Герман Александр, ФИТ-4

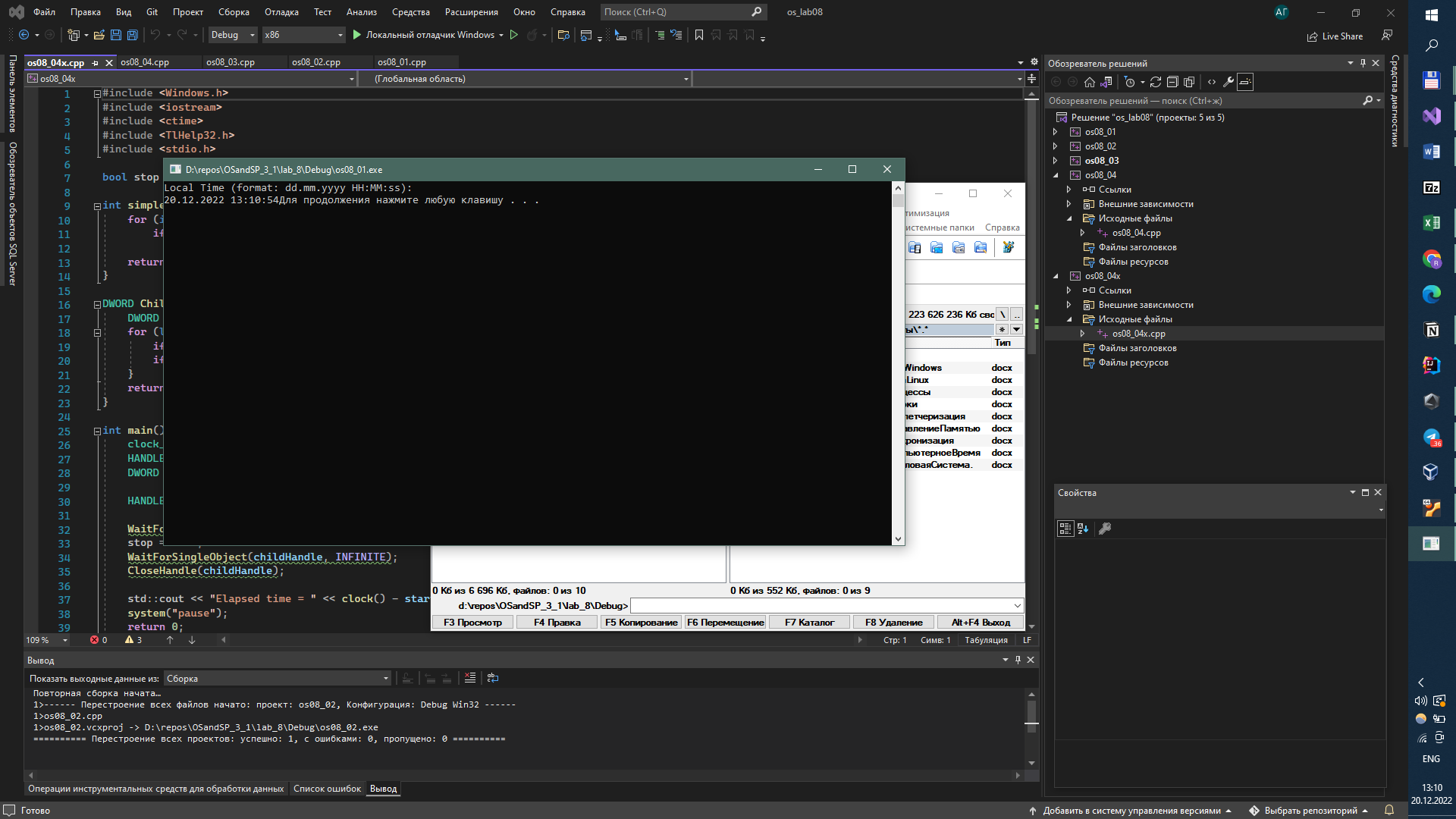
**Лабораторная работа №8**

«Компьютерное время»

**Задание 01. Windows**

1. Разработайте приложение **OS08\_01**.
2. Приложение **OS08\_01** выводит на консоль текущую локальную дату и время в формате ***дд.мм.ггг чч:мин:сек***.

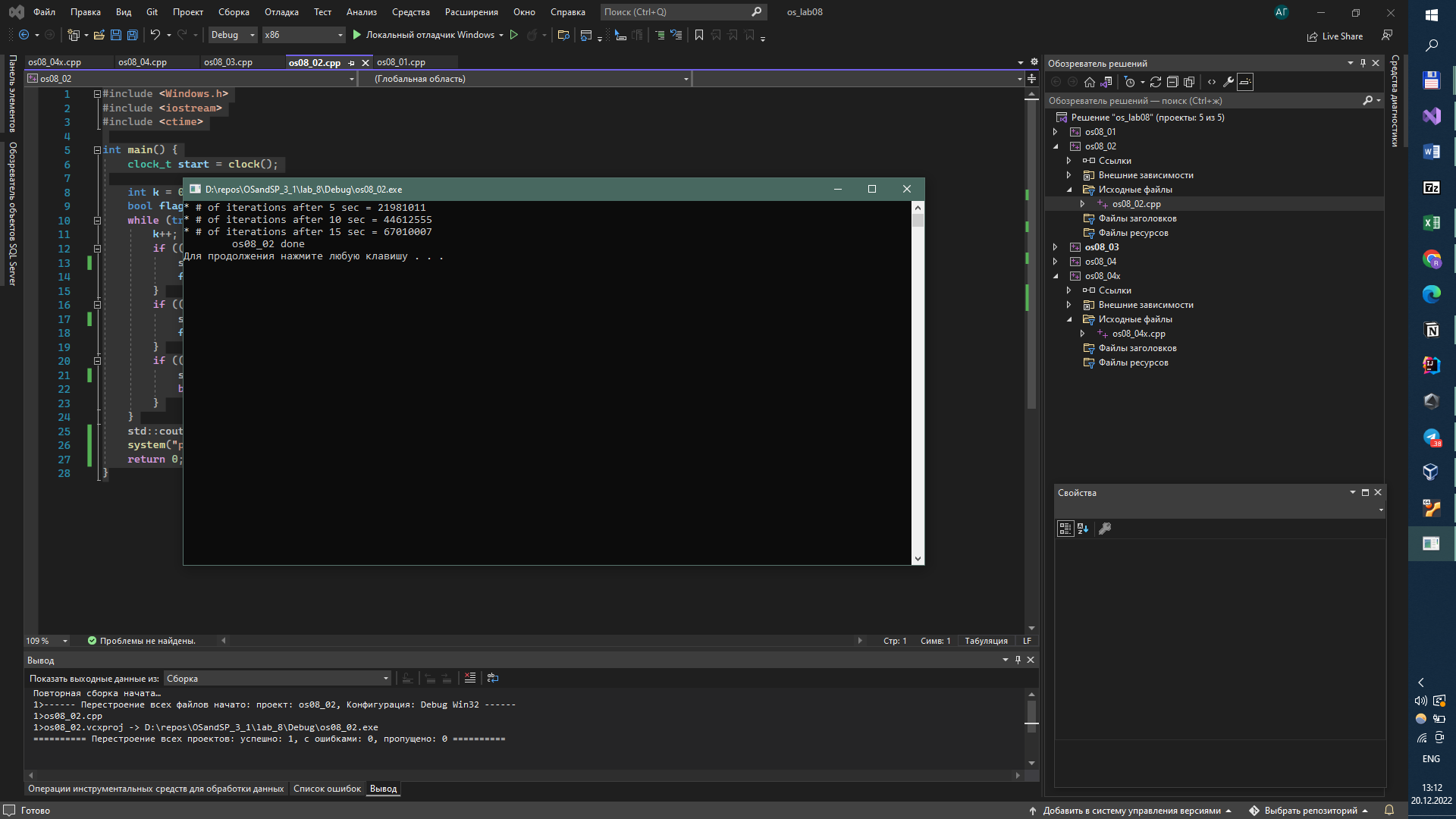
|  |
| --- |
| #include <Windows.h>  #include <iostream>  #include <ctime>  int main() {  time\_t t1;  t1 = time(&t1);  tm ttm;  localtime\_s(&ttm, &t1);  std::cout << "Local Time (format: dd.mm.yyyy HH:MM:ss): \n";  std::cout << ttm.tm\_mday << '.'  << ttm.tm\_mon + 1 << '.'  << ttm.tm\_year + 1900 << ' '  << ttm.tm\_hour << ':'  << ttm.tm\_min << ':'  << ttm.tm\_sec;  } |



**Задание 02. Windows**

1. Разработайте приложение **OS08\_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.

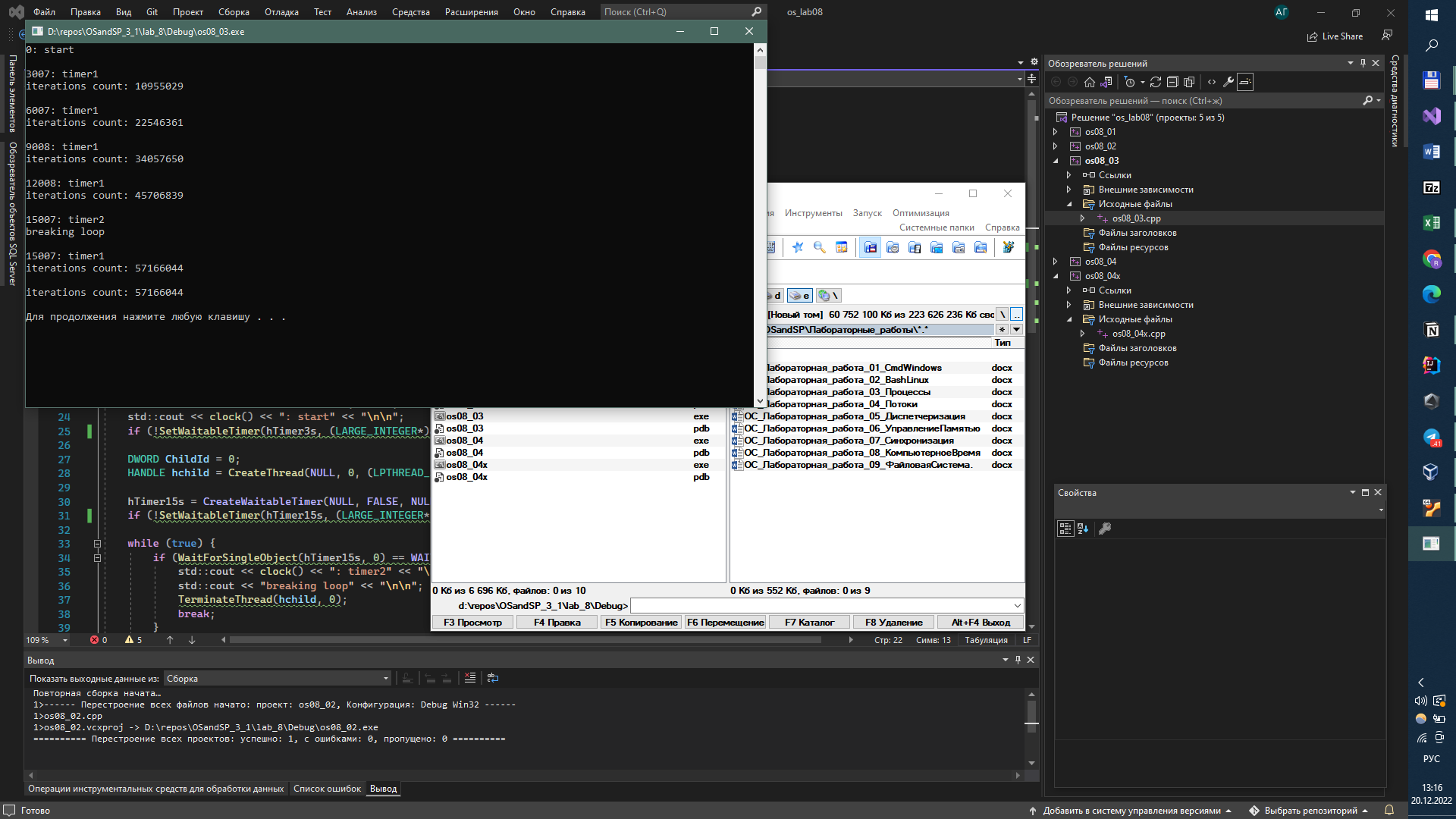
|  |
| --- |
| #include <Windows.h>  #include <iostream>  #include <ctime>  int main() {  clock\_t start = clock();  int k = 0;  bool flag5 = true, flag10 = true;  while (true) {  k++;  if ((clock() - start) / CLOCKS\_PER\_SEC == 5 && flag5) {  std::cout << "\* # of iterations after 5 sec = " << k << '\n';  flag5 = false;  }  if ((clock() - start) / CLOCKS\_PER\_SEC == 10 && flag10) {  std::cout << "\* # of iterations after 10 sec = " << k << '\n';  flag10 = false;  }  if ((clock() - start) / CLOCKS\_PER\_SEC == 15) {  std::cout << "\* # of iterations after 15 sec = " << k << '\n';  break;  }  }  std::cout << "\tos08\_02 done\n";  system("pause");  return 0;  } |



**Задание 03. Windows**

1. **Указание: самостоятельно освойте и примените периодический ожидающий таймер**
2. Разработайте приложение **OS08\_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.

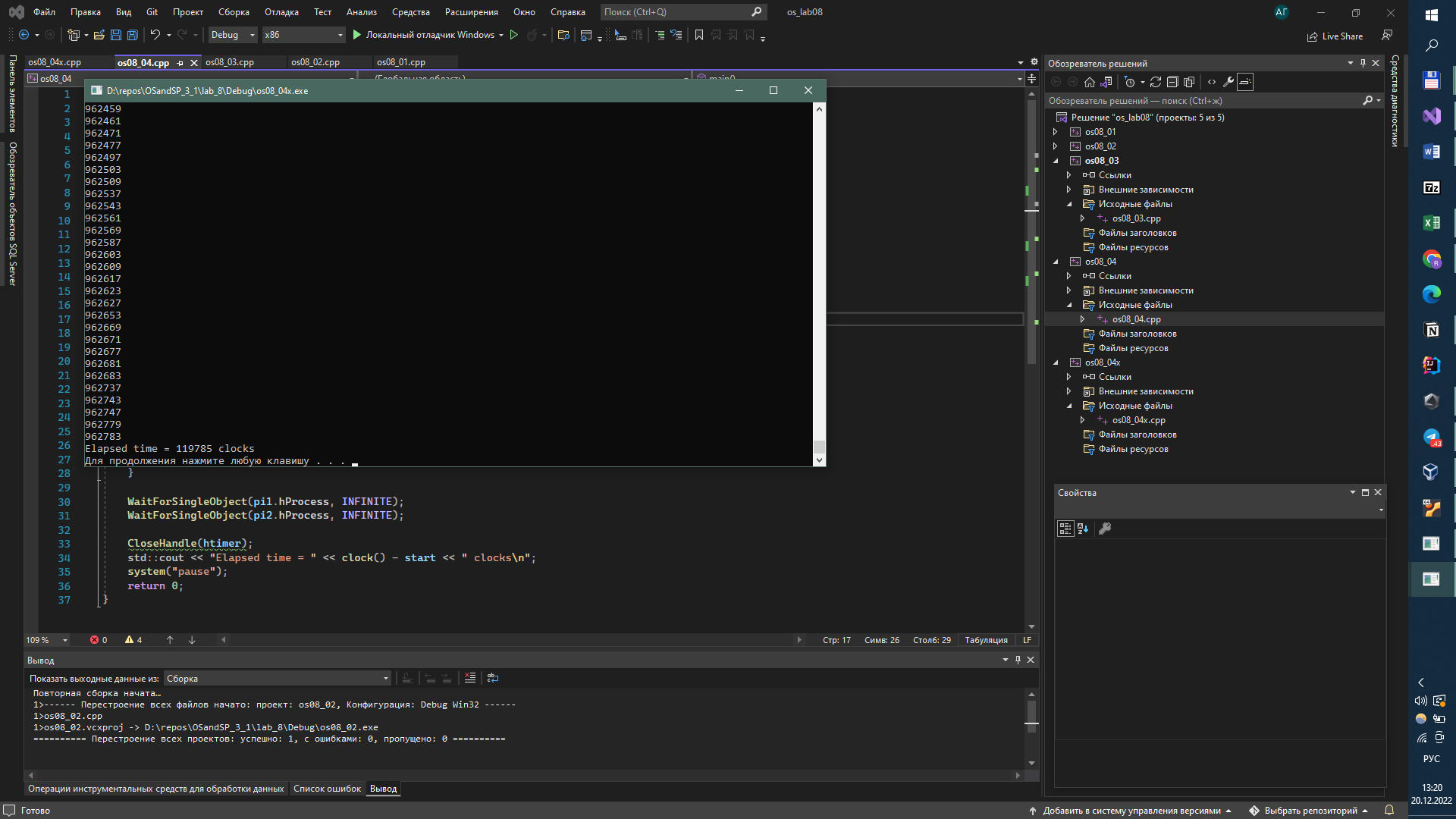
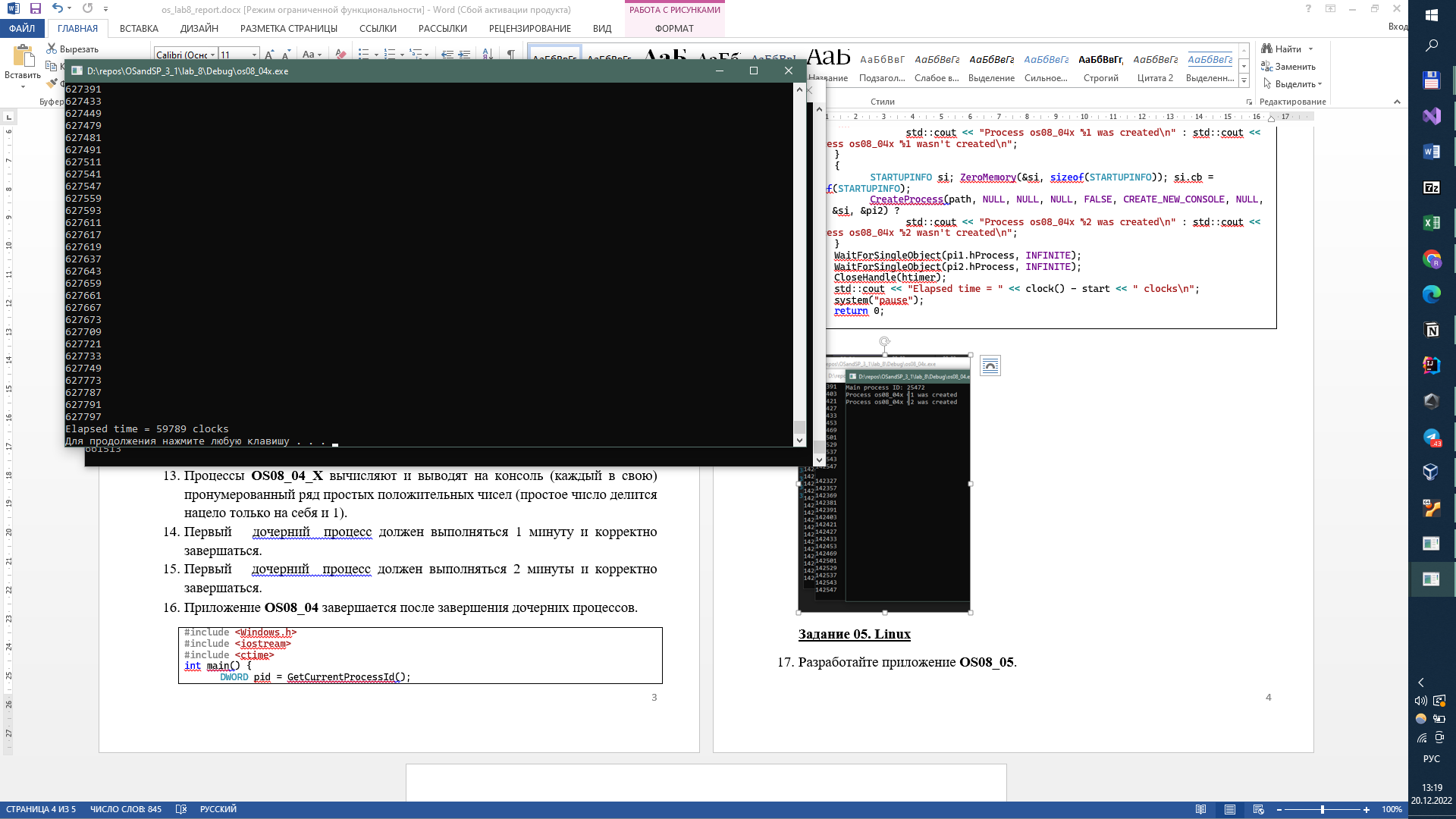
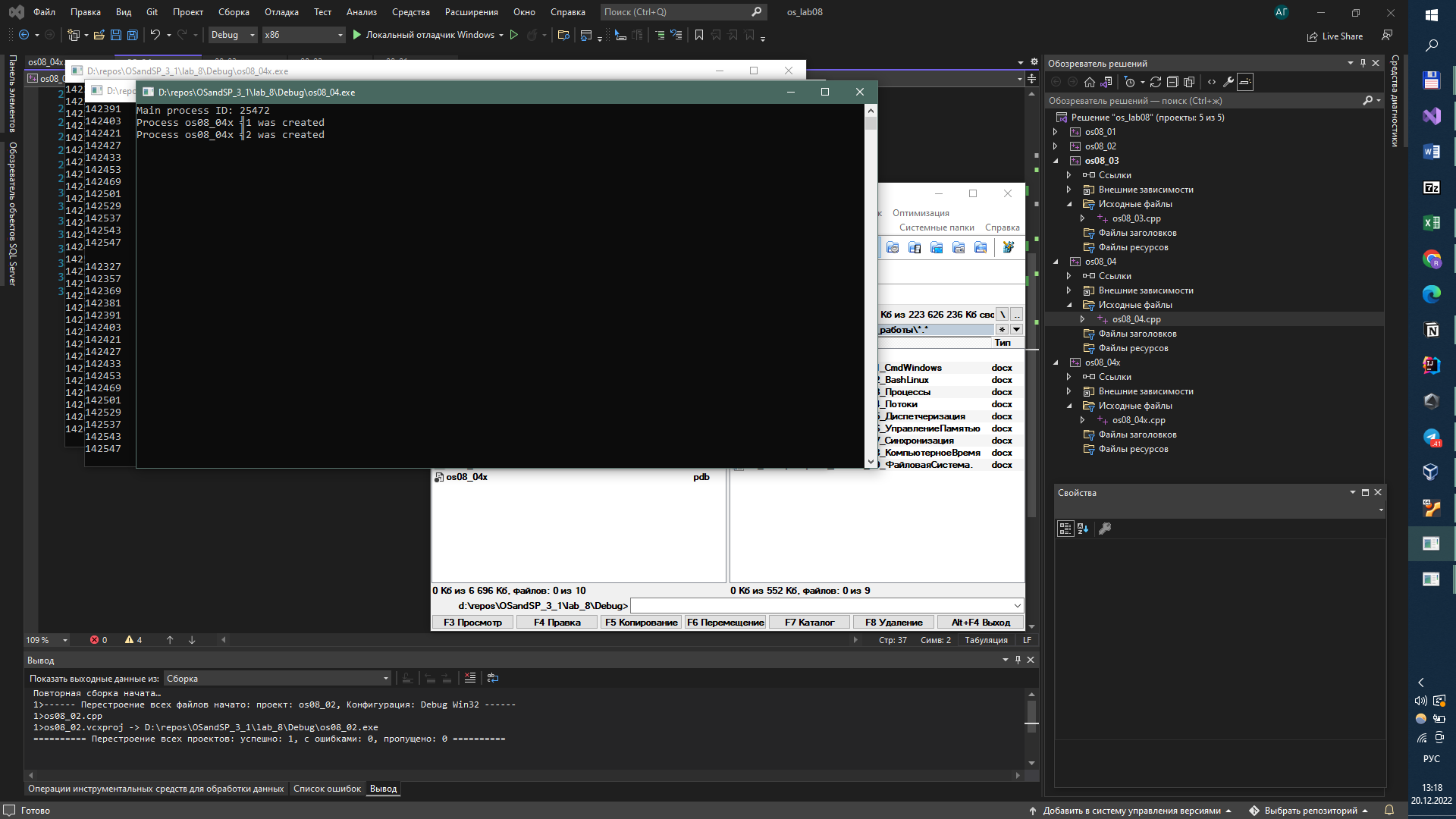
|  |
| --- |
| #include <Windows.h>  #include <iostream>  #include <ctime>  #define SECOND 10000000  HANDLE hTimer3s = NULL;  long long time3s = -3 \* SECOND;  HANDLE hTimer15s = NULL;  long long time15s = -15 \* SECOND;  int i = 0;  DWORD WINAPI ChildThread() {  while (true) {  WaitForSingleObject(hTimer3s, INFINITE);  std::cout << clock() << ": timer1" << "\n";  std::cout << "iterations count: " << i << "\n\n";  }  }  int main() {  hTimer3s = CreateWaitableTimer(NULL, FALSE, NULL);  std::cout << clock() << ": start" << "\n\n";  if (!SetWaitableTimer(hTimer3s, (LARGE\_INTEGER\*)&time3s, 3000, NULL, NULL, FALSE)) throw "SetwaitableTimer (3sec) error";  DWORD ChildId = 0;  HANDLE hchild = CreateThread(NULL, 0, (LPTHREAD\_START\_ROUTINE)ChildThread, NULL, NULL, &ChildId);  hTimer15s = CreateWaitableTimer(NULL, FALSE, NULL);  if (!SetWaitableTimer(hTimer15s, (LARGE\_INTEGER\*)&time15s, 0, NULL, NULL, FALSE)) throw "SetwaitableTimer (15sec) error";  while (true) {  if (WaitForSingleObject(hTimer15s, 0) == WAIT\_OBJECT\_0) {  std::cout << clock() << ": timer2" << "\n";  std::cout << "breaking loop" << "\n\n";  TerminateThread(hchild, 0);  break;  }  i++;  }  std::cout << "iterations count: " << i << "\n\n";  system("pause");  return 0;  } |



**Задание 04. Windows**

1. Разработайте приложение **OS08\_04,** запускающее два одинаковых дочерних процесса **OS08\_04\_X**.
2. Процессы **OS08\_04\_X** вычисляют и выводят на консоль (каждый в свою) пронумерованный ряд простых положительных чисел (простое число делится нацело только на себя и 1).
3. Первый дочерний процесс должен выполняться 1 минуту и корректно завершаться.
4. Первый дочерний процесс должен выполняться 2 минуты и корректно завершаться.
5. Приложение **OS08\_04** завершается после завершения дочерних процессов.

|  |
| --- |
| #include <Windows.h>  #include <iostream>  #include <ctime>  int main() {  DWORD pid = GetCurrentProcessId();  std::cout << "Main process ID: " << pid << '\n';  long long it = -60 \* 10000000;  HANDLE htimer = CreateWaitableTimer(NULL, FALSE, L"os08\_04");  if (!SetWaitableTimer(htimer, (LARGE\_INTEGER\*)&it, 60000, NULL, NULL, FALSE))  throw "SetWaitableTimer Error";  LPCWSTR path = L"D:\\repos\\OSandSP\_3\_1\\lab\_8\\Debug\\os08\_04x.exe";  PROCESS\_INFORMATION pi1, pi2;  clock\_t start = clock();  pi1.dwThreadId = 1; pi2.dwThreadId = 2;  {  STARTUPINFO si; ZeroMemory(&si, sizeof(STARTUPINFO)); si.cb = sizeof(STARTUPINFO);  CreateProcess(path, NULL, NULL, NULL, FALSE, CREATE\_NEW\_CONSOLE, NULL, NULL, &si, &pi1) ?  std::cout << "Process os08\_04x №1 was created\n" : std::cout << "Process os08\_04x №1 wasn't created\n";  }  {  STARTUPINFO si; ZeroMemory(&si, sizeof(STARTUPINFO)); si.cb = sizeof(STARTUPINFO);  CreateProcess(path, NULL, NULL, NULL, FALSE, CREATE\_NEW\_CONSOLE, NULL, NULL, &si, &pi2) ?  std::cout << "Process os08\_04x №2 was created\n" : std::cout << "Process os08\_04x №2 wasn't created\n";  }  WaitForSingleObject(pi1.hProcess, INFINITE);  WaitForSingleObject(pi2.hProcess, INFINITE);  CloseHandle(htimer);  std::cout << "Elapsed time = " << clock() - start << " clocks\n";  system("pause");  return 0;  } |



**Задание 05. Linux**

1. Разработайте приложение **OS08\_05**.
2. Приложение **OS08\_05** выводит на консоль текущую локальную дату и время в формате ***дд.мм.ггг чч:мин:сек***.

**Задание 06. Linux**

1. Разработайте приложение **OS08\_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 **OS08\_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.