create Timer2.\n"; 21. return 1; 22. } 23. if (!SetWaitableTimer(hTimer2, (LARGE\_INTEGER\*)&it2, 0, NULL, NULL, FALSE)) { 24. cerr << "Failed to set Timer2.\n"; 25. return 1; 26. } 27. LPCWSTR applicationName1 = L"C:\\Users\\Влад\\Desktop\\Процессы\\OS07\_04\_X.exe"; 28. LPCWSTR applicationName2 = L"C:\\Users\\Влад\\Desktop\\Процессы\\OS07\_04\_X2.exe"; 29. STARTUPINFO si; 30. PROCESS\_INFORMATION pi1, pi2; 31. ZeroMemory(&si, sizeof(STARTUPINFO)); 32. si.cb = sizeof(STARTUPINFO); 33. if (CreateProcess(applicationName1, NULL, NULL, NULL, FALSE, CREATE\_NEW\_CONSOLE, NULL, NULL, &si, &pi1)) { 34. cout << "[1] OS07\_04\_X - Created\n"; 35. } 36. else { 37. cerr << "[1] OS07\_04\_X - Not Created\n"; 38. return 1; 39. } 40. if (CreateProcess(applicationName2, NULL, NULL, NULL, FALSE, CREATE\_NEW\_CONSOLE, NULL, NULL, &si, &pi2)) { 41. cout << "[2] OS07\_04\_X - Created\n"; 42. } 43. else { 44. cerr << "[2] OS07\_04\_X - Not Created\n"; 45. return 1; 46. } 47. WaitForSingleObject(hTimer1, INFINITE); 48. cout << "[Timer1] 1 minute out\n"; 49. TerminateProcess(pi1.hProcess, 0); 50. CloseHandle(hTimer1); 51. WaitForSingleObject(pi1.hProcess, INFINITE); 52. cout << "[Process 1] Finished\n"; 53. WaitForSingleObject(hTimer2, INFINITE); 54. cout << "[Timer2] 2 minutes out\n"; 55. TerminateProcess(pi2.hProcess, 0); 56. CloseHandle(hTimer2); 57. WaitForSingleObject(pi2.hProcess, INFINITE); 58. cout << "[Process 2] Finished\n"; 59. CloseHandle(pi1.hProcess); 60. CloseHandle(pi1.hThread); 61. CloseHandle(pi2.hProcess); 62. CloseHandle(pi2.hThread); 63. cout << "Main process completed.\n"; 64. system("pause"); 65. return 0; 66. } |

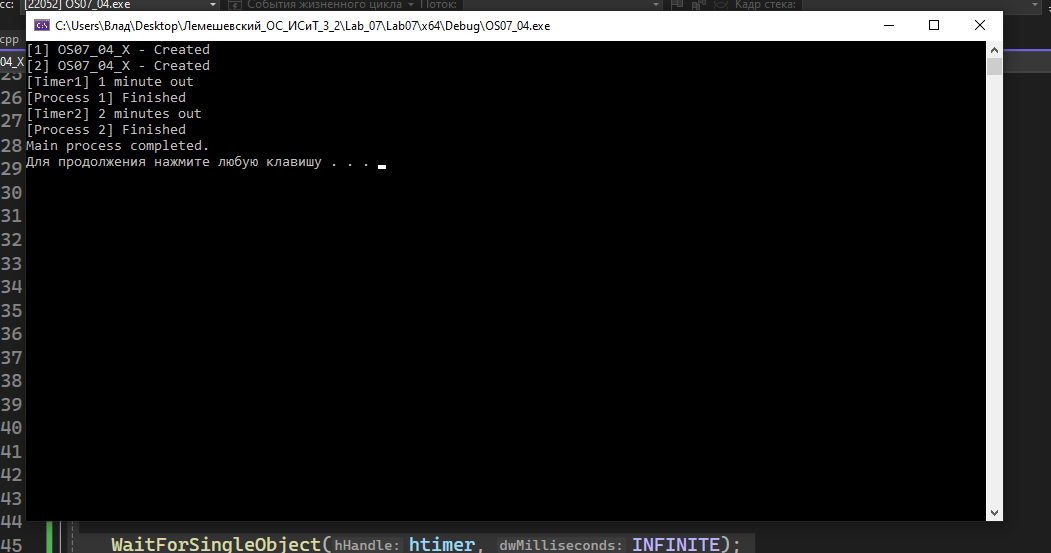
1. Процессы **OS07\_04\_X** вычисляют и выводят на консоль (каждый в свою) пронумерованный ряд простых положительных чисел (простое число делится нацело только на себя и 1).

|  |
| --- |
| 1. #include <Windows.h> 2. #include <iostream> 3. #include <ctime> 4. using namespace std; 5. bool flag = true; 6. bool isSimple(int n) { 7. if (n <= 1) return false; 8. if (n <= 3) return true; 9. if (n % 2 == 0 || n % 3 == 0) return false; 10. for (int i = 5; i \* i <= n; i += 6) { 11. if (n % i == 0 || n % (i + 2) == 0) return false; 12. } 13. return true; 14. } 15. void printNumberedPrimes() { 16. int number = 2; 17. int i = 1; 18. while (true) { 19. if (isSimple(number)) { 20. cout << "[i]:" << i << " [Number]: " << number << std::endl; 21. i++; 22. Sleep(100); 23. } 24. number++; 25. } 26. } 27. int main(){ 28. cout << clock() << " : OS07\_04\_x - 1" << endl; 29. HANDLE htimer = OpenWaitableTimer(TIMER\_ALL\_ACCESS,FALSE,L"vladTimer2"); 30. if (!htimer) 31. cout << "Timer not opened!\n"; 32. printNumberedPrimes(); 33. WaitForSingleObject(htimer, INFINITE); 35. cout << clock() << " : OS07\_04\_x - 2" << endl; 36. exit(0); 37. system("pause"); 38. return 0; 39. } |
|  |

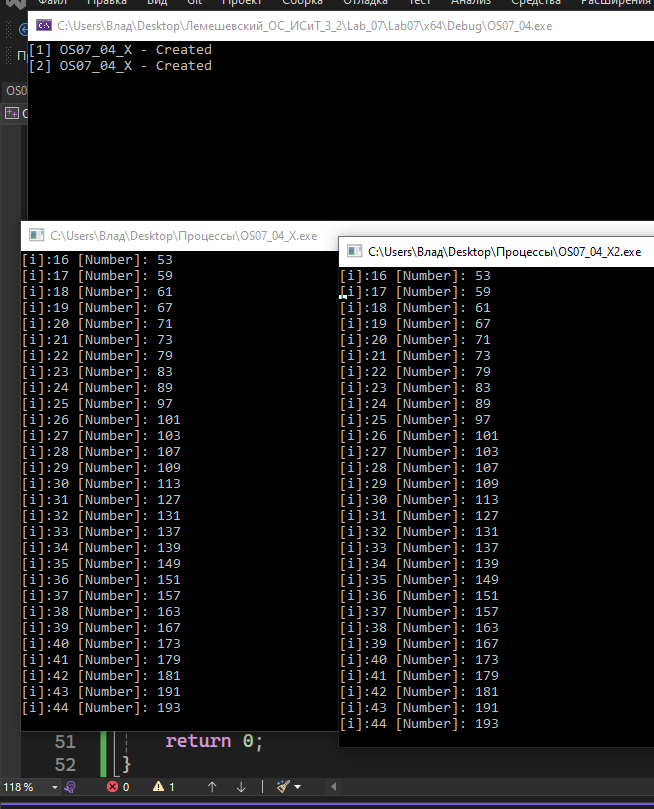
1. Первый дочерний процесс должен выполняться 1 минуту и корректно завершаться.



1. Первый дочерний процесс должен выполняться 2 минуты и корректно завершаться.

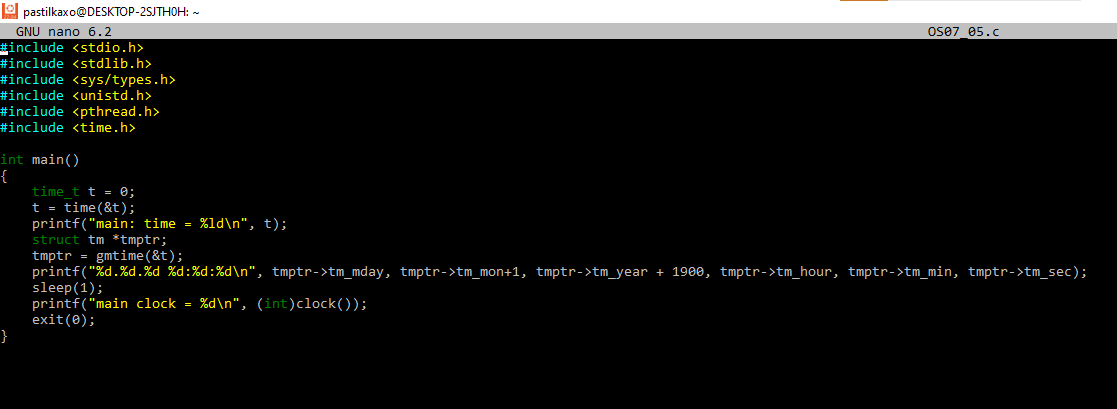


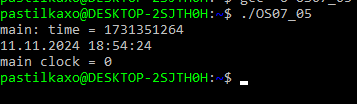
1. Приложение **OS07\_04** завершается после завершения дочерних процессов.

****

**Задание 05. Linux**

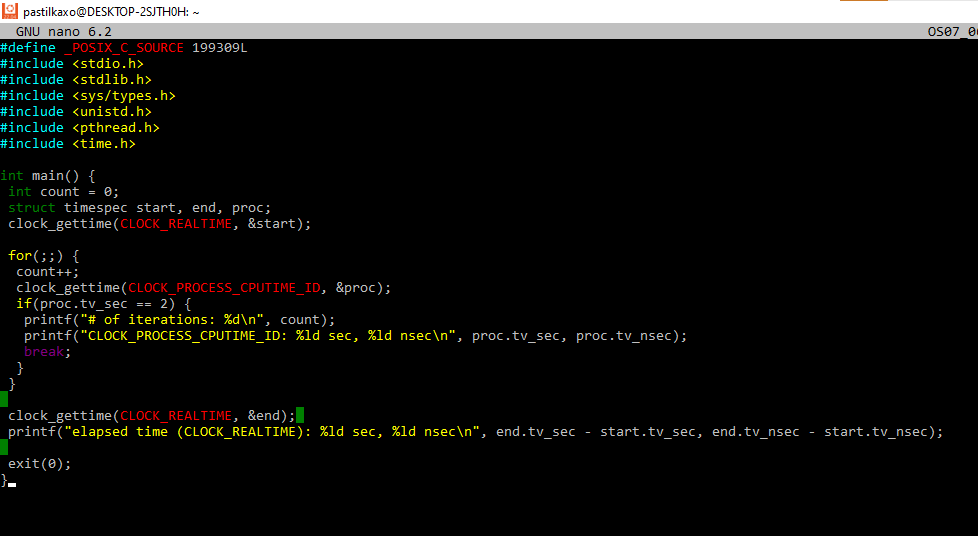
1. Разработайте приложение **OS07\_05**.
2. Приложение **OS07\_05** выводит на консоль текущую локальную дату и время в формате ***дд.мм.ггг чч:мин:сек***.

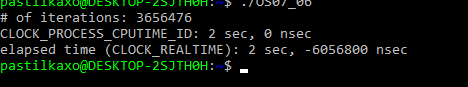
****

****

**Задание 06. Linux**

1. Разработайте приложение **OS07\_06,** выполняющее бесконечный цикл.
2. В теле цикла подсчитывается количество итераций.
3. Выведите на консоль значения счетчика итераций через 2 сек. **процессорного** времени и корректно завершите цикл.
4. Приложение **OS07\_06 должно** выполнять замер реального затраченного на работу цикла времени и выводить его значения на консоль.

****

****

**Задание 07.** Ответьте на следующие вопросы

1. Поясните понятие «социальное время» и почему оно не монотонное?
2. Поясните понятие «эпоха Linux», назовите стартовую дату «эпохи Linux» и в каких единицах изменяется время?
3. Поясните понятие «Universal Coordinated Time (UCT)».
4. Поясните понятия «относительное время» и «абсолютное время».
5. Поясните понятие «тик».
6. Поясните понятие «ожидающий таймер», перечислите типы таймеров, перечислите состояния, в которых может находится таймер.
7. Перечислите типы часов, используемых в Linux, поясните их назначение.
8. Поясните назначение констант HZ, CLOCKS\_PER\_SEC.