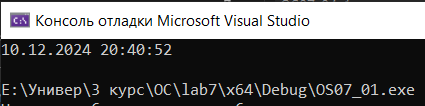
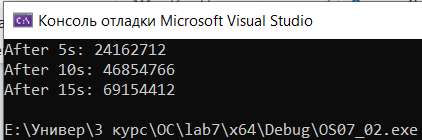
|  |
| --- |
| #include <iostream>  #include <iomanip>  #include <ctime>  using namespace std;  int main() {  time\_t t = time(nullptr);  tm localTime;  localtime\_s(&localTime, &t);  cout << put\_time(&localTime, "%d.%m.%Y %H:%M:%S") << endl;  return 0;  } |

Задание 1



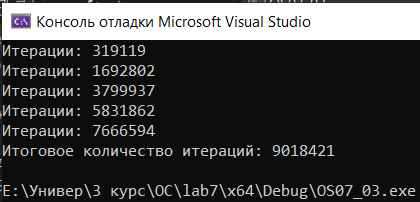
Задание 2

|  |
| --- |
| #include <Windows.h>  #include <iostream>  #include <ctime>  using namespace std;  void checkK(clock\_t start, bool& flag, int k, int time) {  if ((clock() - start) / CLOCKS\_PER\_SEC == time && flag) {  cout << "After " << time << "s: " << k << '\n';  flag = false;  }  }  int main() {  clock\_t start = clock();  int k = 0;  bool flag5sec = true, flag10sec = true;  while (true) {  k++;  checkK(start, flag5sec, k, 5);  checkK(start, flag10sec, k, 10);  if ((clock() - start) / CLOCKS\_PER\_SEC == 15) {  cout << "After 15s: " << k << '\n';  break;  }  }  return 0;  } |



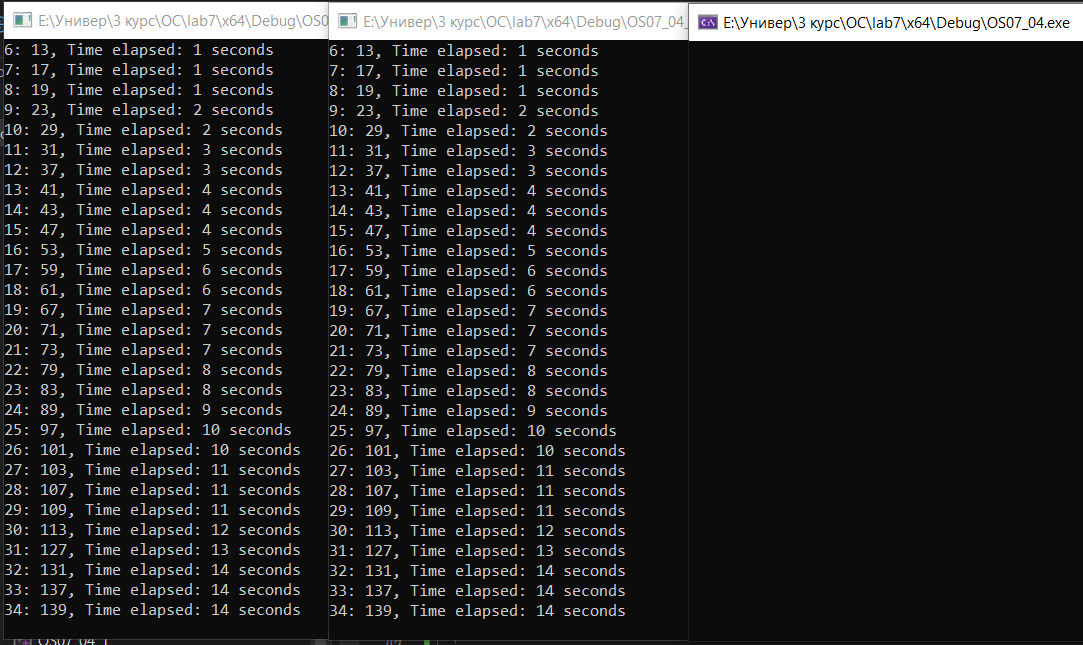
Задание 3

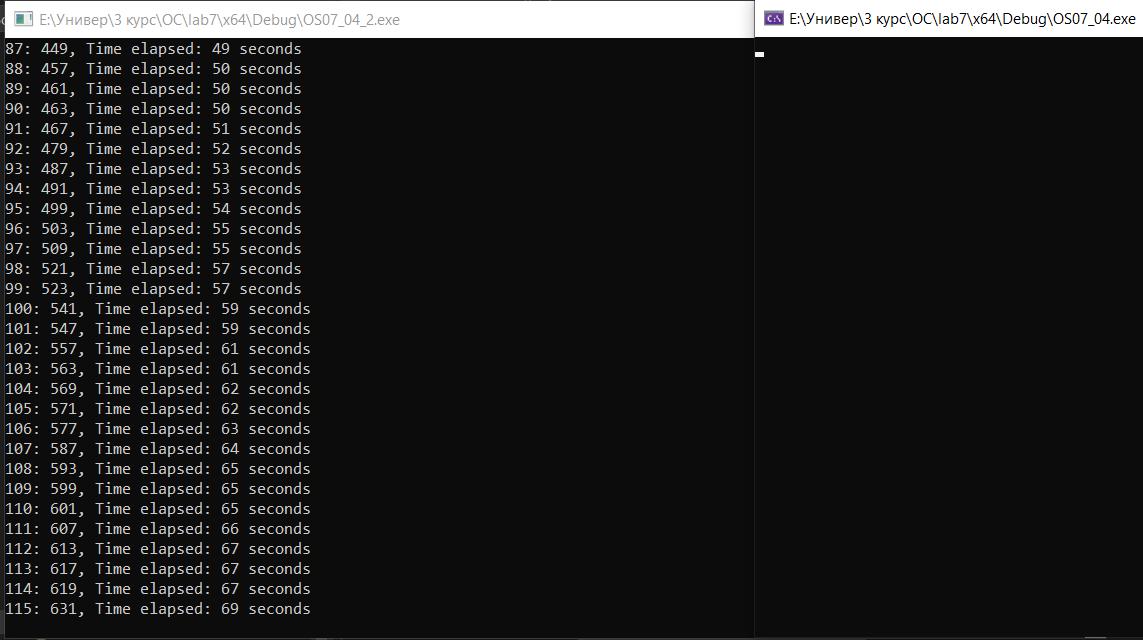
|  |
| --- |
| #include <Windows.h>  #include <iostream>  using namespace std;  void CALLBACK PrintIterations(LPVOID lpArg, DWORD dwTimerLowValue, DWORD dwTimerHighValue) {  int\* counter = reinterpret\_cast<int\*>(lpArg);  cout << "Итерации: " << \*counter << endl;  }  void infintyCycle(int& counter, DWORD startTime) {  while (true) {  counter++;  if (GetTickCount64() - startTime >= 15000)  break;  SleepEx(0, TRUE);  }  }  int main() {  SetConsoleOutputCP(1251);  SetConsoleCP(1251);  HANDLE hTimer = CreateWaitableTimer(NULL, FALSE, NULL);  LARGE\_INTEGER Time;  Time.QuadPart = -10000000;  if (hTimer == NULL) {  cout << "Ошибка создания таймера" << endl;  return 1;  }  int counter = 0;  DWORD startTime = GetTickCount64();  if (!SetWaitableTimer(hTimer, &Time, 3000, PrintIterations, &counter, TRUE)) {  cout << "Ошибка установки таймера" << endl;  CloseHandle(hTimer);  return 1;  }  infintyCycle(counter, startTime);    cout << "Итоговое количество итераций: " << counter << endl;  CancelWaitableTimer(hTimer);  CloseHandle(hTimer);  return 0;  } |



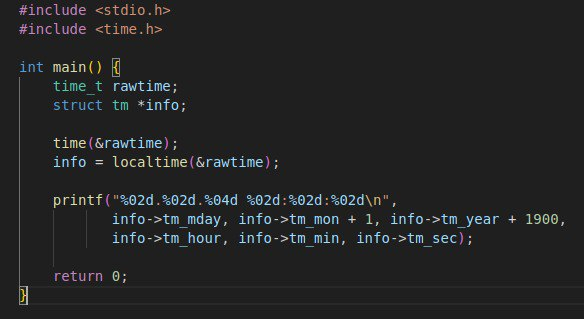
Задание 4

|  |
| --- |
| #include <iostream>  #include "string"  #include <Windows.h>  using namespace std;  STARTUPINFO si1, si2;  PROCESS\_INFORMATION pi1, pi2;  BOOL createProc(const wchar\_t\* path, STARTUPINFO &si, PROCESS\_INFORMATION &pi) {  BOOL bRes = CreateProcessW(path,  NULL, NULL, NULL, FALSE, CREATE\_NEW\_CONSOLE, NULL, NULL, &si, &pi);  if (!bRes) {  printf("error %d\n", GetLastError());  }  return bRes;  }  int main() {    setlocale(LC\_ALL, "rus");  const wchar\_t\* pathProc1 = L"E:\\Универ\\3 курс\\OC\\lab7\\x64\\Debug\\OS07\_04\_1.exe";  const wchar\_t\* pathProc2 = L"E:\\Универ\\3 курс\\OC\\lab7\\x64\\Debug\\OS07\_04\_1.exe";  createProc(pathProc1, si1, pi1);  ZeroMemory(&si1, sizeof(si1));  si1.cb = sizeof(si1);  ZeroMemory(&pi1, sizeof(pi1));  createProc(pathProc2, si2, pi2);  ZeroMemory(&si2, sizeof(si2));  si2.cb = sizeof(si2);  ZeroMemory(&pi2, sizeof(pi2));  WaitForSingleObject(pi2.hProcess, INFINITE);  WaitForSingleObject(pi1.hProcess, INFINITE);  CloseHandle(pi1.hProcess);  CloseHandle(pi2.hProcess);  } |





Задание 5





Задание 6

|  |
| --- |
| #include <stdio.h>  #include <time.h>  int main() {  clock\_t start\_time, current\_time;  time\_t r\_start\_time, r\_end\_time;  // Получаем время в начале выполнения кода  r\_start\_time = time(NULL);    int counter = 0;  start\_time = clock();  while (1) {  counter++;  current\_time = clock();  if (((double)(current\_time - start\_time)) / CLOCKS\_PER\_SEC >= 2.0) {  r\_end\_time = time(NULL);  // Вычисляем разницу во времени  double elapsed\_time = difftime(r\_end\_time, r\_start\_time);    printf("Iterations: %d\n", counter);  printf("Elapsed time: %.20f seconds\n", ((double)(current\_time - start\_time)) / CLOCKS\_PER\_SEC);  printf("RealTime: %.20f senods\n", elapsed\_time);  break;  }  }    return 0;  } |

