МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«ВЯТСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»
Институт математики и информационных систем
Факультет автоматики и вычислительной техники
Кафедра электронных вычислительных машин

Дата сдачи на проверку:

«__»_____ 2025 г.

Проверено:

«__»_____ 2025 г.

ВНЕШНЯЯ СОРТИРОВКА ФАЙЛА. РАБОТА С ФОРМАМИ В LAZARUS.

Отчёт по лабораторной работе №8

по дисциплине

«Программирование»

Разработал студент гр. ИВТб-1301-05-00 _____/Черкасов А. А./

(подпись)

Заведующая кафедры ЭВМ _____/Долженкова М. Л./

(подпись)

Работа защищена «__»_____ 2025 г.

Киров

2025

Цель

Цель работы: Получение навыков реализации алгоритма внешней сортровки в Lazarus на языке pascal.

Задание

– Написать программу с графическим интерфейсом для генерации и внешней сортировки файла.

– Предусмотреть визуальное отображение работы программы с помощью инструментов Lazarus.

Решение

Схемы алгоритмов решения задания представлены на рисунках 1.1, 1.2, 1.3, 1.4.1, 1.4.2, 1.5 и 1.6. Исходный код решений представлен в Приложениях А1.

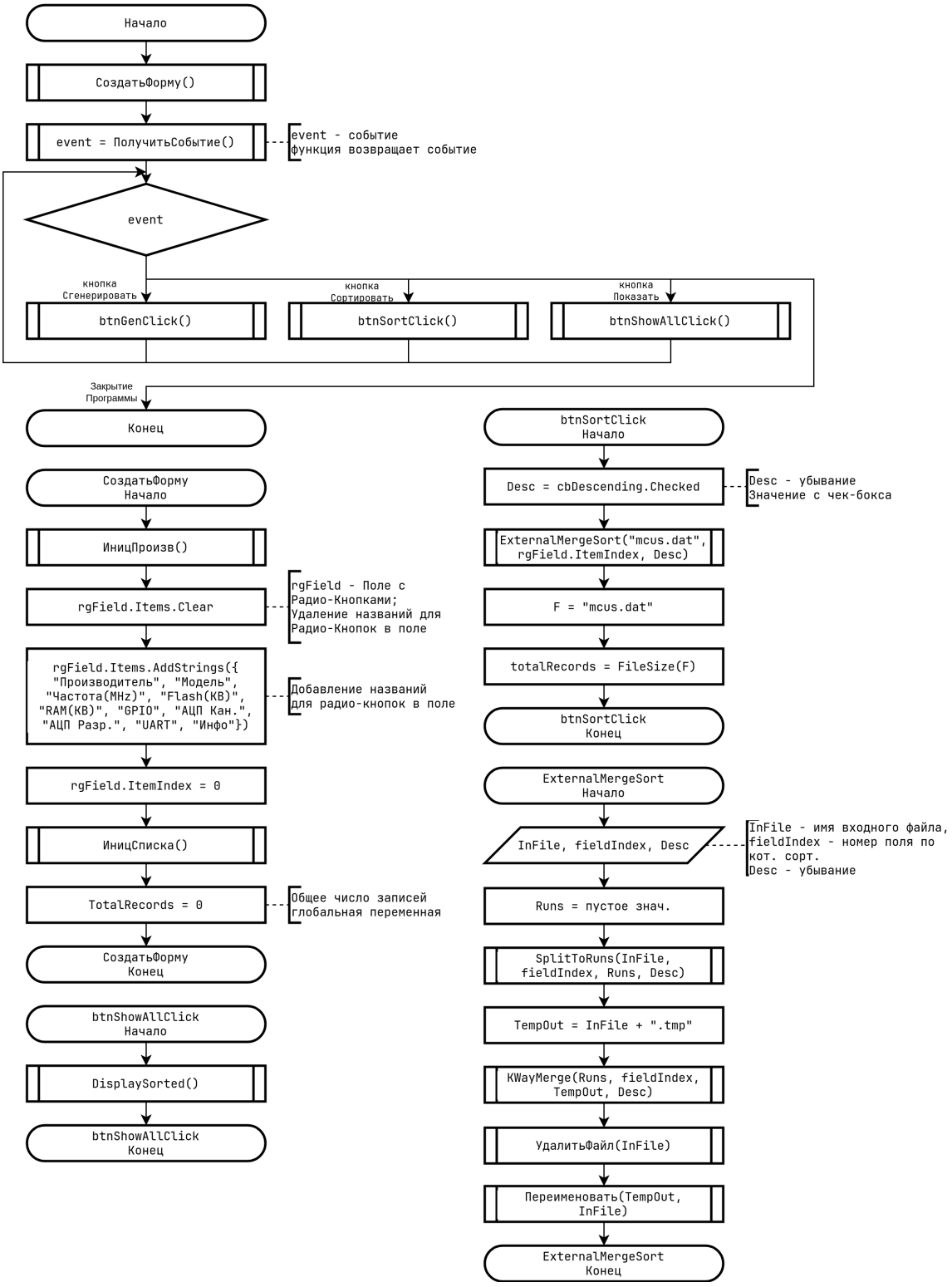

Рисунок 1.1 - Схемы алгоритмов главной программы, подпрограммы инициализации формы, подпрограмм нажатия на кнопки Показать и Сортировать, подпрограммы вызова внешней сортировки.

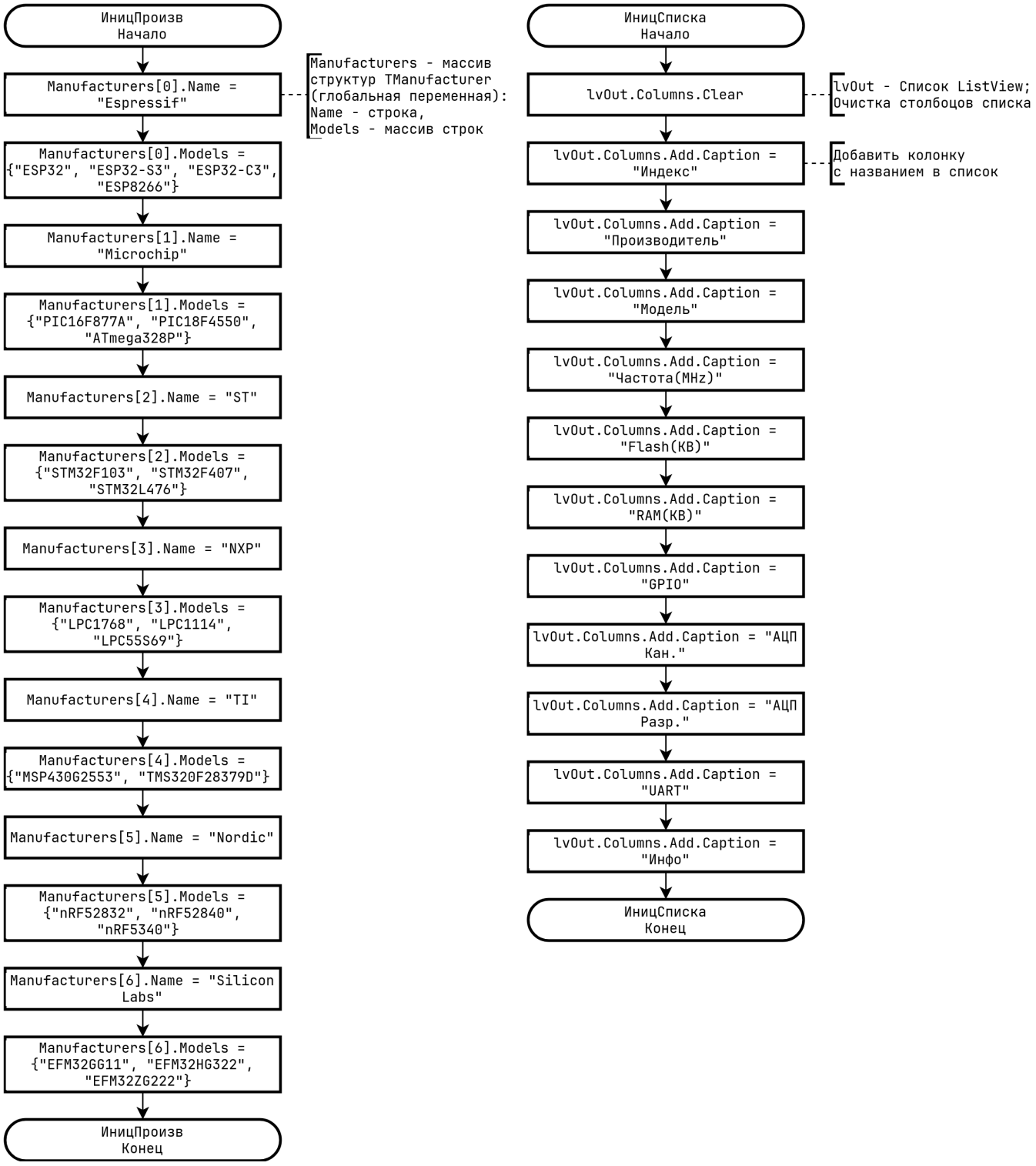

Рисунок 1.2 - Схемы алгоритмов подпрограмм инициализации Производителей и Списка.

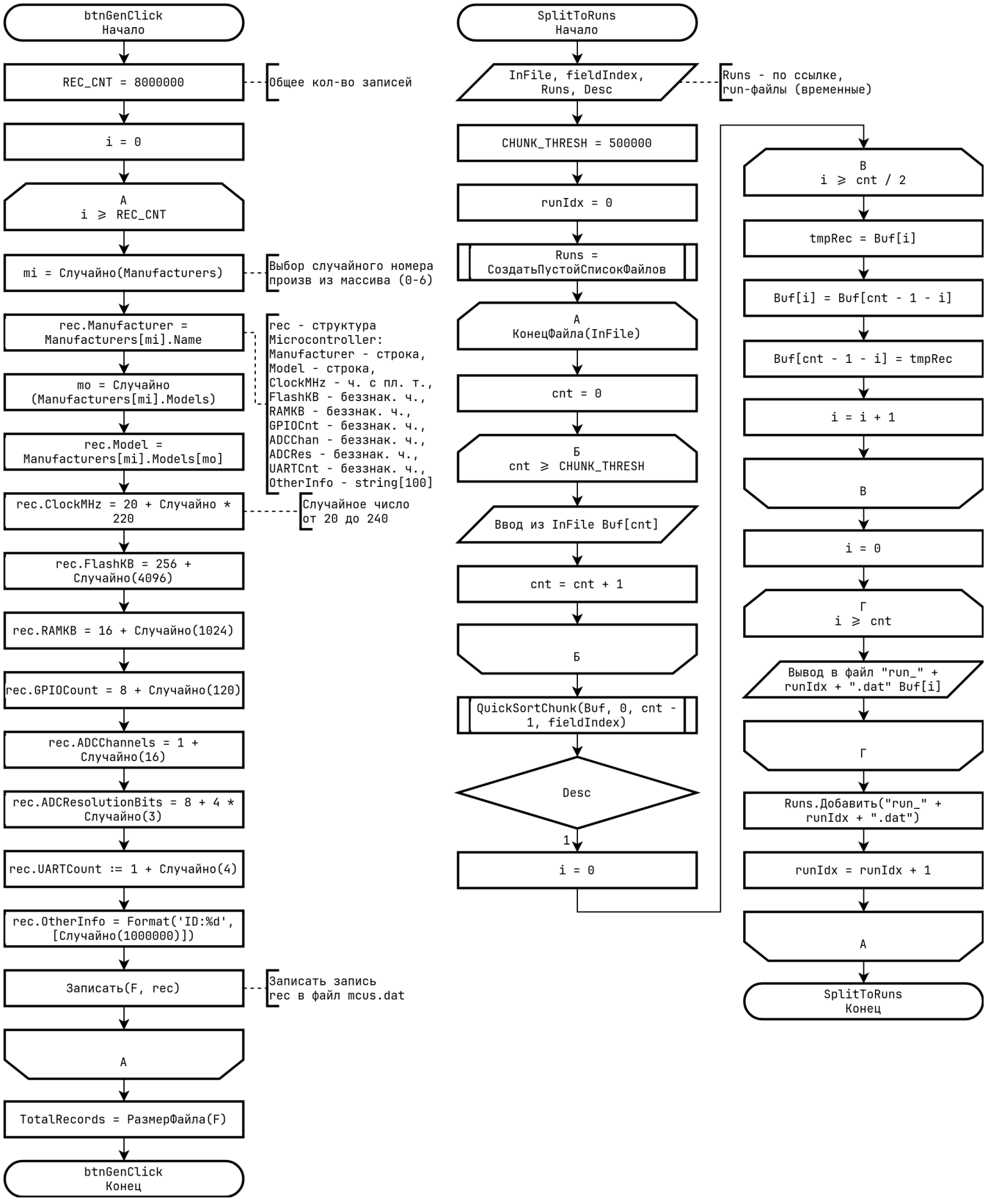

Рисунок 1.3 - Схемы алгоритмов подпрограмм нажатия на кнопку Генерация, разбиения на временные(ран-файлы).

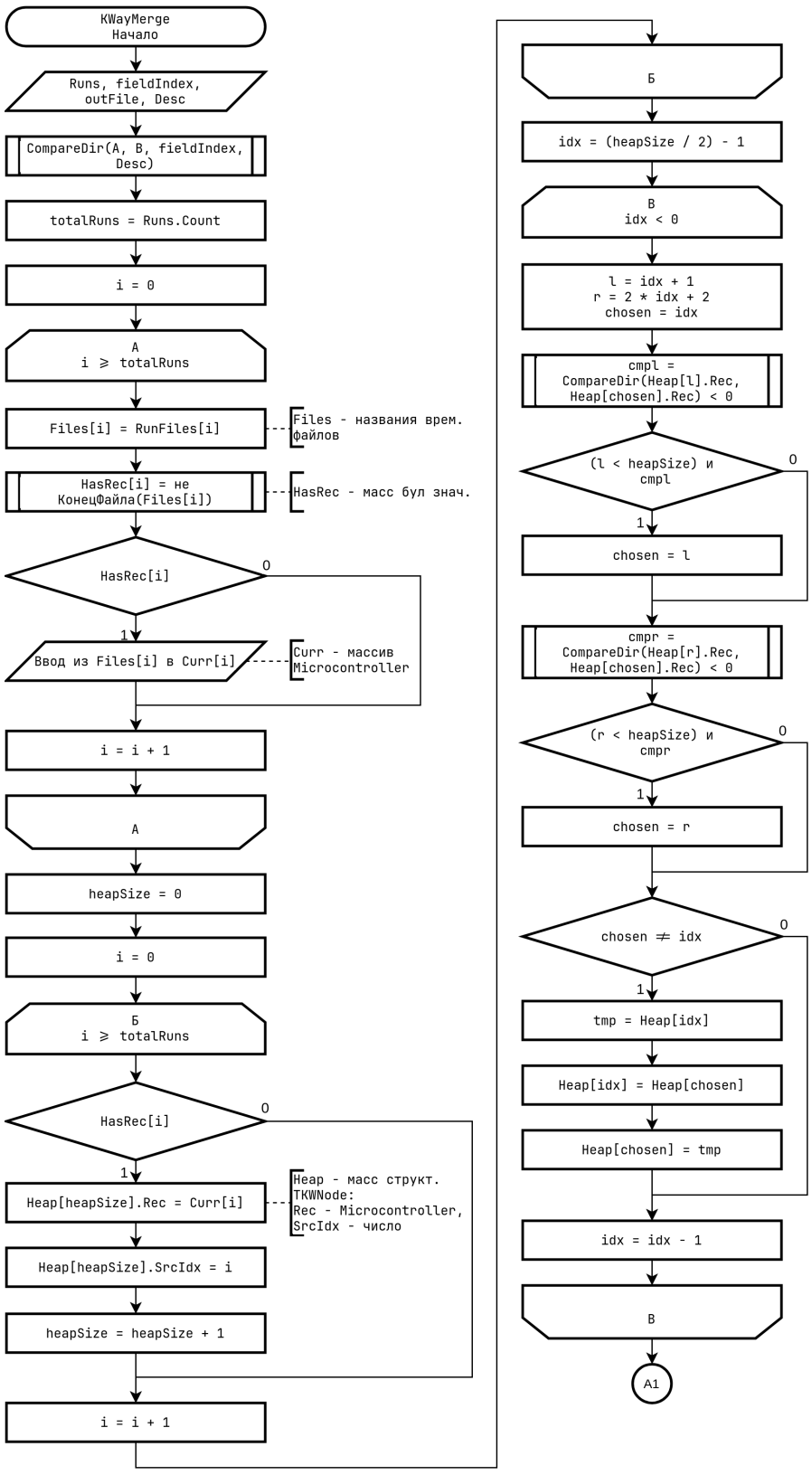

Рисунок 1.4.1 - Схема алгоритма подпрограммы многопутевого слияния ч.1.

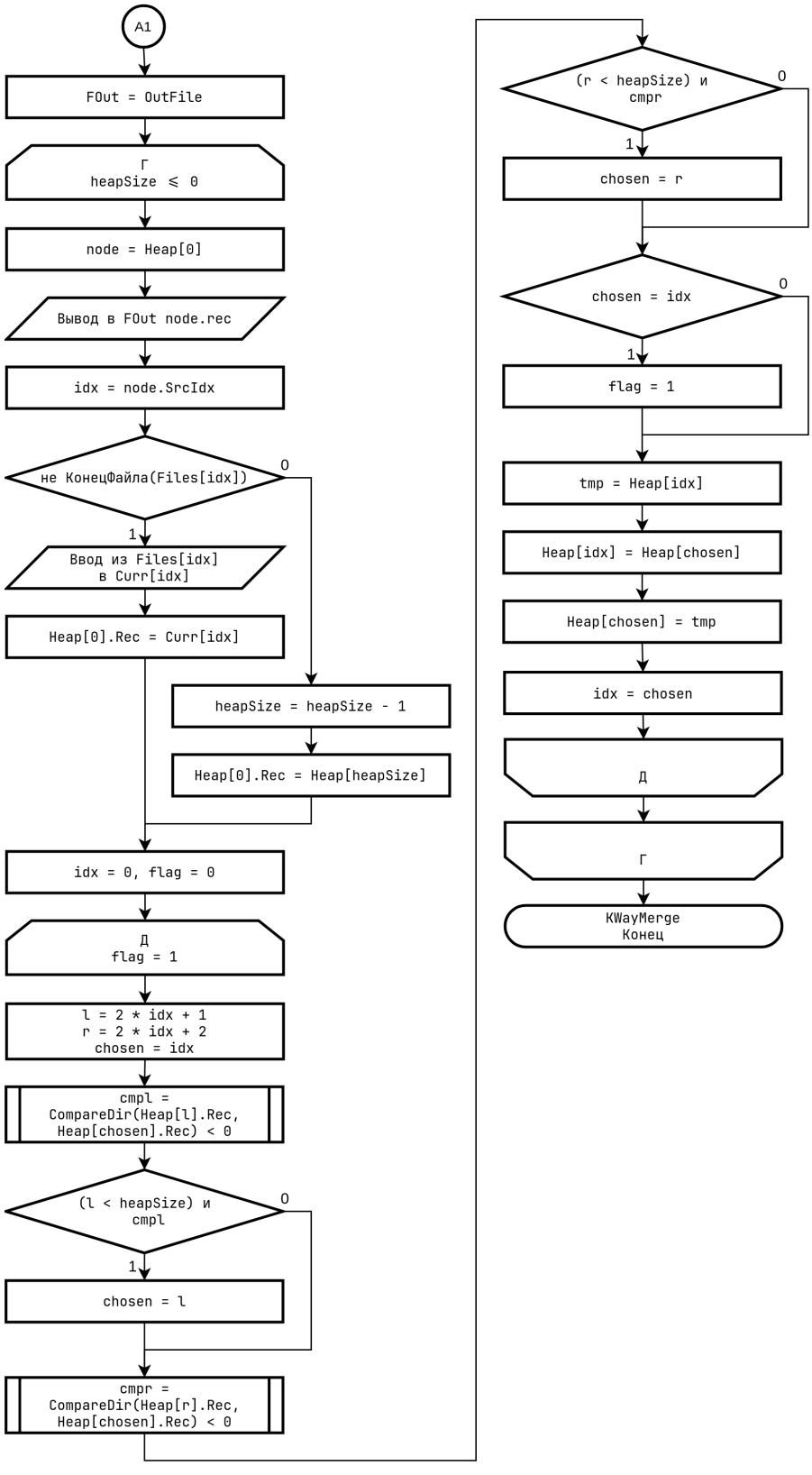

Рисунок 1.4.2 - Схема алгоритма подпрограммы многопутевого слияния ч.2.

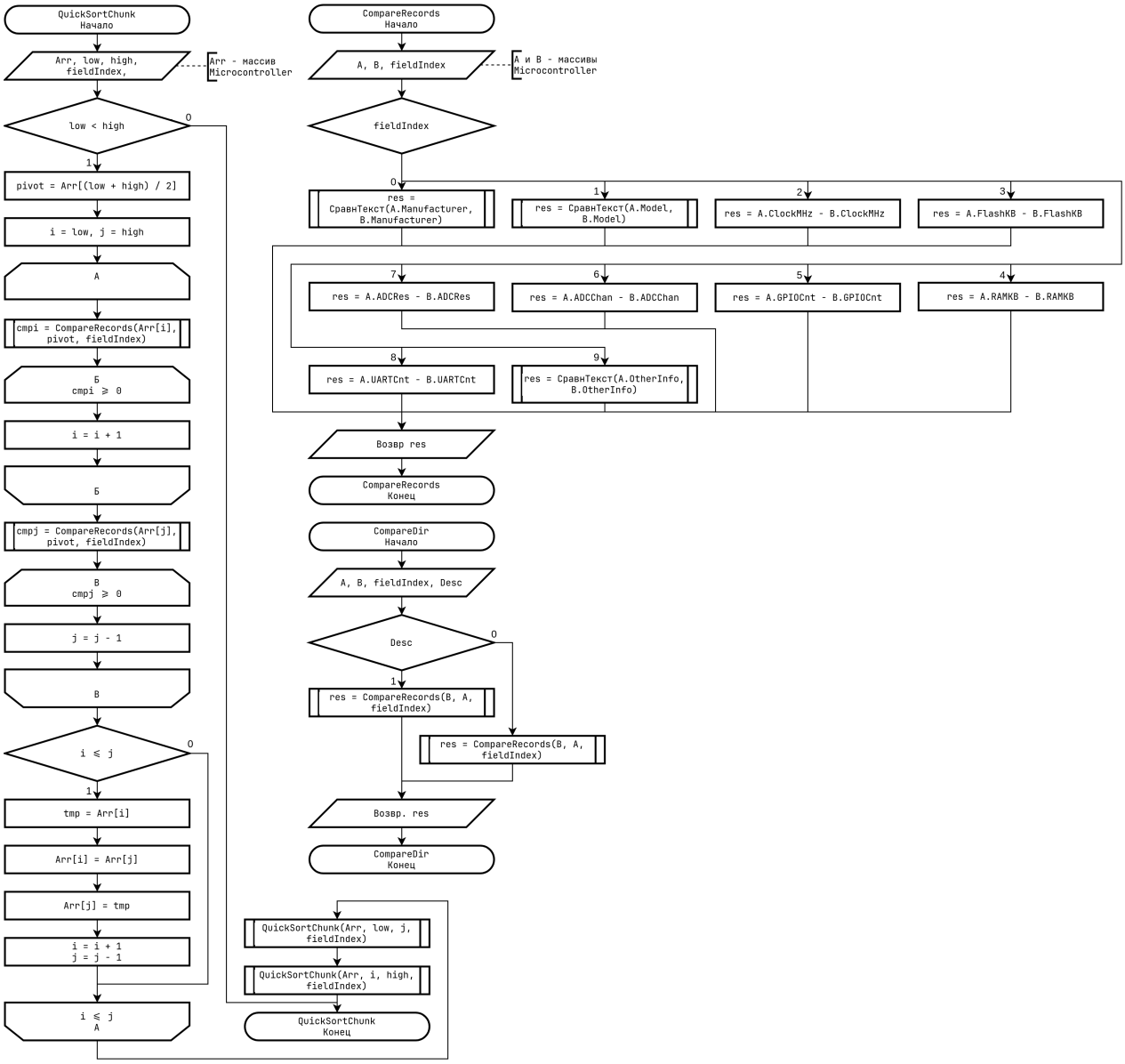

Рисунок 1.5 - Схема алгоритма подпрограммы быстрой сортировки чанка и подпрограммы сравнения двух элементов из списка по полю.

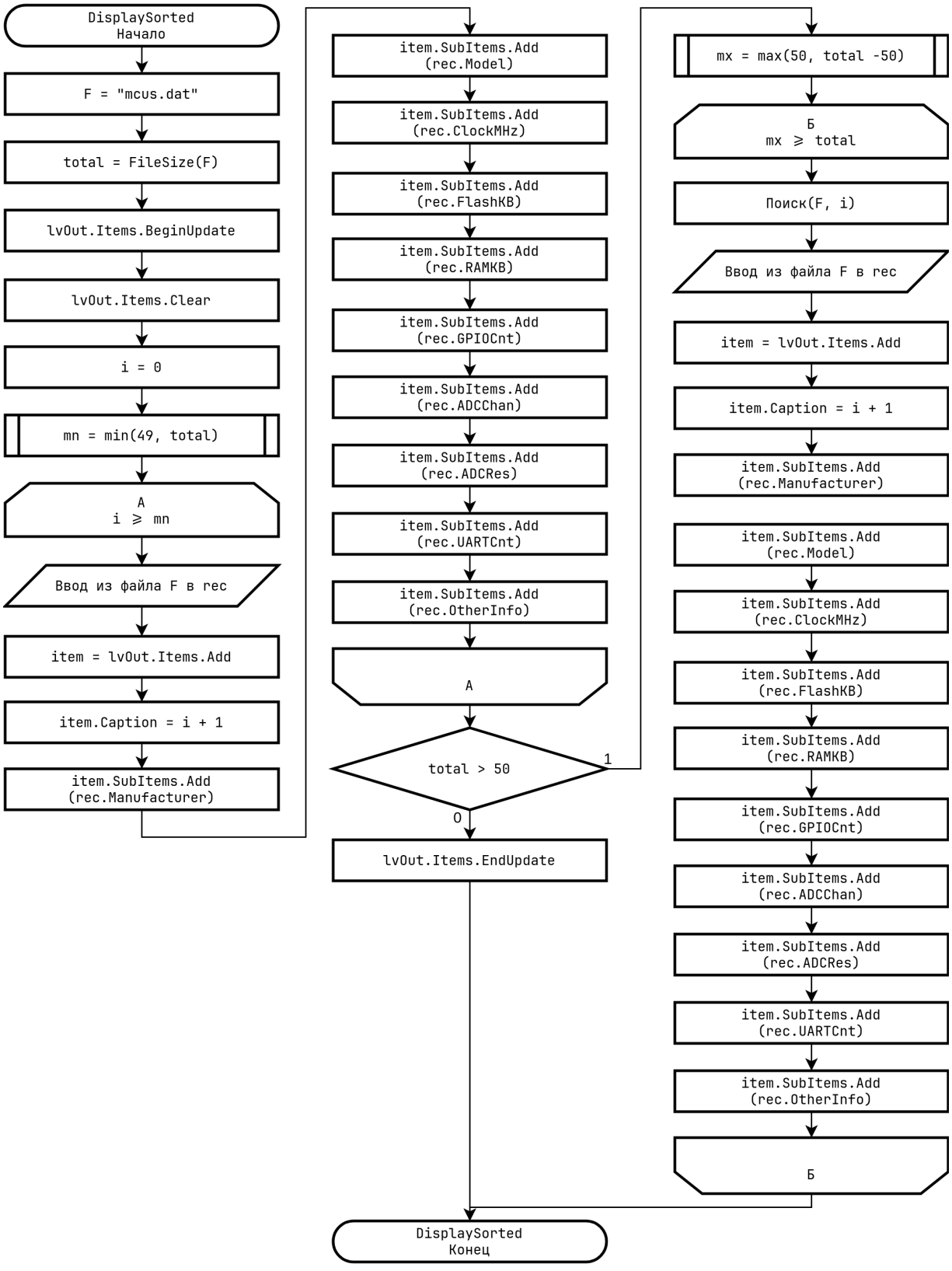

Рисунок 1.6 - Схема алгоритма подпрограммы вывода отсортированного списка.

Вывод

В ходе лабораторной работы была разработана программа с графическим интерфейсом для внешней сортировки файлов, что позволило освоить алгоритмы многопутевого слияния и работу с файлами в Lazarus.

Приложение А1. Исходный код

```
unit lab8_unit;
{$mode objfpc}{$H+}
{$hints off}

interface

uses
  Classes, SysUtils, Forms, Controls, Graphics, Dialogs,
  ComCtrls, StdCtrls, ExtCtrls, Math;

const
  RECORD_COUNT    = 8000000;
  CHUNK_THRESHOLD = 500000;
  PAGE_SIZE       = 100;
  TEMP_SUBDIR     = 'temp_runs';  // подпапка для ран-файлов

type
  TMicrocontroller = record
    Manufacturer: string[50];
    Model: string[50];
    ClockMHz: double;
    FlashKB: cardinal;
    RAMKB: cardinal;
    GPIOCount: word;
    ADCChannels: byte;
    ADCResolutionBits: byte;
    UARTCount: byte;
    OtherInfo: string[100];
  end;
  TMCUFile = file of TMicrocontroller;

  TKWNode = record
```

```
    Rec: TMicrocontroller;
    SourceIdx: integer;
  end;

  TManufacturer = record
    Name: string;
    Models: array of string;
  end;

  { TForm1 }
  TForm1 = class(TForm)
    btnGen: TButton;
    btnSort: TButton;
    btnPrev: TButton;
    btnNext: TButton;
    btnShowAll: TButton;
    lvOut: TListView;
    rgField: TRadioGroup;
    cbDescending: TCheckBox;
    pbProgress: TProgressBar;
    statBar: TStatusBar;

    procedure FormCreate(Sender: TObject);
    procedure btnGenClick(Sender: TObject);
    procedure btnSortClick(Sender: TObject);
    procedure btnShowAllClick(Sender: TObject);
    procedure btnNextClick(Sender: TObject);
    procedure btnPrevClick(Sender: TObject);
  private
    CurrentPage: Int64;
    TotalRecords: Int64;
    Manufacturers: array of TManufacturer;

    procedure InitManufacturers;
```

```
    procedure SetupListView;
    procedure DisplaySample(const ACaption: string);
    procedure DisplayPage(Page: Int64);

    function  CompareRecords(const A, B: TMicrocontroller; fieldIndex: integer): integer;
    procedure QuickSortChunk(var Arr: array of TMicrocontroller; low, high, fieldIndex: int

    procedure ExternalMergeSort(const InFile: string; fieldIndex: integer; Descending: Bool
    procedure SafeDeleteFile(const FileName: string);

    procedure SplitToRuns(const InFile: string; fieldIndex: integer; out RunFiles: TStringL
    procedure KWayMerge(const RunFiles: TStringList; fieldIndex: integer; const OutFile: st
    function  GetTempDir: string;
    procedure RemoveTempDir;
    procedure VerifySorted(const FileName: string; fieldIndex: integer; Descending: Boolean
  public
  end;

var
  Form1: TForm1;

implementation
{$R *.lfm}

{ GetTempDir: возвращает путь "<exe_folder>\temp_runs\" и создает папку при необходимости }
function TForm1.GetTempDir: string;
begin
  Result := IncludeTrailingPathDelimiter(ExtractFilePath(Application.ExeName))
            + IncludeTrailingPathDelimiter(TEMP_SUBDIR);
  if not DirectoryExists(Result) then
    ForceDirectories(Result);
end;

{ RemoveTempDir: удаляет все файлы в "<exe_folder>\temp_runs\" и саму папку }
```

```
procedure TForm1.RemoveTempDir;
var
  TempDir: string;
  sr: TSearchRec;
  FullPath: string;
begin
  TempDir := GetTempDir;
  if not DirectoryExists(TempDir) then Exit;
  if FindFirst(IncludeTrailingPathDelimiter(TempDir) + '*', faAnyFile, sr) = 0 then
  begin
    repeat
      if (sr.Name <> '.') and (sr.Name <> '..') then
      begin
        FullPath := IncludeTrailingPathDelimiter(TempDir) + sr.Name;
        if (sr.Attr and faDirectory) = 0 then
          DeleteFile(FullPath)
        else
          RemoveDir(FullPath);
      end;
    until FindNext(sr) <> 0;
    FindClose(sr);
  end;
  RemoveDir(TempDir);
end;

{ SafeDeleteFile: удаляет файл, если он существует }
procedure TForm1.SafeDeleteFile(const FileName: string);
begin
  if FileExists(FileName) then
    DeleteFile(FileName);
end;

{ SetupListView: настраивает столбцы ListView для отображения полей микроконтроллера }
procedure TForm1.SetupListView;
```

```
begin
  lvOut.ViewStyle := vsReport;
  lvOut.Columns.Clear;
  with lvOut.Columns do
  begin
    with Add do Caption := 'Индекс';
    with Add do Caption := 'Производитель';
    with Add do Caption := 'Модель';
    with Add do Caption := 'Частота(MHz)';
    with Add do Caption := 'Flash(KB)';
    with Add do Caption := 'RAM(KB)';
    with Add do Caption := 'GPIO';
    with Add do Caption := 'АЦП Кан.';
    with Add do Caption := 'АЦП Разр.';
    with Add do Caption := 'UART';
    with Add do Caption := 'Инфо';
  end;
end;

{ InitManufacturers: инициализирует массив производителей и их моделей }
procedure TForm1.InitManufacturers;
begin
  SetLength(Manufacturers, 7);
  Manufacturers[0].Name := 'Espressif';
  Manufacturers[0].Models := ['ESP32', 'ESP32-S3', 'ESP32-C3', 'ESP8266'];
  Manufacturers[1].Name := 'Microchip';
  Manufacturers[1].Models := ['PIC16F877A', 'PIC18F4550', 'ATmega328P'];
  Manufacturers[2].Name := 'ST';
  Manufacturers[2].Models := ['STM32F103', 'STM32F407', 'STM32L476'];
  Manufacturers[3].Name := 'NXP';
  Manufacturers[3].Models := ['LPC1768', 'LPC1114', 'LPC55S69'];
  Manufacturers[4].Name := 'TI';
  Manufacturers[4].Models := ['MSP430G2553', 'TMS320F28379D'];
  Manufacturers[5].Name := 'Nordic';
```

```
  Manufacturers[5].Models := ['nRF52832', 'nRF52840', 'nRF5340'];
  Manufacturers[6].Name := 'Silicon Labs';
  Manufacturers[6].Models := ['EFM32GG11', 'EFM32HG322', 'EFM32ZG222'];
end;

{ FormCreate: выполняется при создании формы, инициализирует UI и данные }
procedure TForm1.FormCreate(Sender: TObject);
begin
  InitManufacturers;
  rgField.Items.Clear;
  rgField.Items.AddStrings([
    'Производитель','Модель','Частота(MHz)','Flash(KB)','RAM(KB)',
    'GPIO','АЦП Кан.','АЦП Разр.','UART','Инфо'
  ]);
  rgField.ItemIndex := 0;
  SetupListView;
  CurrentPage := 0;
  TotalRecords := 0;

  pbProgress.Position := 0;
  pbProgress.Max := 100;

  btnGen.Caption := 'Генерировать';
  btnSort.Caption := 'Сортировать';
  btnPrev.Caption := 'Пред.';
  btnNext.Caption := 'След.';
  btnShowAll.Caption := 'Показать все';
  cbDescending.Caption := 'По убыванию';
end;

{ CompareRecords: сравнивает две записи TMicrocontroller по выбранному полю }
function TForm1.CompareRecords(const A, B: TMicrocontroller; fieldIndex: integer): integer
begin
  case fieldIndex of
```

```
    0: Result := CompareText(A.Manufacturer, B.Manufacturer);
    1: Result := CompareText(A.Model, B.Model);
    2: Result := Sign(A.ClockMHz - B.ClockMHz);
    3: Result := Integer(A.FlashKB) - Integer(B.FlashKB);
    4: Result := Integer(A.RAMKB) - Integer(B.RAMKB);
    5: Result := Integer(A.GPIOCount) - Integer(B.GPIOCount);
    6: Result := Integer(A.ADCChannels) - Integer(B.ADCChannels);
    7: Result := Integer(A.ADCResolutionBits) - Integer(B.ADCResolutionBits);
    8: Result := Integer(A.UARTCount) - Integer(B.UARTCount);
  else
    Result := CompareText(A.OtherInfo, B.OtherInfo);
  end;
end;

{ QuickSortChunk: сортирует массив записей быстрой сортировкой }
procedure TForm1.QuickSortChunk(var Arr: array of TMicrocontroller; low, high, fieldIndex:
var
  i, j: integer;
  pivot, tmp: TMicrocontroller;
begin
  if low >= high then Exit;
  pivot := Arr[(low + high) div 2];
  i := low; j := high;
  repeat
    while CompareRecords(Arr[i], pivot, fieldIndex) < 0 do Inc(i);
    while CompareRecords(Arr[j], pivot, fieldIndex) > 0 do Dec(j);
    if i <= j then
    begin
      tmp := Arr[i]; Arr[i] := Arr[j]; Arr[j] := tmp;
      Inc(i); Dec(j);
    end;
  until i > j;
  QuickSortChunk(Arr, low, j, fieldIndex);
  QuickSortChunk(Arr, i, high, fieldIndex);
```

```
end;

{ btnGenClick: генерирует 'mcus.dat' с RANDOM-записями микроконтроллеров }
procedure TForm1.btnGenClick(Sender: TObject);
var
  F: TMCUFile;
  rec: TMicrocontroller;
  i, mi, mo: Integer;
begin
  Randomize;
  AssignFile(F, 'mcus.dat');
  Rewrite(F);
  try
    statBar.SimpleText := 'Генерация данных...';
    Application.ProcessMessages;
    for i := 1 to RECORD_COUNT do
    begin
      mi := Random(Length(Manufacturers));
      rec.Manufacturer := Manufacturers[mi].Name;
      mo := Random(Length(Manufacturers[mi].Models));
      rec.Model := Manufacturers[mi].Models[mo];
      rec.ClockMHz := 20 + Random * 220;
      rec.FlashKB := 256 + Random(4096);
      rec.RAMKB := 16 + Random(1024);
      rec.GPIOCount := 8 + Random(120);
      rec.ADCChannels := 1 + Random(16);
      rec.ADCResolutionBits := 8 + 4 * Random(3);
      rec.UARTCount := 1 + Random(4);
      rec.OtherInfo := Format('ID:%d', [Random(1000000)]);
      Write(F, rec);
      if i mod CHUNK_THRESHOLD = 0 then
      begin
        statBar.SimpleText := Format('Сгенерировано %d из %d записей', [i, RECORD_COUNT]);
        Application.ProcessMessages;
```

```
      end;
    end;
    statBar.SimpleText := 'Генерация завершена';
  finally
    CloseFile(F);
  end;

  AssignFile(F, 'mcus.dat');
  Reset(F);
  try
    TotalRecords := FileSize(F);
  finally
    CloseFile(F);
  end;
  CurrentPage := 0;
  DisplaySample('После генерации');
end;

{ btnSortClick: запускает внешнюю сортировку слиянием для 'mcus.dat' }
procedure TForm1.btnSortClick(Sender: TObject);
var
  F: TMCUFile;
  Desc: Boolean;
begin
  if not FileExists('mcus.dat') then Exit;
  Desc := cbDescending.Checked;
  statBar.SimpleText := 'Запуск внешней сортировки слиянием...';
  Application.ProcessMessages;

  ExternalMergeSort('mcus.dat', rgField.ItemIndex, Desc);

  AssignFile(F, 'mcus.dat');
  Reset(F);
  try
```

```
    TotalRecords := FileSize(F);
  finally
    CloseFile(F);
  end;
  CurrentPage := 0;

  if Desc then
    statBar.SimpleText := 'После сортировки по ' + rgField.Items[rgField.ItemIndex] + ' (у
  else
    statBar.SimpleText := 'После сортировки по ' + rgField.Items[rgField.ItemIndex] + ' (в

  DisplaySample(statBar.SimpleText);
  RemoveTempDir; // удаляем папку temp_runs
end;

{ btnShowAllClick: показывает первую страницу записей }
procedure TForm1.btnShowAllClick(Sender: TObject);
begin
  CurrentPage := 0;
  DisplayPage(CurrentPage);
end;

{ btnNextClick: показывает следующую страницу }
procedure TForm1.btnNextClick(Sender: TObject);
var
  MaxP: Int64;
begin
  MaxP := (TotalRecords + PAGE_SIZE - 1) div PAGE_SIZE - 1;
  if CurrentPage < MaxP then Inc(CurrentPage);
  DisplayPage(CurrentPage);
end;

{ btnPrevClick: показывает предыдущую страницу }
procedure TForm1.btnPrevClick(Sender: TObject);
```

```
begin
  if CurrentPage > 0 then Dec(CurrentPage);
  DisplayPage(CurrentPage);
end;

{ DisplayPage: загружает и отображает одну «страницу» данных (PAGE_SIZE записей) }
procedure TForm1.DisplayPage(Page: Int64);
var
  F: TMCUFile;
  rec: TMicrocontroller;
  i, startIdx: Int64;
  item: TListItem;
begin
  AssignFile(F, 'mcus.dat');
  Reset(F);
  try
    TotalRecords := FileSize(F);
    startIdx := Page * PAGE_SIZE;
    if startIdx >= TotalRecords then Exit;
    Seek(F, startIdx);

    lvOut.Items.BeginUpdate;
    try
      lvOut.Items.Clear;
      for i := 0 to PAGE_SIZE - 1 do
      begin
        if startIdx + i >= TotalRecords then Break;
        Read(F, rec);
        item := lvOut.Items.Add;
        item.Caption := IntToStr(startIdx + i + 1);
        with item.SubItems do
        begin
          Add(rec.Manufacturer);
          Add(rec.Model);
```

```
          Add(Format('%.2f', [rec.ClockMHz]));
          Add(IntToStr(rec.FlashKB));
          Add(IntToStr(rec.RAMKB));
          Add(IntToStr(rec.GPIOCount));
          Add(IntToStr(rec.ADCChannels));
          Add(IntToStr(rec.ADCResolutionBits));
          Add(IntToStr(rec.UARTCount));
          Add(rec.OtherInfo);
        end;
      end;
    finally
      lvOut.Items.EndUpdate;
      CloseFile(F);
      statBar.SimpleText := Format('Страница %d из %d', [
        Page + 1,
        (TotalRecords + PAGE_SIZE - 1) div PAGE_SIZE
      ]);
    end;
  except
    CloseFile(F);
    raise;
  end;
end;

{ DisplaySample: показывает в ListView первые и последние 50 записей файла }
procedure TForm1.DisplaySample(const ACaption: string);
var
  F: TMCUFile;
  rec: TMicrocontroller;
  i, total: Int64;
  item: TListItem;
begin
  AssignFile(F, 'mcus.dat');
  Reset(F);
```

```
try
  total := FileSize(F);
  lvOut.Items.BeginUpdate;
  try
    lvOut.Items.Clear;
    statBar.SimpleText := ACaption;
    // первые 50
    for i := 0 to Min(49, total - 1) do
    begin
      Read(F, rec);
      item := lvOut.Items.Add;
      item.Caption := IntToStr(i + 1);
      with item.SubItems do
      begin
        Add(rec.Manufacturer);
        Add(rec.Model);
        Add(Format('%.2f', [rec.ClockMHz]));
        Add(IntToStr(rec.FlashKB));
        Add(IntToStr(rec.RAMKB));
        Add(IntToStr(rec.GPIOCount));
        Add(IntToStr(rec.ADCChannels));
        Add(IntToStr(rec.ADCResolutionBits));
        Add(IntToStr(rec.UARTCount));
        Add(rec.OtherInfo);
      end;
    end;
    // последние 50
    if total > 50 then
      for i := Max(50, total - 50) to total - 1 do
      begin
        Seek(F, i);
        Read(F, rec);
        item := lvOut.Items.Add;
        item.Caption := IntToStr(i + 1);
```

```
          with item.SubItems do
          begin
            Add(rec.Manufacturer);
            Add(rec.Model);
            Add(Format('%.2f', [rec.ClockMHz]));
            Add(IntToStr(rec.FlashKB));
            Add(IntToStr(rec.RAMKB));
            Add(IntToStr(rec.GPIOCount));
            Add(IntToStr(rec.ADCChannels));
            Add(IntToStr(rec.ADCResolutionBits));
            Add(IntToStr(rec.UARTCount));
            Add(rec.OtherInfo);
          end;
        end;
    finally
      lvOut.Items.EndUpdate;
      CloseFile(F);
    end;
  except
    CloseFile(F);
    raise;
  end;
end;

{ SplitToRuns: разбивает входной файл на отсортированные чанки (ран-файлы) в папке temp_ru
procedure TForm1.SplitToRuns(const InFile: string; fieldIndex: integer; out RunFiles: TStr
var
  FIn, FRun: TMCUFile;
  Buffer: array of TMicrocontroller;
  cnt, runIdx, i: integer;
  RunName, TempDir: string;
  tmpRec: TMicrocontroller;
  percent: integer;
begin
```

```
TempDir := GetTempDir;
RunFiles := TStringList.Create;
SetLength(Buffer, CHUNK_THRESHOLD);

AssignFile(FIn, InFile);
Reset(FIn);
runIdx := 0;

statBar.SimpleText := 'Фаза 1: Разбиение на ран-файлы...';
pbProgress.Position := 0;
Application.ProcessMessages;

try
  while not EOF(FIn) do
  begin
    cnt := 0;
    while (cnt < CHUNK_THRESHOLD) and not EOF(FIn) do
    begin
      Read(FIn, Buffer[cnt]);
      Inc(cnt);
    end;
    // Быстрая сортировка чанка
    QuickSortChunk(Buffer, 0, cnt - 1, fieldIndex);
    if Descending then
    begin
      // Разворачиваем для убывания
      for i := 0 to (cnt div 2) - 1 do
      begin
        tmpRec := Buffer[i];
        Buffer[i] := Buffer[cnt - 1 - i];
        Buffer[cnt - 1 - i] := tmpRec;
      end;
    end;
```

```
      // Запись ран-файла
      RunName := Format('%srun_%d.dat', [TempDir, runIdx]);
      AssignFile(FRun, RunName);
      Rewrite(FRun);
      for i := 0 to cnt - 1 do
        Write(FRun, Buffer[i]);
      CloseFile(FRun);

      RunFiles.Add(RunName);
      Inc(runIdx);

      percent := Min(100, (runIdx * CHUNK_THRESHOLD * 100) div TotalRecords);
      statBar.SimpleText := Format('Фаза 1: Создан ран-файл %d (%.0d%%)', [runIdx, percent]
      pbProgress.Position := percent;
      Application.ProcessMessages;
    end;
  finally
    CloseFile(FIn);
  end;

  statBar.SimpleText := Format('Фаза 1 завершена: создано %d ран-файлов', [runIdx]);
  pbProgress.Position := 0;
  Application.ProcessMessages;
end;

{ KWayMerge: сливает ран-файлы в один отсортированный, сразу в восх./убыв. порядке }
procedure TForm1.KWayMerge(const RunFiles: TStringList; fieldIndex: integer; const OutFile
var
  Files:    array of TMCUFile;
  Curr:     array of TMicrocontroller;
  HasRec:   array of Boolean;
  Heap:     array of TKWNode;
  heapSize, i, idx, l, r, chosen: integer;
  FOut: TMCUFile;
```

```
  node, tmp: TKWNode;
  totalRuns: integer;
  writtenCount, percent: Int64;

  { CompareDir: сравнение с учётом направления (min-heap vs max-heap) }
  function CompareDir(const A, B: TMicrocontroller): integer;
  begin
    if not Descending then
      Result := CompareRecords(A, B, fieldIndex)
    else
      Result := CompareRecords(B, A, fieldIndex);
  end;

begin
  totalRuns := RunFiles.Count;
  SetLength(Files, totalRuns);
  SetLength(Curr,  totalRuns);
  SetLength(HasRec, totalRuns);
  SetLength(Heap, totalRuns);

  // 1) Открываем все ран-файлы, читаем по одной записи
  statBar.SimpleText := 'Фаза 2: Открытие ран-файлов...';
  Application.ProcessMessages;
  for i := 0 to totalRuns - 1 do
  begin
    AssignFile(Files[i], RunFiles[i]);
    Reset(Files[i]);
    HasRec[i] := not EOF(Files[i]);
    if HasRec[i] then
      Read(Files[i], Curr[i]);
  end;

  // 2) Формируем начальную кучу из первых записей
  heapSize := 0;
```

```
for i := 0 to totalRuns - 1 do
  if HasRec[i] then
  begin
    Heap[heapSize].Rec := Curr[i];
    Heap[heapSize].SourceIdx := i;
    Inc(heapSize);
  end;

// 3) Heapify (приведение кучи к min-heap или max-heap)
for idx := (heapSize div 2) - 1 downto 0 do
begin
  l := 2 * idx + 1; r := 2 * idx + 2; chosen := idx;
  if (l < heapSize) and (CompareDir(Heap[l].Rec, Heap[chosen].Rec) < 0) then
    chosen := l;
  if (r < heapSize) and (CompareDir(Heap[r].Rec, Heap[chosen].Rec) < 0) then
    chosen := r;
  if chosen <> idx then
  begin
    tmp := Heap[idx]; Heap[idx] := Heap[chosen]; Heap[chosen] := tmp;
  end;
end;

// 4) Открываем итоговый файл
AssignFile(FOut, OutFile);
Rewrite(FOut);

writtenCount := 0;
pbProgress.Position := 0;
statBar.SimpleText := 'Фаза 2: Слияние ран-файлов...';
Application.ProcessMessages;

try
  // 5) Основной цикл: пока куча не пуста, извлекаем корень, записываем, читаем следующ
  //    восстанавливаем кучу
```

```
while heapSize > 0 do
begin
  node := Heap[0];
  Write(FOut, node.Rec);
  Inc(writtenCount);
  if writtenCount mod CHUNK_THRESHOLD = 0 then
  begin
    percent := Min(100, (writtenCount * 100) div TotalRecords);
    statBar.SimpleText := Format('Фаза 2: Объединено %d из %d записей (%.0d%%)',
      [writtenCount, TotalRecords, percent]);
    pbProgress.Position := percent;
    Application.ProcessMessages;
  end;

  idx := node.SourceIdx;
  if not EOF(Files[idx]) then
  begin
    Read(Files[idx], Curr[idx]);
    Heap[0].Rec := Curr[idx];
  end
  else
  begin
    Dec(heapSize);
    Heap[0] := Heap[heapSize];
  end;

  // 6) Восстановление heap-heap
  idx := 0;
  while True do
  begin
    l := 2 * idx + 1; r := 2 * idx + 2; chosen := idx;
    if (l < heapSize) and (CompareDir(Heap[l].Rec, Heap[chosen].Rec) < 0) then
      chosen := l;
    if (r < heapSize) and (CompareDir(Heap[r].Rec, Heap[chosen].Rec) < 0) then
```

```
          chosen := r;
        if chosen = idx then
          Break;
        tmp := Heap[idx]; Heap[idx] := Heap[chosen]; Heap[chosen] := tmp;
        idx := chosen;
      end;
    end;

    statBar.SimpleText := 'Фаза 2 завершена: все ран-файлы объединены';
    pbProgress.Position := 0;
    Application.ProcessMessages;
  finally
    // Закрываем и удаляем ран-файлы
    for i := 0 to High(Files) do
    begin
      CloseFile(Files[i]);
      SafeDeleteFile(RunFiles[i]);
    end;
    CloseFile(FOut);
  end;
end;

{ ExternalMergeSort: управляет внешней сортировкой: split→merge }
procedure TForm1.ExternalMergeSort(const InFile: string; fieldIndex: integer; Descending:
var
  Runs: TStringList;
  TempOut: string;
begin
  Runs := nil;
  try
    // Фаза 1: разбить исходный файл на отсортированные чанки
    SplitToRuns(InFile, fieldIndex, Runs, Descending);
    // Фаза 2: слияние чанков в один файл, сразу в нужном порядке
    TempOut := InFile + '.tmp';
```

```
    KWayMerge(Runs, fieldIndex, TempOut, Descending);
    // Переименовать временный файл в mcus.dat
    SafeDeleteFile(InFile);
    RenameFile(TempOut, InFile);
  finally
    Runs.Free;
  end;
end;

{ VerifySorted: проверяет, правильно ли отсортирован файл (по возр./убыв.) }
procedure TForm1.VerifySorted(const FileName: string; fieldIndex: integer; Descending: Boo
var
  F: TMCUFile;
  prevRec, currRec: TMicrocontroller;
  i: Int64;
  cmp: integer;
begin
  AssignFile(F, FileName);
  Reset(F);
  try
    if FileSize(F) < 2 then Exit;
    Read(F, prevRec);
    for i := 1 to FileSize(F) - 1 do
    begin
      Read(F, currRec);
      cmp := CompareRecords(prevRec, currRec, fieldIndex);
      if (not Descending and (cmp > 0)) or (Descending and (cmp < 0)) then
      begin
        statBar.SimpleText := 'ОШИБКА: Файл отсортирован неверно!';
        Exit;
      end;
      prevRec := currRec;
    end;
    statBar.SimpleText := 'Сортировка завершена: OK';
```

```
  finally
    CloseFile(F);
  end;
end;

end.
```