Задание 1.

Написать программу, реализующую упаковку и распаковку zip архивов. Программа должна использовать утилиту 7z.exe, которая будет непосредственно выполнять упаковку и распаковку файлов путем запуска в дочернем процессе. Программа должна поддерживать такие операции как:

1. Распаковка архива в папку
2. Упаковка одного файла в новый архив

Для получения максимальной оценки необходимо выполнить обработку ошибок от дочернего процесса путем перенаправления потока вывода. Это позволит родительскому процессу получить содержимое консоли, сформированное программой 7z.exe и по этому тексту определить была ошибка или нет.

Код программы

```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
#include <windows.h>
#include <locale.h>

#define BUFFER_SIZE 1024
#define MAXFILENUMBERS 8



bool unzippingFiles()
{
  bool result = true;
  LPSTR inputFileName = new CHAR[MAX_PATH];
  LPSTR outputFileName = new CHAR[MAX_PATH];
  printf("Full path to archive: ");
  scanf("%s", inputFileName);
  printf("Full output path: ");
  scanf("%s", outputFileName);
  LPSTR cmdLine = new CHAR[MAX_PATH];
  ZeroMemory(cmdLine, MAX_PATH);
  strncpy(cmdLine, "7z.exe zip ", MAX_PATH - strlen(cmdLine));
  strncat(cmdLine, inputFileName, MAX_PATH - strlen(cmdLine));
  strncat(cmdLine, " -o", MAX_PATH - strlen(cmdLine));
  strncat(cmdLine, outputFileName, MAX_PATH - strlen(cmdLine));
  strncat(cmdLine, " -y", MAX_PATH - strlen(cmdLine));
  STARTUPINFOA startInfo;
  ZeroMemory(&startInfo, sizeof(startInfo));
  startInfo.cb = sizeof(startInfo);
  startInfo.dwFlags |= STARTF_USESTDHANDLES;
  PROCESS_INFORMATION procInfo;
  ZeroMemory(&procInfo, sizeof(procInfo));
  if (!CreateProcessA(NULL, cmdLine, NULL, NULL, TRUE, NORMAL_PRIORITY_CLASS, NULL, NULL,
&startInfo, &procInfo)) {
    result = false;
  }
  return result;
}
bool archivingFiles()
{
  bool result = true;
  LPSTR* inputFiles = new CHAR *[MAXFILENUMBERS];
  int fileCount;
  printf("Number of files: ");
```

```
  scanf("%d", &fileCount);
  printf("Full path to %d files\n", fileCount);
  LPSTR fileName = new CHAR[MAX_PATH];
  for (int i = 0; i < fileCount; i++) {
    printf("Full path to file:");
    scanf("%s", fileName);
    inputFiles[i] = new CHAR[MAX_PATH];
    strcpy(inputFiles[i], fileName);
  }
  LPSTR archiveName = new CHAR[MAX_PATH];
  printf("Full path to archive: ");
  scanf("%s", archiveName);
  // Create anonim channel
  HANDLE hInReadPipe;
  HANDLE hInWritePipe;
  HANDLE hOutReadPipe;
  HANDLE hOutWritePipe;
  CreatePipe(&hInReadPipe, &hInWritePipe, NULL, 0);
  CreatePipe(&hOutReadPipe, &hOutWritePipe, NULL, 0);
  if (hInReadPipe == INVALID_HANDLE_VALUE || hInWritePipe == INVALID_HANDLE_VALUE ||
    hOutReadPipe == INVALID_HANDLE_VALUE || hOutWritePipe == INVALID_HANDLE_VALUE) {
    /*  print_problems();*/
    result = false;
  }
  STARTUPINFOA startInfo;
  ZeroMemory(&startInfo, sizeof(startInfo));
  startInfo.cb = sizeof(startInfo);      // Initialize size of structure in Byte
  startInfo.hStdInput = hInReadPipe;     //Defines an input descriptor for a process
  startInfo.hStdOutput = hOutWritePipe;  //Defines an input descriptor for a process
  startInfo.hStdError = hOutWritePipe;   //Defines an error descriptor for a process
  startInfo.dwFlags |= STARTF_USESTDHANDLES;
  // Set standard input, standard output, and standard error handling
  PROCESS_INFORMATION procInfo;
  ZeroMemory(&procInfo, sizeof(procInfo));
  LPSTR cmdLine = new CHAR[MAX_PATH];
  ZeroMemory(cmdLine, MAX_PATH);
  // a (Add) command
  strncpy(cmdLine, "7z.exe zip ", MAX_PATH - strlen(cmdLine));
  strncat(cmdLine, archiveName, MAX_PATH - strlen(cmdLine));
  for (int i = 0; i < fileCount; i++)
  {
    strncat(cmdLine, " ", MAX_PATH - strlen(cmdLine));
    strncat(cmdLine, inputFiles[i], MAX_PATH - strlen(cmdLine));
  }
  strncat(cmdLine, " ", MAX_PATH - strlen(cmdLine));
  //Create new process.
  if (!CreateProcessA(NULL, cmdLine, NULL, NULL, TRUE, NORMAL_PRIORITY_CLASS, NULL, NULL,
&startInfo, &procInfo)) {
    //print_problems();
    result = false;
  }
  if (result) {
    WaitForSingleObject(procInfo.hProcess, INFINITE);
    DWORD readBytes = 0;
    LPSTR outBuffer = new CHAR[BUFFER_SIZE];
    ZeroMemory(outBuffer, BUFFER_SIZE);
    OVERLAPPED overlapped;
    ReadFile(hOutReadPipe, outBuffer, BUFFER_SIZE, &readBytes, &overlapped);
    CloseHandle(procInfo.hProcess);
    CloseHandle(procInfo.hThread);
  }
  CloseHandle(hOutReadPipe);
  CloseHandle(hOutWritePipe);
  CloseHandle(hInReadPipe);
  CloseHandle(hInWritePipe);
```

```
  return result;
}
int main(int argc, char* argv[])
{
  if (((strcmp(argv[1], "-zip") != 0) && (strcmp(argv[1], "-unzip")) != 0)) {
    printf("\n Something went wrong :(\n");
    exit(0);
  }
  if (strcmp(argv[1], "-zip") == 0) {
    if (archivingFiles())
      printf("Your files have been archived :)");
    else printf("Your files haven't been archived :(");
  }
  else {
    if (unzippingFiles())
      printf("The files were unziped.");
    else printf("The files were not unziped.");
  }
  return 0;
}
```

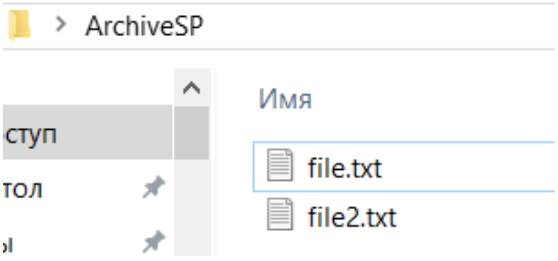

Рисунок 1 – Файлы, которые нужно заархивировать

```
C:\Users\Veronika>C:\Users\Veronika\source\repos\Lab4\Debug\Lab4.exe -zip
Number of files: 2
Full path to 2 files
Full path to file:C:\Users\Veronika\Desktop\ArchiveSP\file2.txt
Full path to file:C:\Users\Veronika\Desktop\ArchiveSP\file.txt
Full path to archive: C:\Users\Veronika\Desktop\newfolder
Your files have been archived :)
```

Рисунок 2 – Архивация 2 файлов

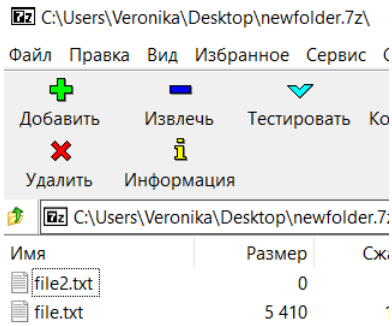

Рисунок 3 – Заархивированные файлы

```
C:\Users\Veronika>C:\Users\Veronika\source\repos\Lab4\Debug\Lab4.exe -unzip
Full path to archive: C:\Users\Veronika\Desktop\newfolder
Full output path: C:\Users\Veronika\Desktop
The files were unziped.
```

Рисунок 4 – Разархивированные файлы

Задание 2

Написать программу, которая может создавать 2 и более потоков (кол-во задается в командной строке). Перед запуском потоков программа заполняет для каждого потока исходный массив целочисленных значений (5-10 элементов) от 10 до 100. Каждый поток должен найти для каждого элемента массива его наибольший делитель, сохраняя полученные значения в TLS память. После нахождения всех значений он должен вывести сумму всех полученных значений и напечатать свой идентификатор. Расчет наибольшего делителя и вычисление конечной суммы должны реализовываться двумя отдельными функциями.

Код программы

```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
#include <windows.h>
#include <locale.h>
#include <tchar.h>


#define MAXVAL 100
#define MINVAL 10

#define MINCOUNT 5
#define MAXCOUNT 10


int addrInd, sizeInd;

int SumElements() {

  int* arr = (int*)TlsGetValue(sizeInd);
  int count = (int)TlsGetValue(addrInd);
  int r = 0;
  for (size_t i = 0; i < count; i++)
  {
    r += arr[i];
  }
  return r;
}
void Divider(int* arr, int count, int* array) {

  for (size_t i = 0; i < count; i++)
  {
    int curr = arr[i];
    array[i] = curr;
  }
}
void Thread() {
```

```
  //making generator random
  DWORD currId = GetCurrentThreadId();
  srand((unsigned int)currId);

  //get count
  int count = rand() % (MAXCOUNT - MINCOUNT) + MINCOUNT;

  //making values random
  int* values = new int[count];
  for (size_t j = 0; j < count; j++)
  {
    values[j] = rand() % (MAXVAL - MINVAL) + MINVAL;
  }

  int* Array = new int[count];
  Divider(values, count, Array);

  //save to TLS
  if (!TlsSetValue(sizeInd, (LPVOID)Array)) {
    printf("First value has't been set\n");
  }

  if (!TlsSetValue(addrInd, (LPVOID)count)) {
    printf("Second value  has't been set\n");
  }

  printf("Thread ID: %d Sum: %d\n", currId, SumElements());

  delete[] values;
  delete[] Array;
}

int main(int argc, TCHAR** argv)
{
  if (argc != 2) {
    printf("You entered an invalid number of arguments");
    exit(0);
  }
  int countThreads = _ttoi(argv[1]);

  HANDLE* threads = new HANDLE[countThreads];
  //allocate tls for all the threads
  addrInd = TlsAlloc();
  sizeInd = TlsAlloc();
  for (size_t i = 0; i < countThreads; i++)
  {
    DWORD IDThread;
    threads[i] = CreateThread(NULL, 0, (LPTHREAD_START_ROUTINE)Thread, NULL, 0,
&IDThread);
  }

  for (size_t i = 0; i < countThreads; i++)
    WaitForSingleObject(threads[i], INFINITE);

  TlsFree(addrInd);//free TLS
  delete[] threads;

  return 0;
}
```

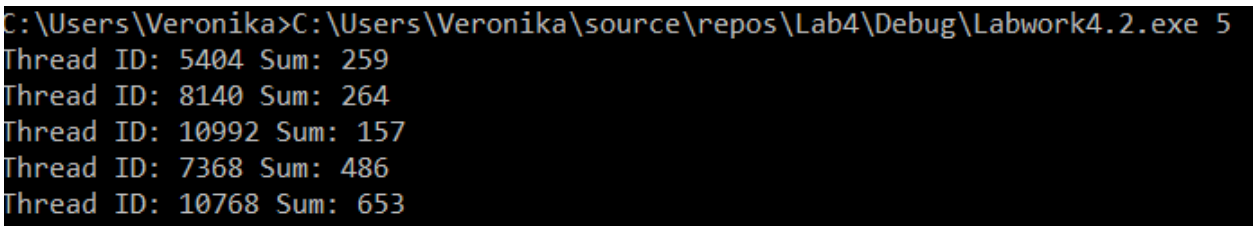

Рисунок 5 – Выполнение программы