```
main.py:
import os
import json
from typing import Dict, List
import pandas as pd
from fastapi import FastAPI, HTTPException, Request, Response
from fastapi.responses import JSONResponse
from fastapi.staticfiles import StaticFiles
import uuid
import pickle
from fastapi.middleware.cors import CORSMiddleware

import tempfile
from io import BytesIO

import pickle  # <-- отсутствовал









BASE_DIR = os.path.dirname(os.path.abspath(__file__))
SETTINGS_PATH = os.path.join(BASE_DIR, "settings.json")
TEMPLATES_PATH = os.path.join(BASE_DIR, "formula_templates.json")
SIGNAL_INDEX_PATH = os.path.join(BASE_DIR, ".signal_index.pkl")

def load_templates() -> Dict:
    if not os.path.exists(TEMPLATES_PATH):
        return {"templates": []}
    with open(TEMPLATES_PATH, "r", encoding="utf-8") as f:
        return json.load(f)

def load_settings() -> Dict:
    with open(SETTINGS_PATH, "r", encoding="utf-8") as f:
        return json.load(f)

def load_signals_from_folder(folder: str) -> List[Dict]:
    # folder может быть относительным
    folder_abs = folder if os.path.isabs(folder) else
os.path.normpath(os.path.join(BASE_DIR, folder))
    if not os.path.isdir(folder_abs):
        raise FileNotFoundError(f"signalDataFolder not found: {folder_abs}")

    signals_map = {}  # Tagname -> Description (последний wins)
    for name in os.listdir(folder_abs):
        if not name.lower().endswith(".csv"):
            continue
        path = os.path.join(folder_abs, name)
        try:
            df = pd.read_csv(path, sep=';')[['Tagname', 'Description', 'Engineering
Unit']]
            df = df.dropna(subset=['Tagname'])
            for _, row in df.iterrows():
                tag = str(row['Tagname']).strip()
                desc = "" if pd.isna(row['Description']) else
str(row['Description']).strip()
                unit = "" if pd.isna(row['Engineering Unit']) else
str(row['Engineering Unit']).strip()
                desc = ", ".join([desc, unit])
                if tag:
                    signals_map[tag] = desc
        except Exception as e:
            # пропускаем "плохие" csv, но можно логировать
            print(f"[WARN] failed to read {path}: {e}")
```

```
    # в список
    out = [{'Tagname': k, 'Description': v} for k, v in signals_map.items()]
    out.sort(key=lambda x: x['Tagname'])
    return out



app = FastAPI()
app.add_middleware(
    CORSMiddleware,
    allow_origins=["http://localhost:8501"],
    allow_credentials=True,
    allow_methods=["*"],
    allow_headers=["*"],
)

# Кэш сигналов в памяти
STATE = {
    "settings": None,
    "signals": None,
    "signal_index": None
}


def build_signal_index(folder: str) -> Dict[str, List[str]]:
    """
    При запуске: проходим по всем CSV файлам и создаем индекс
    signal_name -> list of files where it's present
    """
    folder_abs = folder if os.path.isabs(folder) else os.path.normpath(
        os.path.join(BASE_DIR, folder)
    )

    if not os.path.isdir(folder_abs):
        raise FileNotFoundError(f"Signal data folder not found: {folder_abs}")

    signal_index = {}

    print(f"[INFO] Building signal index from {folder_abs}...")

    for filename in os.listdir(folder_abs):
        if not filename.lower().endswith(".csv"):
            continue

        filepath = os.path.join(folder_abs, filename)

        try:
            # Читаем только первую строку (заголовок)
            df_header = pd.read_csv(
                filepath,
                nrows=0,  # Читаем только заголовок
                encoding="ISO-8859-2",
                sep=";"
            )

            columns = df_header.columns.tolist()

            # Удаляем служебные столбцы из индекса
            signal_columns = [c for c in columns if c not in ["DATE", "TIME",
"datetime"]]

            for signal_name in signal_columns:
                if signal_name not in signal_index:
                    signal_index[signal_name] = []
```

```
                signal_index[signal_name].append(filepath)

            print(f"  ✓ {filename}: {len(signal_columns)} signals")

        except Exception as e:
            print(f"  ✗ Failed to index {filename}: {e}")
            continue

    print(f"[OK] Total unique signals indexed: {len(signal_index)}")

    # Сохраняем индекс в pickle для быстрого восстановления
    try:
        with open(SIGNAL_INDEX_PATH, "wb") as f:
            pickle.dump(signal_index, f)
        print(f"[OK] Signal index cached to {SIGNAL_INDEX_PATH}")
    except Exception as e:
        print(f"[WARN] Failed to cache signal index: {e}")

    return signal_index


def load_signal_index(folder: str) -> Dict[str, List[str]]:
    """
    Загружает индекс либо из кэша, либо перестраивает его
    """
    if os.path.exists(SIGNAL_INDEX_PATH):
        try:
            with open(SIGNAL_INDEX_PATH, "rb") as f:
                index = pickle.load(f)
            print(f"[OK] Signal index loaded from cache")
            return index
        except Exception as e:
            print(f"[WARN] Failed to load cached index: {e}")

    # Если кэша нет, перестраиваем
    return build_signal_index(folder)

def load_signal_data_optimized(signal_names: List[str], folder: str) -> Dict[str,
pd.DataFrame]:
    """
    Загружает только нужные сигналы из только нужных файлов
    Returns: {signal_name -> DataFrame}
    """
    folder_abs = folder if os.path.isabs(folder) else os.path.normpath(
        os.path.join(BASE_DIR, folder)
    )

    signal_index = STATE.get("signal_index", {})
    if not signal_index:
        raise RuntimeError("Signal index not initialized")

    signal_names_set = set(signal_names)
    found_signals = {}
    files_to_load = set()

    # Определяем, какие файлы нужно загружать
    for signal_name in signal_names_set:
        if signal_name in signal_index:
            files_to_load.update(signal_index[signal_name])

    print(f"[INFO] Loading {len(signal_names_set)} signals from {len(files_to_load)}
files")

    # Загружаем данные из файлов
    for filepath in files_to_load:
```

```
        try:
            df = pd.read_csv(
                filepath,
                encoding="ISO-8859-2",
                sep=";"
            )

            # Обработка даты/времени
            df["TIME"] = df["TIME"].str.replace(",", ".", regex=False)
            df["TIME"] = df["TIME"].str.split(".").str[0]
            combined = df["DATE"] + " " + df["TIME"]
            df["datetime"] = pd.to_datetime(
                combined, format="%d.%m.%Y %H:%M:%S", errors="coerce"
            )
            df = df.dropna(subset=["datetime"])
            df = df.drop(['DATE', 'TIME'], axis=1)

            # Сортируем по datetime
            df = df.sort_values("datetime")

            # Извлекаем только нужные сигналы
            available_columns = set(df.columns) & signal_names_set
            for signal_name in available_columns:
                if signal_name not in found_signals:
                    # Сохраняем datetime и значение сигнала
                    found_signals[signal_name] = df[["datetime", signal_name]].copy()
                    found_signals[signal_name].columns = ["datetime", "value"]

        except Exception as e:
            print(f"[WARN] Failed to read {filepath}: {e}")
            continue

    return found_signals




@app.on_event("startup")
def startup():
    settings = load_settings()
    STATE["settings"] = settings
    folder = settings.get("signalDataFolder")
    if not folder:
        raise RuntimeError("settings.json: signalDataFolder is required")
    STATE["signals"] = load_signals_from_folder(folder)
    STATE["templates"] = load_templates()
    STATE["signal_index"] = load_signal_index(settings.get("signalArchiveFolder"))

    print(f"[OK] loaded signals: {len(STATE['signals'])}")
    print(f"[OK] signal index has {len(STATE['signal_index'])} unique signals")
    print(f"[OK] loaded templates: {len(STATE['templates'].get('templates', []))}")

@app.get("/api/settings")
def api_settings():
    return STATE["settings"]

@app.get("/api/signals")
def api_signals(q: str = "", limit: int = 50):
    """
    q — маска со * (например *МАА*СР*)
    """
    signals = STATE["signals"] or []
    if not q:
```

```
254         return {"items": signals[:limit], "total": len(signals)}
255
256     # маска * -> regex
257     import re
258     escaped = re.escape(q).replace(r"\*", ".*")
259     rx = re.compile("^" + escaped + "$", re.IGNORECASE)
260
261     items = [s for s in signals if rx.match(s["Tagname"])]
262     return {"items": items[:max(1, min(limit, 500))], "total": len(items)}
263
264 # Helper: возвращает абсолютный путь к файлу проекта
265 def get_project_path(filename: str):
266     folder = STATE["settings"].get("projectDataFolder")
267     if not folder:
268         raise RuntimeError("projectDataFolder not configured")
269
270     # Нормализуем путь к папке проектов
271     project_dir = folder if os.path.isabs(folder) else
    os.path.normpath(os.path.join(BASE_DIR, folder))
272
273     # Проверяем, что filename безопасен (не пытается выйти за пределы папки)
274     if '..' in filename or '/' in filename or '\\' in filename:
275         raise HTTPException(status_code=400, detail="Invalid filename")
276
277     path = os.path.join(project_dir, filename)
278     # Проверяем, что итоговый путь лежит внутри разрешенной директории
279     if not path.startswith(project_dir):
280          raise HTTPException(status_code=400, detail="Path traversal attempt")
281
282     return path
283
284 @app.post("/api/project/save")
285 async def save_project(request: Request):
286     try:
287         data = await request.json()
288         filename = data.get("filename")
289         content = data.get("content")
290
291         if not filename or not content:
292             raise HTTPException(status_code=400, detail="Filename and content are
    required")
293
294         path = get_project_path(filename)
295
296         # Сохраняем как JSON
297         with open(path, "w", encoding="utf-8") as f:
298             json.dump(content, f, indent=2)
299
300         return {"status": "ok", "message": f"Project saved to {filename}"}
301
302     except HTTPException as e:
303         raise e
304     except Exception as e:
305         print(f"Error saving project: {e}")
306         raise HTTPException(status_code=500, detail="Internal server error during
    save")
307
308 @app.get("/api/project/load/{filename}")
309 def load_project(filename: str):
310     try:
311         path = get_project_path(filename)
312
313         if not os.path.exists(path):
314             raise HTTPException(status_code=404, detail="Project not found")
315
```

```
316          with open(path, "r", encoding="utf-8") as f:
317              content = json.load(f)
318
319          return content
320
321      except HTTPException as e:
322          raise e
323      except Exception as e:
324          print(f"Error loading project: {e}")
325          raise HTTPException(status_code=500, detail="Internal server error during
     load")
326
327  @app.get("/api/formula-templates")
328  def api_formula_templates():
329      return STATE.get("templates") or {"templates": []}
330
331  @app.get("/api/project/list")
332  def list_projects():
333      folder = STATE["settings"].get("projectDataFolder")
334      if not folder:
335          raise HTTPException(status_code=500, detail="Project folder not configured")
336
337      project_dir = folder if os.path.isabs(folder) else
     os.path.normpath(os.path.join(BASE_DIR, folder))
338      os.makedirs(project_dir, exist_ok=True)
339
340      projects = []
341      for fname in sorted(os.listdir(project_dir)):
342          if not fname.endswith(".json"):
343              continue
344          path = os.path.join(project_dir, fname)
345          try:
346              with open(path, "r", encoding="utf-8") as f:
347                  payload = json.load(f)
348          except Exception:
349              continue
350          project_meta = payload.get("project", {})
351          projects.append({
352              "filename": fname,
353              "code": project_meta.get("code") or project_meta.get("tagname") or "",
354              "description": project_meta.get("description") or "",
355              "type": project_meta.get("type") or ""
356          })
357      return {"projects": projects}
358
359  @app.post("/api/signal-data")
360  async def api_signal_data(request: Request):
361      """
362      POST с JSON телом:
363      {
364          "signal_names": ["SIGNAL1", "SIGNAL2", ...],
365          "format": "parquet"  # или "json"
366      }
367
368      Returns: Parquet файл с данными или JSON
369      """
370      try:
371          data = await request.json()
372          signal_names = data.get("signal_names", [])
373          output_format = data.get("format", "parquet")  # По умолчанию Parquet
374
375          if not signal_names:
376              raise HTTPException(status_code=400, detail="signal_names is required")
377
378          folder = STATE["settings"].get("signalArchiveFolder")
```

```
379         if not folder:
380             raise HTTPException(status_code=500, detail="signalArchiveFolder not
    configured")
381
382         # Загружаем данные сигналов
383         signals_data = load_signal_data_optimized(signal_names, folder)
384
385         # Подготавливаем ответ
386         response = {
387             "found": list(signals_data.keys()),
388             "not_found": [s for s in signal_names if s not in signals_data],
389             "format": output_format
390         }
391
392         if not signals_data:
393             raise HTTPException(status_code=404, detail="No signals found")
394
395         # Экспортируем данные в зависимости от формата
396         if output_format == "parquet":
397             return await _export_parquet(signals_data, response)
398         else:
399             return await _export_json(signals_data, response)
400
401     except HTTPException as e:
402         raise e
403     except Exception as e:
404         print(f"Error in api_signal_data: {e}")
405         raise HTTPException(status_code=500, detail=str(e))
406
407 async def _export_parquet(signals_data: Dict[str, pd.DataFrame], meta: Dict):
408     """
409     Экспортирует данные в Parquet (НАМНОГО меньше чем JSON!)
410     """
411     from fastapi.responses import FileResponse
412
413     try:
414         # Создаем временный файл
415         with tempfile.NamedTemporaryFile(suffix=".parquet", delete=False) as tmp:
416             tmp_path = tmp.name
417
418         # Каждый сигнал сохраняем как отдельную таблицу в Parquet
419         # Используем структуру: datetime, signal_name, value
420         rows = []
421         for signal_name, df in signals_data.items():
422             df_copy = df.copy()
423             df_copy["signal_name"] = signal_name
424             rows.append(df_copy)
425
426         combined = pd.concat(rows, ignore_index=True)
427         combined.to_parquet(tmp_path, compression='snappy', index=False)
428
429         file_size = os.path.getsize(tmp_path)
430         print(f"[OK] Exported {len(signals_data)} signals to Parquet: {file_size /
    1024 / 1024:.2f} MB")
431
432         return FileResponse(
433             tmp_path,
434             media_type="application/octet-stream",
435             filename="signal_data.parquet",
436             headers={"X-Signal-Meta": json.dumps(meta)}
437         )
438
439     except Exception as e:
440         print(f"[ERROR] Parquet export failed: {e}")
441         raise
```

```
442
443
444 async def _export_json(signals_data: Dict[str, pd.DataFrame], meta: Dict):
445     """
446     Экспортирует данные в JSON (медленнее и больше, но совместимее)
447     """
448     from fastapi.responses import JSONResponse
449
450     try:
451         # Формируем JSON с каждым сигналом отдельно
452         data_dict = {}
453         for signal_name, df in signals_data.items():
454             df_copy = df.copy()
455             df_copy["datetime"] = df_copy["datetime"].astype(str)
456             data_dict[signal_name] = df_copy.to_dict(orient="records")
457
458         response_data = {
459             **meta,
460             "data": data_dict
461         }
462
463         return JSONResponse(response_data)
464
465     except Exception as e:
466         print(f"[ERROR] JSON export failed: {e}")
467         raise
468
469 # Простое файловое хранилище для сессий визуализации
470 VIS_SESSIONS_DIR = os.path.join(tempfile.gettempdir(), "viz_sessions")
471 os.makedirs(VIS_SESSIONS_DIR, exist_ok=True)
472
473 def _save_viz_session(data: Dict) -> str:
474     token = uuid.uuid4().hex
475     path = os.path.join(VIS_SESSIONS_DIR, f"{token}.pkl")
476     with open(path, "wb") as f:
477         pickle.dump(data, f)
478     return token
479
480 def _load_viz_session(token: str) -> Dict:
481     path = os.path.join(VIS_SESSIONS_DIR, f"{token}.pkl")
482     if not os.path.exists(path):
483         return None
484     with open(path, "rb") as f:
485         return pickle.load(f)
486
487 @app.post("/api/visualize/session")
488 async def create_visualize_session(payload: Dict):
489     signals = payload.get("signals", [])
490     code = payload.get("code", "")
491     if not isinstance(signals, list):
492         raise HTTPException(status_code=400, detail="signals must be a list")
493     token = _save_viz_session({"signals": signals, "code": code})
494     return {"token": token}
495
496 @app.get("/api/visualize/session/{token}")
497 def get_visualize_session(token: str):
498     data = _load_viz_session(token)
499     if not data:
500         raise HTTPException(status_code=404, detail="session not found")
501     return data
502
503
504 # Раздаём фронтенд
505 WEB_DIR = os.path.normpath(os.path.join(BASE_DIR, "..", "web"))
506 app.mount("/", StaticFiles(directory=WEB_DIR, html=True), name="web")
```

```

code_signal.py:
import re
from typing import List, Tuple, Dict

import numpy as np
import pandas as pd


class CodeEvaluationError(Exception):
    """Ошибка во время вычисления выражения CODE."""


def sanitize_numeric_column(series: pd.Series) -> pd.Series:
    if series.dtype.kind in ("i", "u", "f"):
        return series
    text = series.astype(str).str.replace(",", ".", regex=False)
    return pd.to_numeric(text, errors="coerce")


def evaluate_code_expression(code_str: str, df_all: pd.DataFrame) -> Tuple[pd.Series, List[str]]:
    if df_all is None or df_all.empty:
        raise CodeEvaluationError("Нет данных для расчёта синтетического сигнала.")
    if not code_str or not code_str.strip():
        raise CodeEvaluationError("Строка CODE пуста.")

    index = df_all.index
    numeric_df = df_all.apply(sanitize_numeric_column)
    series_map = {col: numeric_df[col] for col in numeric_df.columns}
    warnings: List[str] = []

    # ---------- обработка «неправильных» имён сигналов ----------
    safe_name_map: Dict[str, str] = {}
    used_safe_names = set()

    def _make_safe_name(original: str, idx: int) -> str:
        base = re.sub(r"\W", "_", original)
        if not base or not re.match(r"[A-Za-z_]", base):
            base = f"SIG_{idx}"
        while base in used_safe_names:
            base += "_"
        used_safe_names.add(base)
        return base

    sorted_signals = sorted(series_map.keys(), key=len, reverse=True)
    for idx, sig_name in enumerate(sorted_signals):
        safe = _make_safe_name(sig_name, idx)
        safe_name_map[sig_name] = safe

    def _replace_signal_names(expr: str) -> str:
        result = []
        i = 0
        in_string = False
        string_char = ""

        while i < len(expr):
            ch = expr[i]
            if in_string:
                result.append(ch)
                if ch == string_char and expr[i - 1] != "\\":
                    in_string = False
                i += 1
                continue

```

```
            if ch in ("'", '"'):
                in_string = True
                string_char = ch
                result.append(ch)
                i += 1
                continue

            matched = None
            for name in sorted_signals:
                if expr.startswith(name, i):
                    matched = name
                    break
            if matched:
                result.append(safe_name_map[matched])
                i += len(matched)
            else:
                result.append(ch)
                i += 1

        return "".join(result)

    # ---------- вспомогательные функции ----------
    def _ensure_series(value) -> pd.Series:
        if isinstance(value, pd.Series):
            return value.reindex(index)
        if isinstance(value, pd.DataFrame):
            if value.shape[1] == 1:
                return value.iloc[:, 0].reindex(index)
            raise CodeEvaluationError("Невозможно привести DataFrame с несколькими колонками к Series.")
        if isinstance(value, (list, tuple, np.ndarray)):
            arr = np.asarray(value, dtype=float)
            if arr.size == 1:
                arr = np.full(len(index), arr.item())
            elif arr.shape[0] != len(index):
                return pd.Series(np.nan, index=index)
            return pd.Series(arr, index=index)
        if value is None or np.isscalar(value):
            return pd.Series(value, index=index)
        try:
            return pd.Series(value, index=index)
        except Exception as exc:
            raise CodeEvaluationError(f"Невозможно преобразовать значение '{value}' к Series.") from exc

    def _aggregate_nanfunc(func, args, empty_value=np.nan):
        if not args:
            return pd.Series(empty_value, index=index)
        stacked = np.vstack([_ensure_series(arg).values for arg in args])
        return pd.Series(func(stacked, axis=0), index=index)

    def GETPOINT(*_):
        if "GETPOINT" not in warnings:
            warnings.append("GETPOINT пока не поддержан — возвращается NaN.")
        return pd.Series(np.nan, index=index)

    def PREV(param_name):
        if param_name not in series_map:
            return pd.Series(np.nan, index=index)
        return series_map[param_name].shift(1)

    def _history_series(param_name):
        if param_name not in series_map:
            return None
        return series_map[param_name]
```

```
634
635      def _history_window(period):
636          try:
637              minutes = int(period)
638          except (TypeError, ValueError):
639              return None
640          if minutes <= 0:
641              return None
642          return f"{minutes}min"
643
644      def _history_apply(param_name, period, fn):
645          s = _history_series(param_name)
646          window = _history_window(period)
647          if s is None or window is None:
648              return pd.Series(np.nan, index=index)
649          return fn(s.rolling(window))
650
651      HISTORYAVG = lambda n, p: _history_apply(n, p, lambda r: r.mean())
652      HISTORYCOUNT = lambda n, p: _history_apply(n, p, lambda r: r.count())
653      HISTORYSUM = lambda n, p: _history_apply(n, p, lambda r: r.sum())
654      HISTORYMAX = lambda n, p: _history_apply(n, p, lambda r: r.max())
655      HISTORYMIN = lambda n, p: _history_apply(n, p, lambda r: r.min())
656      HISTORYDIFF = lambda n, p: _history_apply(n, p, lambda r: r.max() - r.min())
657
658      def HISTORYGRADIENT(param_name, period):
659          s = _history_series(param_name)
660          window = _history_window(period)
661          if s is None or window is None:
662              return pd.Series(np.nan, index=index)
663
664          def slope(window_series: pd.Series):
665              valid = window_series.dropna()
666              if len(valid) < 2:
667                  return np.nan
668              x = valid.index.view(np.int64).astype(float) / 1e9
669              y = valid.values.astype(float)
670              x_mean = x.mean()
671              y_mean = y.mean()
672              denom = np.sum((x - x_mean) ** 2)
673              if denom == 0:
674                  return np.nan
675              return np.sum((x - x_mean) * (y - y_mean)) / denom
676
677          return s.rolling(window).apply(slope, raw=False)
678
679      def ROUND(a, b=0):
680          a_values = _ensure_series(a).values
681          b_values = _ensure_series(b).values
682          decimals = [
683              0 if np.isnan(dec) else int(round(dec))
684              for dec in b_values
685          ]
686          rounded = np.array([
687              np.round(val, dec) if not np.isnan(val) else np.nan
688              for val, dec in zip(a_values, decimals)
689          ])
690          return pd.Series(rounded, index=index)
691
692      # ---------- окружение eval ----------
693      env = {
694          "np": np,
695          "ABS": lambda a: pd.Series(np.abs(_ensure_series(a).values), index=index),
696          "POW": lambda a, b: pd.Series(np.power(_ensure_series(a).values,
     _ensure_series(b).values), index=index),
697          "MIN": lambda *args: _aggregate_nanfunc(np.nanmin, args),
```

```
        "MAX": lambda *args: _aggregate_nanfunc(np.nanmax, args),
        "AVG": lambda *args: _aggregate_nanfunc(np.nanmean, args, empty_value=0.0),
        "MED": lambda *args: _aggregate_nanfunc(np.nanmedian, args),
        "ROUND": ROUND,
        "WHEN": lambda cond, t_val, f_val: pd.Series(
            np.where(_ensure_series(cond).astype(bool).values,
                     _ensure_series(t_val).values,
                     _ensure_series(f_val).values),
            index=index,
        ),
        "PREV": PREV,
        "HISTORYAVG": HISTORYAVG,
        "HISTORYCOUNT": HISTORYCOUNT,
        "HISTORYSUM": HISTORYSUM,
        "HISTORYMAX": HISTORYMAX,
        "HISTORYMIN": HISTORYMIN,
        "HISTORYDIFF": HISTORYDIFF,
        "HISTORYGRADIENT": HISTORYGRADIENT,
        "GETPOINT": GETPOINT,
    }

    for original_name, safe_name in safe_name_map.items():
        env[safe_name] = series_map[original_name]

    def _normalize_expression(expr: str) -> str:
        expr = re.sub(r"\bAND\b", "&", expr, flags=re.IGNORECASE)
        expr = re.sub(r"\bOR\b", "|", expr, flags=re.IGNORECASE)
        expr = re.sub(r"\bNOT\b", "~", expr, flags=re.IGNORECASE)
        expr = expr.replace("<>", "!=")
        expr = re.sub(r"(?<![<>=!])=(?![<>=])", "==", expr)
        return expr

    normalized_code = _normalize_expression(code_str)
    normalized_code = _replace_signal_names(normalized_code)

    try:
        raw_result = eval(normalized_code, {"__builtins__": {}}, env)
    except Exception as exc:
        raise CodeEvaluationError(str(exc)) from exc

    result_series = _ensure_series(raw_result)
    result_series.name = result_series.name or "CODE_RESULT"
    return result_series, warnings

def compute_code_signal(
    code_str: str,
    df_all: pd.DataFrame,
    warn_callback=lambda msg: None,
) -> pd.Series:
    """
    Совместимость с визуализатором: считает синтетический сигнал по CODE
    и прокидывает предупреждения через колбэк.
    """
    series, warnings = evaluate_code_expression(code_str, df_all)
    for message in warnings:
        warn_callback(message)
    return series

    visualizer_app.py:
    import pandas as pd
import requests
import streamlit as st
import plotly.express as px
import numpy as np
import plotly.graph_objects as go
```

```
763
764 from code_signal import compute_code_signal, sanitize_numeric_column
765
766 st.set_page_config(page_title="Signal Visualizer", layout="wide")
767 st.title("📊 Визуализация сигналов")
768
769 query_params = st.query_params
770 session_token = query_params.get("session", None)
771 api_url = query_params.get("api_url", "http://localhost:8000")
772
773 signal_codes = query_params.get("signals", [])
774 if isinstance(signal_codes, str):
775     signal_codes = [signal_codes]
776
777 CODE = ""
778 if session_token:
779     try:
780         resp = requests.get(f"{api_url}/api/visualize/session/{session_token}")
781         resp.raise_for_status()
782         payload = resp.json()
783         signal_codes = payload.get("signals", signal_codes)
784         CODE = payload.get("code", CODE)
785     except Exception as e:
786         st.error(f"Не удалось получить данные сессии: {e}")
787
788 if "signals_data" not in st.session_state:
789     st.session_state.signals_data = None
790 if "selected_signals" not in st.session_state:
791     st.session_state.selected_signals = set()
792 if "plot_areas" not in st.session_state:
793     st.session_state.plot_areas = []
794 if "derived_signals" not in st.session_state:
795     st.session_state.derived_signals = {}
796 if "code_signal_name" not in st.session_state:
797     st.session_state.code_signal_name = None
798
799
800 def load_signals(signal_codes_list):
801     if not signal_codes_list:
802         st.info("Список сигналов пуст — ничего загружать.")
803         return None, [], []
804     try:
805         response = requests.post(
806             f"{api_url}/api/signal-data",
807             json={"signal_names": signal_codes_list, "format": "json"},
808         )
809         response.raise_for_status()
810         result = response.json()
811         found = result.get("found", [])
812         not_found = result.get("not_found", [])
813         data_dict = result.get("data", {})
814
815         if not data_dict:
816             st.warning("Нет данных по запрошенным сигналам.")
817             return None, found, not_found
818
819         frames = []
820         for sig, records in data_dict.items():
821             if not records:
822                 continue
823             df = pd.DataFrame(records)
824             if "datetime" not in df or "value" not in df:
825                 continue
826             df["datetime"] = pd.to_datetime(df["datetime"], errors="coerce")
827             df = df.dropna(subset=["datetime"])
```

```
            df = df.set_index("datetime").sort_index()
            df = df.rename(columns={"value": sig})
            frames.append(df[[sig]])

        if not frames:
            return None, found, not_found
        return pd.concat(frames, axis=1).sort_index(), found, not_found

    except Exception as exc:
        st.error(f"❌ Ошибка загрузки данных: {exc}")
        return None, [], []


def get_all_signals_df(exclude: set[str] | None = None):
    exclude = exclude or set()
    base = st.session_state.signals_data
    derived = st.session_state.derived_signals

    dfs = []
    if base is not None:
        dfs.append(base)
    for name, ddf in derived.items():
        if name in exclude:
            continue
        dfs.append(ddf)

    if not dfs:
        return None
    return pd.concat(dfs, axis=1).sort_index()


def compute_stats_numeric(df: pd.DataFrame) -> pd.DataFrame:
    if df is None or df.empty:
        return pd.DataFrame()

    numeric = df.apply(sanitize_numeric_column)
    valid_cols = [col for col in numeric.columns if numeric[col].count() > 0]
    if not valid_cols:
        return pd.DataFrame()

    numeric = numeric[valid_cols]
    stats = pd.DataFrame(index=numeric.columns)
    stats["count"] = numeric.count()
    stats["min"] = numeric.min()
    stats["max"] = numeric.max()
    stats["mean"] = numeric.mean()
    stats["std"] = numeric.std()
    stats["median"] = numeric.median()

    starts, ends = [], []
    for col in numeric.columns:
        series = numeric[col].dropna()
        starts.append(series.index.min() if not series.empty else pd.NaT)
        ends.append(series.index.max() if not series.empty else pd.NaT)

    stats["start"] = starts
    stats["end"] = ends
    return stats


def make_unique_name(base_name: str) -> str:
    existing = set()
    if st.session_state.signals_data is not None:
        existing |= set(st.session_state.signals_data.columns)
    existing |= set(st.session_state.derived_signals.keys())
```

```
    if base_name not in existing:
        return base_name
    idx = 2
    while f"{base_name}_{idx}" in existing:
        idx += 1
    return f"{base_name}_{idx}"


if signal_codes and st.session_state.signals_data is None:
    with st.spinner("Загружаем данные сигналов..."):
        df_base, found_codes, not_found_codes = load_signals(signal_codes)
        st.session_state.signals_data = df_base
        st.success(f"✅ Загружено сигналов: {len(found_codes)}")
        if not_found_codes:
            st.warning(f"⚠️ Не найдены: {', '.join(not_found_codes)}")

# --- синтетический сигнал из CODE (считаем один раз, потом не пересчитываем) ---
code_signal_name = st.session_state.code_signal_name
df_for_code = get_all_signals_df(exclude={code_signal_name} if code_signal_name else None)

# Ключ "какой CODE мы уже считали" (можно оставить просто CODE; session_token добавил на всякий)
code_key = (session_token, CODE)

already_have_series = (
    st.session_state.code_signal_name is not None
    and st.session_state.code_signal_name in st.session_state.derived_signals
)

if CODE and df_for_code is not None:
    need_recalc = (st.session_state.get("code_key") != code_key) or (not already_have_series)

    if need_recalc:
        try:
            synthetic_series = compute_code_signal(
                CODE,
                df_for_code,
                warn_callback=lambda msg: st.warning(msg, icon="⚠️"),
            )
            target_name = code_signal_name or make_unique_name("CODE_RESULT")
            synthetic_series.name = target_name

            st.session_state.derived_signals[target_name] = pd.DataFrame({target_name: synthetic_series})
            st.session_state.code_signal_name = target_name
            st.session_state.selected_signals.add(target_name)

            st.session_state.code_key = code_key
            st.success(f"Синтетический сигнал обновлён: {target_name}")
        except Exception as exc:
            st.warning(f"Не удалось вычислить CODE: {exc}")

elif not CODE:
    # если CODE исчез — удаляем синтетический сигнал и сбрасываем ключ
    if code_signal_name:
        st.session_state.derived_signals.pop(code_signal_name, None)
        st.session_state.selected_signals.discard(code_signal_name)
        st.session_state.code_signal_name = None
    st.session_state.code_key = None

# --- итоговый DataFrame со всеми сигналами ---
df_all_signals = get_all_signals_df()

```

```
with st.sidebar:
    st.header("Выбор сигналов")

    if df_all_signals is not None:
        available_signals = df_all_signals.columns.tolist()
        for signal in available_signals:
            checked = st.checkbox(
                signal,
                value=(signal in st.session_state.selected_signals),
            )
            if checked:
                st.session_state.selected_signals.add(signal)
            else:
                st.session_state.selected_signals.discard(signal)

        st.divider()
        st.subheader("Создать обрезанный сигнал")

        base_df = st.session_state.signals_data
        if base_df is not None and not base_df.empty:
            base_choice = st.selectbox("Исходный сигнал", base_df.columns)
            series = base_df[base_choice].dropna()
            if not series.empty:
                col1, col2 = st.columns(2)
                with col1:
                    start_date = st.date_input(
                        "Начало",
                        value=series.index.min().date(),
                    )
                with col2:
                    end_date = st.date_input(
                        "Конец",
                        value=series.index.max().date(),
                    )

                start_ts = pd.Timestamp(start_date)
                end_ts = pd.Timestamp(end_date) + pd.Timedelta(days=1) - pd.Timedelta(
                    microseconds=1
                )

                default_name = f"{base_choice}__{start_ts.date()}_{end_ts.date()}"
                new_name = st.text_input("Имя нового сигнала", value=default_name)

                col3, col4 = st.columns(2)
                if col3.button("Создать"):
                    name_unique = make_unique_name(new_name.strip())
                    cut_series = series[(series.index >= start_ts) & (series.index <=
end_ts)]
                    if cut_series.empty:
                        st.warning("В выбранном диапазоне нет точек.")
                    else:
                        st.session_state.derived_signals[name_unique] = pd.DataFrame(
                            {name_unique: cut_series}
                        )
                        st.success(f"Создан обрезанный сигнал: {name_unique}")
                        st.rerun()
                if col4.button("Очистить все обрезанные"):
                    st.session_state.derived_signals = {
                        k: v
                        for k, v in st.session_state.derived_signals.items()
                        if k == st.session_state.code_signal_name
                    }
                    st.session_state.selected_signals = {
                        sig
                        for sig in st.session_state.selected_signals
```

```
                        if (st.session_state.signals_data is not None and sig in
    st.session_state.signals_data.columns)
                        or sig == st.session_state.code_signal_name
                    }
                    st.rerun()

        if st.session_state.derived_signals:
            st.subheader("Удалить обрезанный/синтетический сигнал")
            derived_names = [name for name in st.session_state.derived_signals.keys()]
            delete_candidate = st.selectbox("Выберите", ["—"] + derived_names)
            if st.button("Удалить выбранный") and delete_candidate != "—":
                st.session_state.derived_signals.pop(delete_candidate, None)
                st.session_state.selected_signals.discard(delete_candidate)
                if delete_candidate == st.session_state.code_signal_name:
                    st.session_state.code_signal_name = None
                st.rerun()

        st.divider()
        st.subheader("Области построения")
        col_a, col_b = st.columns(2)
        if col_a.button("➕ Добавить график"):
            new_id = max([area.get("id", 0) for area in st.session_state.plot_areas] +
    [0]) + 1
            st.session_state.plot_areas.append({"id": new_id, "signals": []})
            st.rerun()
        if col_b.button("❌ Очистить все"):
            st.session_state.plot_areas = []
            st.session_state.selected_signals = set()
            st.rerun()
    else:
        st.info("📥 Данные сигналов еще не загружены.")

if df_all_signals is not None and st.session_state.selected_signals:
    if not st.session_state.plot_areas:
        st.session_state.plot_areas.append(
            {"id": 1, "signals": list(st.session_state.selected_signals)}
        )

    for i, plot_area in enumerate(st.session_state.plot_areas):
        with st.container():
            col1, col2 = st.columns([3, 1])
            with col1:
                st.subheader(f"График #{plot_area['id']}")
            with col2:
                if st.button("Удалить", key=f"remove_area_{i}"):
                    st.session_state.plot_areas.pop(i)
                    st.rerun()

            selected = st.multiselect(
                "Выберите сигнал(ы):",
                list(st.session_state.selected_signals),
                default=plot_area.get("signals", []),
                key=f"signals_sel_{i}",
            )
            st.session_state.plot_areas[i]["signals"] = selected

            if selected:
                df_plot = df_all_signals[selected].copy()

                # Для графика приводим к числам (поддержка запятых)
                df_plot_num = df_plot.apply(sanitize_numeric_column)

                valid_index = df_plot_num.dropna(how="all").index
                if len(valid_index) == 0:
                    st.warning("Нет числовых данных для выбранных сигналов.")
```

```
                else:
                    ts_idx = st.slider(
                        "Вертикальная линия (время)",
                        min_value=0,
                        max_value=len(valid_index) - 1,
                        value=len(valid_index) - 1,
                        key=f"vline_{i}",
                    )
                    ts = valid_index[ts_idx]

                    # график с вертикальной линией
                    fig = px.line(
                        df_plot_num,
                        x=df_plot_num.index,
                        y=selected,
                        title=f"График #{plot_area['id']}",
                        render_mode="webgl"
                    )
                    fig.add_vline(x=ts, line_width=2, line_dash="dash",
    line_color="red")
                    fig.update_layout(
                        uirevision=f"plot_area_{plot_area['id']}",
                        height=650,
                        legend_title_text="Сигналы",
                        xaxis_title="Время",
                        yaxis_title="Значение",
                        margin=dict(l=20, r=20, t=40, b=20),
                    )
                    st.plotly_chart(fig, use_container_width=True)

                    # значения на линии
                    nearest =
    df_plot_num.reindex(df_plot_num.index.union([ts])).sort_index()
                    nearest = nearest.ffill().loc[ts]

                    # статистика + колонка значений на линии
                    st.markdown("**📊 Статистика (по всему сигналу):**")
                    stats_df = compute_stats_numeric(df_plot)
                    if stats_df.empty:
                        st.info("Нет числовых данных для расчёта статистики.")
                    else:
                        stats_view = stats_df.copy()
                        stats_view["value"] = nearest.reindex(stats_view.index)
                        stats_view["start"] = (
                            pd.to_datetime(stats_view["start"], errors="coerce")
                            .dt.strftime("%Y-%m-%d %H:%M:%S")
                        )
                        stats_view["end"] = (
                            pd.to_datetime(stats_view["end"], errors="coerce")
                            .dt.strftime("%Y-%m-%d %H:%M:%S")
                        )
                        st.dataframe(
                            stats_view.style.format(
                                {
                                    "count": "{:.0f}",
                                    "min": "{:.6g}",
                                    "max": "{:.6g}",
                                    "mean": "{:.6g}",
                                    "std": "{:.6g}",
                                    "median": "{:.6g}",
                                    "value_at_line": "{:.6g}",
                                },
                                na_rep="",
                            ),
                            use_container_width=True,
```

```
                        )
            else:
                st.info("Выберите сигналы для отображения.")
        st.divider()

elif df_all_signals is None:
    st.info("📥 Данные сигналов ещё не загружены.")
else:
    st.info("👈 Выберите сигналы слева для визуализации.")

if df_all_signals is not None:
    with st.expander("ℹ️ Информация о данных"):
        col1, col2, col3 = st.columns(3)
        with col1:
            st.metric("Всего сигналов (вкл. обрезанные/синтет.)",
len(df_all_signals.columns))
        with col2:
            st.metric("Количество записей", len(df_all_signals))
        with col3:
            try:
                dt_range = df_all_signals.index.max() - df_all_signals.index.min()
                st.metric("Диапазон времени", str(dt_range).split(".")[0])
            except Exception:
                st.metric("Диапазон времени", "—")

if CODE:
    with st.expander("🧩 Сгенерированный код (оригинал)"):
        st.code(CODE, language="text")

index.html:
<!DOCTYPE html>
<html lang="ru">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Редактор логических схем</title>
    <link rel="stylesheet" href="css/styles.css">
</head>
<body>
    <div id="app">
        <div id="menu">
            <button class="menu-btn" id="btn-new">📄 Новый</button>
            <button class="menu-btn" id="btn-save">💾 Сохранить</button>
            <button class="menu-btn" id="btn-load">📂 Загрузить</button>
            <button class="menu-btn" id="btn-generate-code">🧩 Код</button>
            <button class="menu-btn" id="btn-project-settings">⚙️ Свойства проекта</
button>
            <button class="menu-btn" id="btn-visualize">📈 Визуализировать</button>
            <div class="menu-separator"></div>
            <div class="zoom-controls">
                <button class="menu-btn zoom-btn" id="btn-zoom-out">−</button>
                <span id="zoom-level">100%</span>
                <button class="menu-btn zoom-btn" id="btn-zoom-in">+</button>
                <button class="menu-btn" id="btn-zoom-fit">⊡ Вписать</button>
                <button class="menu-btn" id="btn-zoom-reset">1:1</button>
            </div>
            <input type="file" id="file-input" accept=".json">
        </div>

        <div id="main">
            <div id="palette">
                <h3>📦 Элементы</h3>
                <div class="palette-section">
                    <div class="palette-section-title">ВИЗУАЛЬНОЕ</div>

```

```
                    <div class="palette-item" data-type="group">
                        <svg viewBox="0 0 60 40">
                        <rect x="6" y="8" width="48" height="24" rx="4"
                                fill="none" stroke="#6b7280" stroke-width="2" stroke-dasharray="4,2"/>
                        <text x="14" y="25" fill="#6b7280" font-size="10" font-weight="bold">GROUP</text>
                        </svg>
                        <div class="palette-item-name">Группа</div>
                    </div>
                    </div>

                <div class="palette-section">
                    <div class="palette-section-title">ВХОДЫ</div>

                    <div class="palette-item" data-type="input-signal">
                        <svg viewBox="0 0 60 40">
                            <polygon points="0,5 40,5 55,20 40,35 0,35" fill="#0f3460" stroke="#4a90d9" stroke-width="2"/>
                            <text x="12" y="24" fill="#eee" font-size="10">IN</text>
                        </svg>
                        <div class="palette-item-name">Входной сигнал</div>
                    </div>
                </div>
                <div class="palette-section">
                    <div class="palette-section-title">ВЫХОДЫ</div>

                    <div class="palette-item" data-type="output">
                        <svg viewBox="0 0 60 40">
                            <rect x="5" y="5" width="50" height="30" rx="6" fill="none" stroke="#10b981" stroke-width="2" stroke-dasharray="4,2"/>
                            <text x="12" y="24" fill="#10b981" font-size="9">ВЫХОД</text>
                        </svg>
                        <div class="palette-item-name">Выход</div>
                    </div>
                </div>

                <div class="palette-section">
                    <div class="palette-section-title">ЛОГИЧЕСКИЕ</div>

                    <div class="palette-item" data-type="and">
                        <svg viewBox="0 0 60 40">
                            <rect x="5" y="5" width="50" height="30" rx="5" fill="#0f3460" stroke="#a855f7" stroke-width="2"/>
                            <text x="22" y="25" fill="#eee" font-size="12" font-weight="bold">И</text>
                        </svg>
                        <div class="palette-item-name">И (AND)</div>
                    </div>

                    <div class="palette-item" data-type="or">
                        <svg viewBox="0 0 60 40">
                            <rect x="5" y="5" width="50" height="30" rx="5" fill="#0f3460" stroke="#a855f7" stroke-width="2"/>
                            <text x="12" y="25" fill="#eee" font-size="11" font-weight="bold">ИЛИ</text>
                        </svg>
                        <div class="palette-item-name">ИЛИ (OR)</div>
                    </div>

                    <div class="palette-item" data-type="not">
                        <svg viewBox="0 0 60 40">
                            <rect x="5" y="5" width="50" height="30" rx="5" fill="#0f3460" stroke="#a855f7" stroke-width="2"/>
```

```
                            <text x="12" y="25" fill="#eee" font-size="11" font-weight="bold">НЕТ</text>
                        </svg>
                        <div class="palette-item-name">НЕТ (NOT)</div>
                    </div>
                </div>

                <div class="palette-section">
                    <div class="palette-section-title">СРАВНЕНИЕ</div>

                    <div class="palette-item" data-type="if">
                        <svg viewBox="0 0 60 40">
                            <polygon points="30,3 57,20 30,37 3,20" fill="#0f3460" stroke="#e94560" stroke-width="2"/>
                            <text x="14" y="24" fill="#eee" font-size="9" font-weight="bold">ЕСЛИ</text>
                        </svg>
                        <div class="palette-item-name">ЕСЛИ (IF)</div>
                    </div>
                </div>

                <div class="palette-section">
                    <div class="palette-section-title">РАЗВЕТВЛЕНИЕ</div>

                    <div class="palette-item" data-type="separator">
                        <svg viewBox="0 0 60 40">
                            <rect x="5" y="8" width="50" height="24" rx="3" fill="#0f3460" stroke="#f59e0b" stroke-width="2"/>
                            <text x="8" y="25" fill="#f59e0b" font-size="10" font-weight="bold">✓/✗</text>
                        </svg>
                        <div class="palette-item-name">Сепаратор</div>
                    </div>
                </div>

                <div class="palette-section">
                    <div class="palette-section-title">ЗНАЧЕНИЯ</div>

                    <div class="palette-item" data-type="const">
                        <svg viewBox="0 0 60 40">
                            <rect x="10" y="8" width="40" height="24" rx="3" fill="#0f3460" stroke="#3b82f6" stroke-width="2"/>
                            <text x="24" y="25" fill="#3b82f6" font-size="14" font-weight="bold">C</text>
                        </svg>
                        <div class="palette-item-name">Константа</div>
                    </div>

                    <div class="palette-item" data-type="formula">
                        <svg viewBox="0 0 60 40">
                            <rect x="5" y="5" width="50" height="30" rx="5" fill="#0f3460" stroke="#f59e0b" stroke-width="2"/>
                            <text x="12" y="25" fill="#f59e0b" font-size="11" font-weight="bold">f(x)</text>
                        </svg>
                        <div class="palette-item-name">Формула</div>
                    </div>
                </div>

                <div class="type-legend">
                    <div class="type-legend-item">
                        <div class="type-legend-dot logic"></div>
                        <span>Логический</span>
                    </div>
                    <div class="type-legend-item">
```

```
                        <div class="type-legend-dot number"></div>
                        <span>Числовой</span>
                    </div>
                </div>
            </div>

            <div id="workspace-container">
                <svg id="connections-svg"></svg>
                <div id="workspace"></div>
                  <!-- Прямоугольник для выделения элементов -->
                <div id="selection-rect"></div>

                <!-- Мини-карта -->
                <div id="minimap">
                    <div id="minimap-viewport"></div>
                    <canvas id="minimap-canvas"></canvas>
                </div>

                <!-- Координаты и информация -->
                <div id="viewport-info">
                    <span id="cursor-pos">X: 0, Y: 0</span>
                    <span id="selection-info"></span>
                </div>
            </div>
        </div>
    </div>

    <!-- Модальные окна -->
    <div id="modal-overlay">
        <div id="modal">
            <h3 id="modal-title">Свойства элемента</h3>
            <div id="modal-content"></div>
            <div class="modal-buttons">
                <button class="modal-btn cancel" id="modal-cancel">Отмена</button>
                <button class="modal-btn save" id="modal-save">Сохранить</button>
            </div>
        </div>
    </div>

    <!-- Модальное окно свойств проекта -->
    <div id="project-modal-overlay" class="modal-overlay-class">
        <div id="project-modal" class="modal-class">
            <h3>Свойства проекта</h3>
            <div id="project-modal-content"></div>
            <div class="modal-buttons">
                <button class="modal-btn cancel" id="project-modal-cancel">Отмена</
    button>
                <button class="modal-btn save" id="project-modal-save">Сохранить</
    button>
            </div>
        </div>
    </div>

    <div id="code-modal-overlay" class="modal-overlay-class">
        <div id="code-modal" class="modal-class">
            <h3>Сгенерированный код</h3>
            <textarea id="code-output" style="width:100%; height:300px;"></textarea>
            <div class="modal-buttons">
                <button class="modal-btn cancel" id="code-modal-close">Закрыть</
    button>
            </div>
        </div>
    </div>

    <div id="context-menu">
```

```
        <div class="context-item" id="ctx-properties">⚙️ Свойства</div>
        <div class="context-item" id="ctx-delete">🗑️ Удалить</div>
    </div>

   <!-- Модули JavaScript -->
    <!-- Модули JavaScript -->
    <script src="js/config.js"></script>
    <script src="js/state.js"></script>
    <script src="js/utils.js"></script>
    <script src="js/viewport.js"></script>
    <script src="js/elements.js"></script>
    <script src="js/connections.js"></script>
    <script src="js/outputs.js"></script>  <!-- ← Этот файл опционален теперь -->
    <script src="js/modal.js"></script>
    <script src="js/project.js"></script>
    <script src="js/codegen_graph.js"></script>
    <script src="js/codegen_optimizer.js"></script>
    <script src="js/codegen.js"></script>
    <script src="js/settings.js"></script>

    <script src="js/app.js"></script>

    <div id="modal-project-list" class="modal hidden">
        <div class="modal__content modal__content--wide">
            <h2 class="modal__title">Выбор проекта</h2>

            <div class="project-list__toolbar">
            <input id="project-search" type="text" placeholder="Фильтр по имени или описанию…" />
            <button id="project-refresh" class="btn btn-secondary">Обновить</button>
            </div>

            <div class="project-list__table-container">
            <table class="project-list__table">
                <thead>
                <tr>
                    <th>Файл</th>
                    <th>Tagname</th>
                    <th>Description</th>
                    <th>Тип</th>
                </tr>
                </thead>
                <tbody id="project-list-body">
                <tr><td colspan="4" class="project-list__empty">Загрузка…</td></tr>
                </tbody>
            </table>
            </div>

            <div class="modal__actions">
                <button id="project-cancel" class="btn btn-secondary">Отмена</button>
                <button id="project-load" class="btn btn-primary" disabled>Загрузить</button>
            </div>
        </div>
    </div>

</body>
</html>

app.js:
/**
 * Главный модуль приложения
 */

const App = {
```

```
1443      /**
1444       * Инициализация приложения
1445       */
1446      init() {
1447          Settings.init().catch(console.error);
1448          //Settings.init().then(() => {
1449          //    // если хочешь — можно обновить UI (например, статус “Сигналы
      загружены”)
1450          //    console.log('Settings loaded, signals:', Settings.signals.length);
1451          //    }).catch(err => console.error(err));
1452          //console.log('signals loaded:', Settings.signals.slice(0, 5));
1453          this.setupPaletteDragDrop();
1454          this.setupGlobalMouseHandlers();
1455          this.setupContextMenu();
1456          this.setupWorkspaceClick();
1457          this.setupOutputCounter();
1458          this.setupMultiSelection();
1459
1460          // Инициализация модулей
1461          Viewport.init();
1462          Modal.init();
1463          Project.init();
1464
1465          // Первоначальное определение выходов (только если модуль загружен)
1466          if (typeof Outputs !== 'undefined' && Outputs.updateOutputStatus) {
1467              Outputs.updateOutputStatus();
1468          }
1469
1470          console.log('Logic Scheme Editor initialized');
1471          document.getElementById('btn-generate-code').addEventListener('click', () => {
1472              const code = CodeGen.generate();
1473              document.getElementById('code-output').value = code;
1474              document.getElementById('code-modal-overlay').style.display = 'flex';
1475          });
1476
1477          document.getElementById('code-modal-close').addEventListener('click', () => {
1478              document.getElementById('code-modal-overlay').style.display = 'none';
1479          });
1480          document.getElementById('btn-visualize').addEventListener('click', () => {
1481              App.openSignalVisualizer();
1482  });
1483      },
1484
1485      openSignalVisualizer() {
1486          try {
1487              // 1) Собираем входные сигналы
1488              const signals = Object.values(AppState.elements)
1489              .filter(e => e && e.type === 'input-signal')
1490              .map(e => e.props?.name || e.id);
1491              const uniqSignals = [...new Set(signals)];
1492              if (uniqSignals.length === 0) {
1493              alert('Нет входных сигналов в схеме.');
1494              return;
1495              }
1496
1497              // 2) Генерируем код (может быть длинным)
1498              let codeStr = '';
1499              if (typeof CodeGen !== 'undefined' && typeof CodeGen.generate ===
      'function') {
1500              codeStr = CodeGen.generate() || '';
1501              }
1502
1503              // 3) Создаём сессию на backend, чтобы не тащить код в URL
1504              fetch('/api/visualize/session', {
1505              method: 'POST',
```

```
            headers: { 'Content-Type': 'application/json' },
            body: JSON.stringify({ signals: uniqSignals, code: codeStr })
            })
            .then(r => {
            if (!r.ok) throw new Error('Failed to create visualize session');
            return r.json();
            })
            .then(data => {
            const token = data.token;
            const apiUrl = window.location.origin; // http://localhost:8000
            const params = new URLSearchParams();
            params.set('session', token);
            params.set('api_url', apiUrl);
            // signals можно не передавать — визуализатор возьмет их из session
            const visualizerUrl = `http://localhost:8501?${params.toString()}`;
            window.open(visualizerUrl, '_blank');
            })
            .catch(err => {
            console.error(err);
            alert('Не удалось открыть визуализатор: ' + err.message);
            });

        } catch (e) {
            console.error(e);
            alert('Ошибка при подготовке визуализации: ' + e.message);
        }
    },

    /**
     * Отмена состояния drag из палитры (helper)
     */
    cancelPaletteDrag() {
        if (AppState.dragPreview) {
            try { AppState.dragPreview.remove(); } catch (e) { /* ignore */ }
            AppState.dragPreview = null;
        }
        AppState.isDraggingFromPalette = false;
        AppState.dragType = null;
    },

    /**
     * Настройка счётчика выходов в меню
     */
    setupOutputCounter() {
        // Не создавать повторно, если уже есть
        if (document.getElementById('btn-outputs')) return;

        const menu = document.getElementById('menu');

        // Создаём кнопку с счётчиком выходов
        const outputBtn = document.createElement('button');
        outputBtn.className = 'menu-btn output-btn';
        outputBtn.id = 'btn-outputs';
        outputBtn.innerHTML = `
            📤 Выходы
            <span id="output-counter" class="output-counter">0</span>
        `;

        // Вставляем после кнопки свойств проекта
        const projectBtn = document.getElementById('btn-project-settings');
        if (projectBtn) {
            projectBtn.after(outputBtn);
        } else {
            menu.appendChild(outputBtn);
        }
```

```
1571
1572          outputBtn.addEventListener('click', () => {
1573              Modal.showProjectPropertiesModal();
1574          });
1575      },
1576
1577      /**
1578       * Настройка drag & drop из палитры
1579       */
1580      setupPaletteDragDrop() {
1581          document.querySelectorAll('.palette-item').forEach(item => {
1582              item.addEventListener('mousedown', (e) => {
1583                  // Только левая кнопка мыши должна запускать drag из палитры
1584                  if (e.button !== 0) return;
1585                  e.preventDefault();
1586
1587                  AppState.isDraggingFromPalette = true;
1588                  AppState.dragType = item.dataset.type;
1589
1590                  AppState.dragPreview = document.createElement('div');
1591                  AppState.dragPreview.className = 'drag-preview';
1592                  AppState.dragPreview.textContent =
      ELEMENT_TYPES[AppState.dragType]?.name || 'Элемент';
1593                  AppState.dragPreview.style.left = `${e.clientX - 40}px`;
1594                  AppState.dragPreview.style.top = `${e.clientY - 20}px`;
1595                  document.body.appendChild(AppState.dragPreview);
1596              });
1597          });
1598      },
1599
1600      /**
1601       * Глобальные обработчики мыши
1602       */
1603  /**
1604   * Глобальные обработчики мыши
1605   */
1606  setupGlobalMouseHandlers() {
1607      document.addEventListener('mousemove', (e) => {
1608          if (AppState.isDraggingFromPalette && AppState.dragPreview) {
1609              AppState.dragPreview.style.left = `${e.clientX - 40}px`;
1610              AppState.dragPreview.style.top = `${e.clientY - 20}px`;
1611          }
1612          if (AppState.resizing) {
1613              Elements.handleResize(e);
1614              return;
1615          }
1616          if (AppState.draggingElement) {
1617              Elements.handleDrag(e);
1618          }
1619          if (AppState.tempLine && AppState.connectingFrom) {
1620              Connections.drawTempConnection(e);
1621          }
1622      });
1623
1624      document.addEventListener('mouseup', (e) => {
1625          if (AppState.resizing) {
1626              AppState.resizing = null;
1627              if (typeof Outputs !== 'undefined') Outputs.updateOutputStatus();
1628          }
1629
1630          if (AppState.isDraggingFromPalette) {
1631              try {
1632                  if (AppState.dragPreview) {
1633                      AppState.dragPreview.remove();
1634                      AppState.dragPreview = null;
```

```
                }

                const container = document.getElementById('workspace-container');
                const rect = container.getBoundingClientRect();

                if (e.clientX >= rect.left && e.clientX <= rect.right &&
                    e.clientY >= rect.top && e.clientY <= rect.bottom) {

                    const canvasPos = screenToCanvas(e.clientX, e.clientY);
                    const config = ELEMENT_TYPES[AppState.dragType];
                    if (config) {
                        const defaultWidth = config.minWidth || 120;
                        const defaultHeight = config.minHeight || 60;

                        // ИСПРАВЛЕНО: addElement возвращает DOM-элемент, его надо обработать
                        const newElement = Elements.addElement(
                            AppState.dragType,
                            canvasPos.x - defaultWidth / 2,
                            canvasPos.y - defaultHeight / 2
                        );

                        if (newElement && typeof Outputs !== 'undefined') {
                            Outputs.updateOutputStatus();
                        }
                    } else {
                        console.error('Неизвестный тип элемента при drop:', AppState.dragType);
                    }
                }
            } finally {
                App.cancelPaletteDrag();
            }
        }

        if (AppState.draggingElement) {
            AppState.draggingElement = null;
        }

        Connections.clearConnectionState();
    });

    document.addEventListener('keydown', (e) => {
        if (e.key === 'Delete' && AppState.selectedElement) {
            Elements.deleteElement(AppState.selectedElement);
            if (typeof Outputs !== 'undefined') Outputs.updateOutputStatus();
        }
        if (e.key === 'Escape') {
            Elements.deselectAll();
            Connections.clearConnectionState();
            if (AppState.isDraggingFromPalette) App.cancelPaletteDrag();
        }
    });
},

    /**
     * Настройка контекстного меню
     */
    setupContextMenu() {
        document.addEventListener('click', (e) => {
            const menu = document.getElementById('context-menu');
            if (!menu.contains(e.target)) {
                menu.style.display = 'none';
            }
        });
```

```
1698
1699          document.getElementById('ctx-properties').addEventListener('click', () => {
1700              const elemId = document.getElementById('context-menu').dataset.elementId;
1701              document.getElementById('context-menu').style.display = 'none';
1702              const config = ELEMENT_TYPES[AppState.elements[elemId]?.type];
1703              if (config?.hasProperties) {
1704                  Modal.showPropertiesModal(elemId);
1705              }
1706          });
1707
1708          document.getElementById('ctx-delete').addEventListener('click', () => {
1709              const elemId = document.getElementById('context-menu').dataset.elementId;
1710              document.getElementById('context-menu').style.display = 'none';
1711              Elements.deleteElement(elemId);
1712              // Обновляем выходы только если модуль загружен
1713              if (typeof Outputs !== 'undefined' && Outputs.updateOutputStatus) {
1714                  Outputs.updateOutputStatus();
1715              }
1716          });
1717      },
1718
1719      /**
1720       * Клик по рабочей области
1721       */
1722      setupWorkspaceClick() {
1723          const workspace = document.getElementById('workspace');
1724
1725          workspace.addEventListener('click', (e) => {
1726              if (e.target === workspace) {
1727                  Elements.deselectAll();
1728              }
1729          });
1730      },
1731      /**
1732   * --- Выделение рамкой и множественное перемещение ---
1733   */
1734      setupMultiSelection() {
1735          const container = document.getElementById('workspace-container');
1736          const rectEl = document.getElementById('selection-rect');
1737
1738          container.addEventListener('mousedown', (e) => {
1739              if (e.button !== 0) return;
1740              if (e.target !== document.getElementById('workspace')) return;
1741
1742              const pos = screenToCanvas(e.clientX, e.clientY);
1743              AppState.multiSelecting = true;
1744              AppState.selectionRect = { startX: pos.x, startY: pos.y, x: pos.x, y:
      pos.y, w: 0, h: 0 };
1745
1746              rectEl.style.left = e.clientX + 'px';
1747              rectEl.style.top = e.clientY + 'px';
1748              rectEl.style.width = '0px';
1749              rectEl.style.height = '0px';
1750              rectEl.style.display = 'block';
1751          });
1752
1753          document.addEventListener('mousemove', (e) => {
1754              if (!AppState.multiSelecting) return;
1755
1756              const pos = screenToCanvas(e.clientX, e.clientY);
1757              const sx = AppState.selectionRect.startX;
1758              const sy = AppState.selectionRect.startY;
1759              const x = Math.min(sx, pos.x);
1760              const y = Math.min(sy, pos.y);
1761              const w = Math.abs(pos.x - sx);
```

```
            const h = Math.abs(pos.y - sy);

            rectEl.style.left = x * AppState.viewport.zoom + AppState.viewport.panX +
    'px';
            rectEl.style.top = y * AppState.viewport.zoom + AppState.viewport.panY +
    'px';
            rectEl.style.width = w * AppState.viewport.zoom + 'px';
            rectEl.style.height = h * AppState.viewport.zoom + 'px';

            const selected = [];
            for (const [id, elData] of Object.entries(AppState.elements)) {
            if (!elData || elData.type === 'output-frame') continue;
            if (
                elData.x >= x && elData.x + elData.width <= x + w &&
                elData.y >= y && elData.y + elData.height <= y + h
            ) selected.push(id);
            }

            AppState.selectedElements = selected;
            document.querySelectorAll('.element').forEach(el =>
            el.classList.toggle('selected', selected.includes(el.id))
            );
        });

        document.addEventListener('mouseup', () => {
            if (AppState.multiSelecting) {
            AppState.multiSelecting = false;
            rectEl.style.display = 'none';
            }
        });
    }
};

// Запуск приложения при загрузке страницы
document.addEventListener('DOMContentLoaded', () => {
    App.init();
});

codegen_graph.js
// js/codegen_graph.js


const CodeGenGraph = {
        /**
     * Собрать все условия вверх по цепочке cond-портов (до корня).
     * Возвращает null или объединённое через AND условие.
     */
/**
 * Собрать ВСЕ условия: и через cond-порты, и через контекст обычных входов
 */
    collectAllCond(graph) {
        if (!graph) return null;

        let c = null;
        const elem = graph.elem;

        // 1. Собираем условия через cond-порт (как было)
        if (graph.condInput) {
            const condConn = graph.condInput.conn;
            const fromGraph = graph.condInput.fromGraph;
            const oneCond = this.evalConditionFromPort(fromGraph, condConn.fromPort);
            c = oneCond;

            // Рекурсивно идём вверх по cond-цепочке
            const upCond = this.collectAllCond(fromGraph);
```

```
            if (upCond) {
                c = c ? Optimizer.And(c, upCond) : upCond;
            }
        }

        // 2. НОВОЕ: если это separator — учитываем контекст его входа
        if (elem.type === 'separator' && graph.inputs.length > 0) {
            const inputGraph = graph.inputs[0].fromGraph;
            const inputContext = this.collectAllCond(inputGraph);
            if (inputContext) {
                c = c ? Optimizer.And(c, inputContext) : inputContext;
            }
        }

        return c;
    },
    buildDependencyGraph(elementId) {
        const graph = {
            nodeId: elementId,
            elem: AppState.elements[elementId],
            inputs: [],
            condInput: null,
        };

        if (!graph.elem) return null;

        const inConns = AppState.connections
        .filter(c => c.toElement === elementId && c.toPort.startsWith('in-'))
        .sort((a, b) => {
            const ai = parseInt(a.toPort.split('-')[1] || '0', 10);
            const bi = parseInt(b.toPort.split('-')[1] || '0', 10);
            return ai - bi;
        });

        inConns.forEach(conn => {
        graph.inputs.push({
            conn,
            fromGraph: this.buildDependencyGraph(conn.fromElement)
        });
        });

        const condConn = AppState.connections.find(c =>
            c.toElement === elementId && c.toPort === 'cond-0'
        );
        if (condConn) {
            graph.condInput = {
                conn: condConn,
                fromGraph: this.buildDependencyGraph(condConn.fromElement)
            };
        }

        return graph;
    },

    /**
     * Получить ЛОГИКУ из графа (для IF/AND/OR/NOT/SEPARATOR)
     */
    evalLogic(graph) {
        if (!graph) return Optimizer.TrueCond;
        const elem = graph.elem;

        switch (elem.type) {
            case 'if': {
                const left = graph.inputs[0]?.fromGraph;
                const right = graph.inputs[1]?.fromGraph;
```

```
                const leftVal = left ? this.evalValue(left) : Optimizer.Const(0);
                const rightVal = right ? this.evalValue(right) : Optimizer.Const(0);

                const op = elem.props.operator || '=';
                return this.buildIfLogic(leftVal, op, rightVal);
            }

            case 'and': {
                let result = null;
                for (const inp of graph.inputs) {
                    const inLogic = this.evalLogic(inp.fromGraph);
                    result = result ? Optimizer.And(result, inLogic) : inLogic;
                }
                return result || Optimizer.TrueCond;
            }

            case 'or': {
                let result = null;
                for (const inp of graph.inputs) {
                    const inLogic = this.evalLogic(inp.fromGraph);
                    result = result ? Optimizer.Or(result, inLogic) : inLogic;
                }
                return result || Optimizer.FalseCond;
            }

            case 'not': {
                const inLogic = this.evalLogic(graph.inputs[0]?.fromGraph);
                return Optimizer.Not(inLogic);
            }

            case 'separator': {
                return this.evalLogic(graph.inputs[0]?.fromGraph);
            }

            default:
                return Optimizer.TrueCond;
        }
    },

    /**
     * Получить ЗНАЧЕНИЕ из графа (для INPUT/CONST/FORMULA)
     */
    evalValue(graph) {
        if (!graph) return Optimizer.Const(0);
        const elem = graph.elem;

        switch (elem.type) {
            case 'input-signal':
                return Optimizer.Var(elem.props.name || graph.nodeId);

            case 'const':
                return Optimizer.Const(Number(elem.props.value) || 0);

            case 'formula': {
                const expr = this.buildFormulaExpr(elem);
                return Optimizer.Var(expr);
            }

            case 'separator':
                return this.evalValue(graph.inputs[0]?.fromGraph);

            default:
                return Optimizer.Const(0);
        }
```

```
    },

    // js/codegen_graph.js

    /**
     * Рекурсивно собрать полный контекст условий для элемента
     * через всю цепочку cond-портов вверх
     */
 // В codegen_graph.js, в evalFullContext добавь:

    evalFullContext(graph) {
        if (!graph) return null;

        let context = null;
        const elem = graph.elem;

        console.log(`evalFullContext для ${elem.id} (${elem.type})`);

        // 1. Если сам элемент имеет cond-порт — собираем его условие
        if (graph.condInput) {
            const condConn = graph.condInput.conn;
            console.log(`  → имеет cond-0 от ${graph.condInput.fromGraph.elem.id}.${condConn.fromPort}`);

            const condLogic = this.evalConditionFromPort(
                graph.condInput.fromGraph,
                condConn.fromPort
            );
            console.log(`  → условие от cond-0: ${Optimizer.printCond(condLogic)}`);
            context = condLogic;

            // 2. Рекурсивно собираем контекст элемента, на который указывает cond-порт
            const upstreamContext = this.evalFullContext(graph.condInput.fromGraph);
            if (upstreamContext) {
                console.log(`  → upstreamContext: ${Optimizer.printCond(upstreamContext)}`);
                context = context ? Optimizer.And(context, upstreamContext) : upstreamContext;
            }
        } else {
            console.log(`  → нет cond-0`);
        }

        console.log(`  → итоговый контекст: ${Optimizer.printCond(context)}`);
        return context;
    },

    /**
     * Получить УСЛОВИЕ для cond-порта элемента
     * Учитывает цепочку сепараторов с TRUE/FALSE ветвлением
     */
    evalConditionFromPort(graph, fromPort) {
        if (!graph) return null;
        const elem = graph.elem;

        // Если это сепаратор — вычисляем его вход и применяем ветвление
        if (elem.type === 'separator') {
            const inputLogic = this.evalLogic(graph.inputs[0]?.fromGraph);

            if (fromPort === 'out-0') {
                return inputLogic;
            } else if (fromPort === 'out-1') {
                return Optimizer.Not(inputLogic);
            }
```

```
2016          }
2017
2018          // Если это логический элемент (AND/OR/NOT/IF) — просто вычисляем логику
2019          if (elem.type === 'and' || elem.type === 'or' || elem.type === 'not' ||
      elem.type === 'if') {
2020              return this.evalLogic(graph);
2021          }
2022
2023          return null;
2024      },
2025
2026      /**
2027       * Главная функция: получить {cond, expr} для элемента
2028       */
2029      evalGraphValue(graph) {
2030
2031          if (!graph) return { cond: null, expr: Optimizer.Const(0) };
2032
2033          const elem = graph.elem;
2034          //let cond = null;
2035
2036          // ← НОВОЕ: собираем полный контекст через цепочку cond-портов
2037          let cond = this.collectAllCond(graph);
2038
2039          let expr = null;
2040
2041          switch (elem.type) {
2042              case 'input-signal':
2043                  expr = Optimizer.Var(elem.props.name || graph.nodeId);
2044                  break;
2045
2046              case 'const':
2047                  expr = Optimizer.Const(Number(elem.props.value) || 0);
2048                  break;
2049
2050              case 'formula': {
2051                  // Для формулы также собираем условия от всех входных элементов
2052                  const inputConds = graph.inputs.map(inp => {
2053                      const inResult = this.evalGraphValue(inp.fromGraph);
2054                      return inResult.cond;
2055                  }).filter(c => c);
2056
2057                  // Объединяем cond-порт с условиями от входов
2058                  for (const inCond of inputConds) {
2059                      cond = cond ? Optimizer.And(cond, inCond) : inCond;
2060                  }
2061
2062                  expr = Optimizer.Var(this.buildFormulaExpr(elem));
2063                  break;
2064              }
2065
2066              case 'separator':
2067                  // Сепаратор — просто пробрасываем значение дальше
2068                  return this.evalGraphValue(graph.inputs[0]?.fromGraph);
2069
2070              // Логические элементы не должны здесь быть
2071              case 'and':
2072              case 'or':
2073              case 'not':
2074              case 'if':
2075              default:
2076                  expr = Optimizer.Const(0);
2077          }
2078
2079          return { cond, expr };
```

```
    },

    buildIfLogic(leftVal, op, rightVal) {
        const leftName = leftVal.type === 'var' ? leftVal.name : String(leftVal.n);
        const rightName = rightVal.type === 'var' ? rightVal.name :
String(rightVal.n);

        const leftZero = leftVal.type === 'const' && leftVal.n === 0;
        const rightZero = rightVal.type === 'const' && rightVal.n === 0;

        switch (op) {
            case '=':
                if (rightZero) return Optimizer.Eq0(leftName);
                if (leftZero) return Optimizer.Eq0(rightName);
                return Optimizer.Cmp(leftName, '=', rightName);
            case '!=':
                if (rightZero) return Optimizer.Ne0(leftName);
                if (leftZero) return Optimizer.Ne0(rightName);
                return Optimizer.Cmp(leftName, '!=', rightName);
            case '>':
            case '<':
            case '>=':
            case '<=':
                return Optimizer.Cmp(leftName, op, rightName);
            default:
                return Optimizer.TrueCond;
        }
    },


    buildFormulaExpr(elem) {
        let result = elem.props.expression || '0';

        // 1) Сначала раскрываем шаблоны (h и др.)
        const map = (typeof Settings !== 'undefined' && Settings.getTemplatesMap)
            ? Settings.getTemplatesMap()
            : null;
        result = expandFormulaTemplates(result, map);

        // 2) Потом раскрываем ссылки на формулы
        const formulaRefs = result.match(/formula[_-]\d+/g) || [];
        for (const ref of formulaRefs) {
            const refElem = AppState.elements[ref];
            if (refElem && refElem.type === 'formula') {
                const refExpr = this.buildFormulaExpr(refElem);
                result = result.replace(new RegExp(ref, 'g'), `(${refExpr})`);
            }
        }

        return result;
    }
};

window.CodeGenGraph = CodeGenGraph;

codegen_optimizer.js

// js/codegen_optimizer.js

let _depth = 0;
const MAX_DEPTH = 200;

// === Конструкторы ===
function Eq0(v) { return { kind: 'cond', type: 'eq0', v }; }
function Ne0(v) { return { kind: 'cond', type: 'ne0', v }; }
```

```
function Cmp(l, op, r) { return { kind: 'cond', type: 'cmp', l, op, r }; }
function And(a, b) {
    if (!a) return b;
    if (!b) return a;
    return { kind: 'cond', type: 'and', a, b };
}
function Or(a, b) {
    if (!a) return b;
    if (!b) return a;
    return { kind: 'cond', type: 'or', a, b };
}
function Not(x) {
    if (!x) return null;
    return { kind: 'cond', type: 'not', x };
}
const TrueCond = { kind: 'cond', type: 'true' };
const FalseCond = { kind: 'cond', type: 'false' };

function Const(n) { return { kind: 'expr', type: 'const', n }; }
function Var(name) { return { kind: 'expr', type: 'var', name }; }
function Op(op, l, r) { return { kind: 'expr', type: 'op', op, l, r }; }
function When(c, t, e) { return { kind: 'expr', type: 'when', c, t, e }; }

// === Утилиты ===
function atomKey(c) {
    if (!c) return null;
    switch (c.type) {
        case 'eq0': return `eq0:${c.v}`;
        case 'ne0': return `ne0:${c.v}`;
        case 'cmp': return `cmp:${c.l}:${c.op}:${c.r}`;
        case 'true': return 'true';
        case 'false': return 'false';
        default: return null;
    }
}

function splitAndCond(c) {
    if (!c || c.type !== 'and') return null;
    return [c.a, c.b];
}

function findSharedAndComplement(c1, c2) {
    const p1 = splitAndCond(c1);
    const p2 = splitAndCond(c2);
    if (!p1 || !p2) return null;

    const combos = [
        [p1[0], p1[1], p2[0], p2[1]],
        [p1[0], p1[1], p2[1], p2[0]],
        [p1[1], p1[0], p2[0], p2[1]],
        [p1[1], p1[0], p2[1], p2[0]],
    ];

    for (const [s1, x1, s2, x2] of combos) {
        if (condEq(s1, s2) && condNegationEq(x1, x2)) {
            return { shared: s1 };
        }
    }
    return null;
}

function negateOp(op) {
    switch (op) {
        case '=': return '!=';
        case '!=': return '=';
```

```
        case '>': return '<=';
        case '<': return '>=';
        case '>=': return '<';
        case '<=': return '>';
        default: return null;
    }
}

// Преобразует cmp-условие в интервал по одной переменной
// Возвращает { varName, min, minInc, max, maxInc } или null
function cmpToInterval(c) {
    if (!c || c.type !== 'cmp') return null;

    const lNum = parseNumberLiteral(c.l);
    const rNum = parseNumberLiteral(c.r);

    let varName, op, val;

    if (lNum == null && rNum != null) {
        // var OP const
        varName = c.l;
        op = c.op;
        val = rNum;
    } else if (lNum != null && rNum == null) {
        // const OP var  ->  var (OP') const
        varName = c.r;
        op = reverseOp(c.op);
        if (!op) return null;
        val = lNum;
    } else {
        // Либо обе стороны числа, либо обе не числа — не трогаем
        return null;
    }

    // Интересуют только упорядочивающие операторы
    switch (op) {
        case '<':
        case '<=':
        case '>':
        case '>=':
        case '=':
            break;
        default:
            return null;
    }

    let min = Number.NEGATIVE_INFINITY;
    let max = Number.POSITIVE_INFINITY;
    let minInc = false;
    let maxInc = false;

    switch (op) {
        case '<':
            max = val; maxInc = false; break;
        case '<=':
            max = val; maxInc = true; break;
        case '>':
            min = val; minInc = false; break;
        case '>=':
            min = val; minInc = true; break;
        case '=':
            min = val; minInc = true;
            max = val; maxInc = true;
            break;
    }
```

```
2274
2275      return { varName, min, minInc, max, maxInc };
2276  }
2277
2278  function intervalSubset(a, b) {
2279      if (!a || !b) return false;
2280
2281      // Нижняя граница: a.min >= b.min
2282      const amin = a.min, bmin = b.min;
2283      if (amin === Number.NEGATIVE_INFINITY) {
2284          if (bmin !== Number.NEGATIVE_INFINITY) return false;
2285          // оба -∞ — ок
2286      } else if (bmin === Number.NEGATIVE_INFINITY) {
2287          // b начинается “раньше” — ок
2288      } else if (amin > bmin) {
2289          // a стартует правее b — ок
2290      } else if (amin < bmin) {
2291          // a захватывает меньшее значение — не подмножество
2292          return false;
2293      } else {
2294          // amin === bmin
2295          if (a.minInc && !b.minInc) {
2296              // a включает границу, а b — нет → в a есть точка, не входящая в b
2297              return false;
2298          }
2299      }
2300
2301      // Верхняя граница: a.max <= b.max
2302      const amax = a.max, bmax = b.max;
2303      if (amax === Number.POSITIVE_INFINITY) {
2304          if (bmax !== Number.POSITIVE_INFINITY) return false;
2305      } else if (bmax === Number.POSITIVE_INFINITY) {
2306          // b идёт дальше — ок
2307      } else if (amax < bmax) {
2308          // a заканчивается раньше — ок
2309      } else if (amax > bmax) {
2310          return false;
2311      } else {
2312          // amax === bmax
2313          if (a.maxInc && !b.maxInc) {
2314              return false;
2315          }
2316      }
2317
2318      return true;
2319  }
2320
2321  // Удаляет избыточные cmp-условия в массиве атомов
2322  // mode: 'and' | 'or'
2323  function removeRedundantCmpAtoms(atoms, mode) {
2324      if (!atoms || atoms.length < 2) return atoms;
2325
2326      const keep = new Array(atoms.length).fill(true);
2327
2328      for (let i = 0; i < atoms.length; i++) {
2329          if (!keep[i]) continue;
2330          const a = atoms[i];
2331          if (!a || a.type !== 'cmp') continue;
2332
2333          for (let j = 0; j < atoms.length; j++) {
2334              if (i === j || !keep[j]) continue;
2335              const b = atoms[j];
2336              if (!b || b.type !== 'cmp') continue;
2337
2338              const rel = cmpImplicationRelation(a, b);
```

```
            if (!rel) continue;

            if (rel === 'a_in_b') {
                if (mode === 'or') {
                    // A ⊆ B → A OR B = B  → A лишнее
                    keep[i] = false;
                    break;
                } else if (mode === 'and') {
                    // A ⊆ B → A AND B = A → B лишнее
                    keep[j] = false;
                }
            } else if (rel === 'b_in_a') {
                if (mode === 'or') {
                    // B ⊆ A → A OR B = A → B лишнее
                    keep[j] = false;
                } else if (mode === 'and') {
                    // B ⊆ A → A AND B = B → A лишнее
                    keep[i] = false;
                    break;
                }
            }
        }
    }

    return atoms.filter((_, idx) => keep[idx]);
}

// Отношение между двумя cmp-условиями через интервалы
// 'a_in_b'  — A ⊆ B
// 'b_in_a'  — B ⊆ A
// 'equal'   — одинаковые интервалы (редко используем)
// null      — не можем определить
function cmpImplicationRelation(c1, c2) {
    const i1 = cmpToInterval(c1);
    const i2 = cmpToInterval(c2);
    if (!i1 || !i2) return null;
    if (i1.varName !== i2.varName) return null;

    const aInB = intervalSubset(i1, i2);
    const bInA = intervalSubset(i2, i1);

    if (aInB && bInA) return 'equal';
    if (aInB) return 'a_in_b';
    if (bInA) return 'b_in_a';
    return null;
}

// Разворот оператора при перестановке аргументов (левый/правый)
function reverseOp(op) {
    switch (op) {
        case '<':  return '>';
        case '>':  return '<';
        case '<=': return '>=';
        case '>=': return '<=';
        case '=':
        case '!=':
            return op;
        default:
            return null;
    }
}

// Аккуратный парсер числового литерала.
// Возвращает число или null, если строка не чисто числовая.
function parseNumberLiteral(s) {
```

```
    if (typeof s !== 'string') return null;
    const trimmed = s.trim().replace(',', '.');

    // Только простые вещи: -123, 45, 3.14
    if (!/^-?\d+(\.\d+)?$/.test(trimmed)) return null;

    const n = Number(trimmed);
    return Number.isFinite(n) ? n : null;
}


function negateAtomKey(key) {
    if (!key) return null;
    if (key.startsWith('eq0:')) return 'ne0:' + key.slice(4);
    if (key.startsWith('ne0:')) return 'eq0:' + key.slice(4);
    if (key.startsWith('cmp:')) {
        const parts = key.slice(4).split(':');
        if (parts.length === 3) {
            const negOp = negateOp(parts[1]);
            if (negOp) return `cmp:${parts[0]}:${negOp}:${parts[2]}`;
        }
    }
    return null;
}

function isNegation(a, b) {
    if (!a || !b) return false;
    if (a.type === 'eq0' && b.type === 'ne0' && a.v === b.v) return true;
    if (a.type === 'ne0' && b.type === 'eq0' && a.v === b.v) return true;
    if (a.type === 'cmp' && b.type === 'cmp' && a.l === b.l && a.r === b.r) {
        return a.op === negateOp(b.op);
    }
    if (a.type === 'not' && condEq(a.x, b)) return true;
    if (b.type === 'not' && condEq(b.x, a)) return true;
    return false;
}

function isAtomCond(t) {
    return t && (t.type === 'eq0' || t.type === 'ne0' || t.type === 'cmp');
}

function pruneOrByContext(orTerm, contextAtoms) {
    const branches = flattenOr(orTerm);
    const kept = [];

    for (const br of branches) {
        let contradicts = false;

        for (const ctx of contextAtoms) {
            if (isNegation(br, ctx)) {
                contradicts = true;
                break;
            }
        }

        if (!contradicts) kept.push(br);
    }

    if (kept.length === 0) return FalseCond;
    if (kept.length === 1) return kept[0];
    return buildOr(kept);
}

function condNegationEq(a, b) {
    if (!a || !b) return false;
```

```
2469
2470      // Простая проверка: a == NOT(b)
2471      if (condEq(a, Not(b)) || condEq(b, Not(a))) return true;
2472
2473      // Де Морган: NOT(A OR B) == (NOT A AND NOT B)
2474      // Проверяем: если a = (A OR B), то b должно быть (NOT A AND NOT B)
2475      if (a.type === 'or' && b.type === 'and') {
2476          return condNegationEq(a.a, b.a) && condNegationEq(a.b, b.b) ||
2477                 condNegationEq(a.a, b.b) && condNegationEq(a.b, b.a);
2478      }
2479      // Симметрично
2480      if (a.type === 'and' && b.type === 'or') {
2481          return condNegationEq(a.a, b.a) && condNegationEq(a.b, b.b) ||
2482                 condNegationEq(a.a, b.b) && condNegationEq(a.b, b.a);
2483      }
2484
2485      // Проверка атомов: (X = 0) vs (X != 0)
2486      if (a.type === 'eq0' && b.type === 'ne0' && a.v === b.v) return true;
2487      if (a.type === 'ne0' && b.type === 'eq0' && a.v === b.v) return true;
2488
2489      // Проверка сравнений: (X > Y) vs (X <= Y) и т.д.
2490      if (a.type === 'cmp' && b.type === 'cmp' && a.l === b.l && a.r === b.r) {
2491          return a.op === negateOp(b.op);
2492      }
2493
2494      return false;
2495  }
2496
2497
2498
2499  function condEq(a, b) {
2500      if (a === b) return true;
2501      if (!a || !b) return false;
2502      if (a.type !== b.type) return false;
2503
2504      switch (a.type) {
2505          case 'eq0':
2506          case 'ne0':
2507              return a.v === b.v;
2508          case 'cmp':
2509              return a.l === b.l && a.op === b.op && a.r === b.r;
2510          case 'true':
2511          case 'false':
2512              return true;
2513          case 'not':
2514              return condEq(a.x, b.x);
2515          case 'and':
2516          case 'or':
2517              return (condEq(a.a, b.a) && condEq(a.b, b.b)) ||
2518                     (condEq(a.a, b.b) && condEq(a.b, b.a));
2519          default:
2520              return false;
2521      }
2522  }
2523
2524  function flattenAnd(c) {
2525      if (!c) return [];
2526      if (c.type === 'and') return [...flattenAnd(c.a), ...flattenAnd(c.b)];
2527      return [c];
2528  }
2529
2530  function flattenOr(c) {
2531      if (!c) return [];
2532      if (c.type === 'or') return [...flattenOr(c.a), ...flattenOr(c.b)];
2533      return [c];
```

```
}

function buildAnd(terms) {
    if (terms.length === 0) return TrueCond;
    let result = terms[0];
    for (let i = 1; i < terms.length; i++) {
        result = And(result, terms[i]);
    }
    return result;
}

function buildOr(terms) {
    if (terms.length === 0) return FalseCond;
    let result = terms[0];
    for (let i = 1; i < terms.length; i++) {
        result = Or(result, terms[i]);
    }
    return result;
}

// Поглощение для AND: X AND (X OR Y) = X
function applyAndAbsorption(terms) {
    if (!terms || terms.length < 2) return terms;

    const keep = new Array(terms.length).fill(true);

    for (let i = 0; i < terms.length; i++) {
        if (!keep[i]) continue;
        const ti = terms[i];
        if (!ti || ti.type !== 'or') continue;

        const orParts = flattenOr(ti);
        let drop = false;

        outer:
        for (const part of orParts) {
            for (let j = 0; j < terms.length; j++) {
                if (j === i || !keep[j]) continue;
                if (condEq(part, terms[j])) {
                    drop = true;
                    break outer;
                }
            }
        }

        if (drop) {
            keep[i] = false;
        }
    }

    return terms.filter((_, idx) => keep[idx]);
}

// Поглощение для OR: X OR (X AND Y) = X
function applyOrAbsorption(terms) {
    if (!terms || terms.length < 2) return terms;

    const keep = new Array(terms.length).fill(true);

    for (let i = 0; i < terms.length; i++) {
        if (!keep[i]) continue;
        const ti = terms[i];
        if (!ti || ti.type !== 'and') continue;

        const andParts = flattenAnd(ti);
```

```
        let drop = false;

        outer:
        for (const part of andParts) {
            for (let j = 0; j < terms.length; j++) {
                if (j === i || !keep[j]) continue;
                if (condEq(part, terms[j])) {
                    drop = true;
                    break outer;
                }
            }
        }

        if (drop) {
            keep[i] = false;
        }
    }

    return terms.filter((_, idx) => keep[idx]);
}

// === Упрощение условий ===
function simplifyCond(c) {
    _depth++;
    if (_depth > MAX_DEPTH) {
        _depth--;
        return c;
    }

    try {
        return simplifyCondCore(c);
    } finally {
        _depth--;
    }
}

function simplifyCondCore(c) {
    if (!c || c.kind !== 'cond') return c;

    switch (c.type) {
        case 'true':
        case 'false':
        case 'eq0':
        case 'ne0':
        case 'cmp':
            return c;

        case 'not': {
            const x = simplifyCondCore(c.x);
            if (!x) return TrueCond;
            if (x.type === 'true') return FalseCond;
            if (x.type === 'false') return TrueCond;
            if (x.type === 'not') return simplifyCondCore(x.x);
            if (x.type === 'eq0') return Ne0(x.v);
            if (x.type === 'ne0') return Eq0(x.v);
            if (x.type === 'cmp') {
                const negOp = negateOp(x.op);
                if (negOp) return Cmp(x.l, negOp, x.r);
            }
            if (x.type === 'and') return simplifyCondCore(Or(Not(x.a), Not(x.b)));
            if (x.type === 'or') return simplifyCondCore(And(Not(x.a), Not(x.b)));
            return Not(x);
        }

case 'and': {
```

```
    const a = simplifyCondCore(c.a);
    const b = simplifyCondCore(c.b);

    if (!a) return b;
    if (!b) return a;
    if (a.type === 'false' || b.type === 'false') return FalseCond;
    if (a.type === 'true') return b;
    if (b.type === 'true') return a;

    const allTerms = [...flattenAnd(a), ...flattenAnd(b)];

    // === НОВОЕ: Сразу собираем все eq0/ne0 для быстрой проверки ===
    const eq0Vars = new Map(); // var -> term
    const ne0Vars = new Map(); // var -> term
    const cmpTerms = [];
    const otherTerms = [];

    for (const t of allTerms) {
        if (t.type === 'true') continue;
        if (t.type === 'false') return FalseCond;

        if (t.type === 'eq0') {
            // Проверка на противоречие сразу
            if (ne0Vars.has(t.v)) {
                console.log(`Противоречие найдено: ${t.v} = 0 AND ${t.v} != 0`);
                return FalseCond;
            }
            eq0Vars.set(t.v, t);
        } else if (t.type === 'ne0') {
            // Проверка на противоречие сразу
            if (eq0Vars.has(t.v)) {
                console.log(`Противоречие найдено: ${t.v} != 0 AND ${t.v} = 0`);
                return FalseCond;
            }
            ne0Vars.set(t.v, t);
        } else if (t.type === 'cmp') {
            cmpTerms.push(t);
        } else if (t.type === 'or') {
            // === НОВОЕ: Проверяем каждую ветку OR на противоречие с контекстом ===
            const orTerms = flattenOr(t);
            const validBranches = [];

            for (const branch of orTerms) {
                let branchValid = true;

                if (branch.type === 'ne0' && eq0Vars.has(branch.v)) {
                    console.log(`OR ветка ${branch.v} != 0 противоречит контексту $
{branch.v} = 0`);
                    branchValid = false;
                } else if (branch.type === 'eq0' && ne0Vars.has(branch.v)) {
                    console.log(`OR ветка ${branch.v} = 0 противоречит контексту $
{branch.v} != 0`);
                    branchValid = false;
                }

                if (branchValid) {
                    validBranches.push(branch);
                }
            }

            if (validBranches.length === 0) {
                console.log(`Все ветки OR противоречат контексту → FALSE`);
                return FalseCond;
            } else if (validBranches.length === 1) {
                // Если осталась только одна ветка OR, добавляем её напрямую
```

```
                const singleBranch = validBranches[0];
                if (singleBranch.type === 'eq0') {
                    if (ne0Vars.has(singleBranch.v)) return FalseCond;
                    eq0Vars.set(singleBranch.v, singleBranch);
                } else if (singleBranch.type === 'ne0') {
                    if (eq0Vars.has(singleBranch.v)) return FalseCond;
                    ne0Vars.set(singleBranch.v, singleBranch);
                } else {
                    otherTerms.push(singleBranch);
                }
            } else {
                // Перестраиваем OR только с валидными ветками
                otherTerms.push(buildOr(validBranches));
            }
        } else {
            otherTerms.push(t);
        }
    }

    // Собираем уникальные атомы
    const atomMap = new Map();

    for (const [v, term] of eq0Vars) {
        const key = atomKey(term);
        if (key) atomMap.set(key, term);
    }

    for (const [v, term] of ne0Vars) {
        const key = atomKey(term);
        if (key) atomMap.set(key, term);
    }

    for (const term of cmpTerms) {
        const key = atomKey(term);
        if (key) {
            const negKey = negateAtomKey(key);
            if (negKey && atomMap.has(negKey)) {
                return FalseCond;
            }
            if (!atomMap.has(key)) {
                atomMap.set(key, term);
            }
        }
    }

    let uniqueAtoms = Array.from(atomMap.values());
    uniqueAtoms = removeRedundantCmpAtoms(uniqueAtoms, 'and');

    let result = [...uniqueAtoms, ...otherTerms];

    // Поглощение: X AND (X OR Y) = X
    // === НОВОЕ: выбрасываем из OR ветки, противоречащие контексту AND ===
    const contextAtoms = result.filter(t => isAtomCond(t));
    result = result.map(t => {
        if (t.type !== 'or') return t;
        return pruneOrByContext(t, contextAtoms);
    }).filter(t => t.type !== 'true'); // на всякий случай

    result = applyAndAbsorption(result);

    if (result.length === 0) return TrueCond;
    if (result.length === 1) return result[0];

    return buildAnd(result);
}
```

```
2792
2793          case 'or': {
2794              const a = simplifyCondCore(c.a);
2795              const b = simplifyCondCore(c.b);
2796
2797              if (!a) return b;
2798              if (!b) return a;
2799              if (a.type === 'true' || b.type === 'true') return TrueCond;
2800              if (a.type === 'false') return b;
2801              if (b.type === 'false') return a;
2802
2803              const allTerms = [...flattenOr(a), ...flattenOr(b)];
2804              const atomMap = new Map();
2805              const otherTerms = [];
2806
2807              for (const t of allTerms) {
2808                  if (t.type === 'true') return TrueCond;
2809                  if (t.type === 'false') continue;
2810
2811                  const key = atomKey(t);
2812                  if (key) {
2813                      const negKey = negateAtomKey(key);
2814                      if (negKey && atomMap.has(negKey)) {
2815                          return TrueCond;
2816                      }
2817                      if (!atomMap.has(key)) {
2818                          atomMap.set(key, t);
2819                      }
2820                  } else {
2821                      otherTerms.push(t);
2822                  }
2823              }
2824
2825              let uniqueAtoms = Array.from(atomMap.values());
2826              uniqueAtoms = removeRedundantCmpAtoms(uniqueAtoms, 'or');
2827
2828              let result = [...uniqueAtoms, ...otherTerms];
2829
2830              // Поглощение: X OR (X AND Y) = X
2831              result = applyOrAbsorption(result);
2832
2833              if (result.length === 0) return FalseCond;
2834              if (result.length === 1) return result[0];
2835
2836              return buildOr(result);
2837          }
2838
2839          default:
2840              return c;
2841      }
2842  }
2843
2844  // === Сравнение выражений ===
2845  function exprEq(a, b) {
2846      if (a === b) return true;
2847      if (!a && !b) return true;
2848      if (!a || !b) return false;
2849      if (a.type !== b.type) return false;
2850
2851      switch (a.type) {
2852          case 'const': return a.n === b.n;
2853          case 'var': return a.name === b.name;
2854          case 'op': return a.op === b.op && exprEq(a.l, b.l) && exprEq(a.r, b.r);
2855          case 'when': return condEq(a.c, b.c) && exprEq(a.t, b.t) && exprEq(a.e, b.e);
2856          default: return false;
```

```
    }
}

// === Упрощение выражений ===
function simplifyExpr(expr) {
    _depth++;
    if (_depth > MAX_DEPTH) {
        _depth--;
        return expr;
    }

    try {
        return simplifyExprCore(expr);
    } finally {
        _depth--;
    }
}

function simplifyExprCore(expr) {
    if (!expr || expr.kind !== 'expr') return expr;

    switch (expr.type) {
        case 'const':
        case 'var':
            return expr;

        case 'op': {
            const l = simplifyExprCore(expr.l);
            const r = simplifyExprCore(expr.r);

            if (expr.op === '+') {
                if (r?.type === 'const' && r.n === 0) return l;
                if (l?.type === 'const' && l.n === 0) return r;
            }
            if (expr.op === '*') {
                if (l?.type === 'const' && l.n === 0) return Const(0);
                if (r?.type === 'const' && r.n === 0) return Const(0);
                if (l?.type === 'const' && l.n === 1) return r;
                if (r?.type === 'const' && r.n === 1) return l;
            }
            return Op(expr.op, l, r);
        }

        case 'when': {
            const c = simplifyCond(expr.c);
            const t = simplifyExprCore(expr.t);
            const e = simplifyExprCore(expr.e);

            if (c?.type === 'true') return t;
            if (c?.type === 'false') return e;
            if (exprEq(t, e)) return t;
            // ✅ НОВОЕ: WHEN(C, T, WHEN(NOT C, X, 0)) => WHEN(C, T, X)
            if (e && e.type === 'when') {
                const c2 = simplifyCond(e.c);
                const t2 = simplifyExprCore(e.t);
                const e2 = simplifyExprCore(e.e);

                if (e2?.type === 'const' && e2.n === 0 && condNegationEq(c, c2)) {
                    return When(c, t, t2);
                }
            }
            // Узкое правило: WHEN(A∧B, t1, WHEN(A∧¬B, t2, WHEN(¬A, t3, e3))) -> ...
    t3
            if (e && e.type === 'when') {
                const c2 = e.c, t2 = e.t, e2 = e.e;
```

```
2921
2922                 if (e2 && e2.type === 'when') {
2923                     const c3 = e2.c, t3 = e2.t;
2924
2925                     const shared = findSharedAndComplement(c, c2);
2926                     if (shared && condNegationEq(c3, shared.shared)) {
2927                         return When(c, t, When(c2, t2, t3));
2928                     }
2929                 }
2930             }
2931
2932             return When(c, t, e);
2933         }
2934
2935         default:
2936             return expr;
2937     }
2938 }
2939
2940 // === Печать ===
2941 function printCond(c) {
2942     if (!c) return 'TRUE';
2943
2944     switch (c.type) {
2945         case 'eq0': return `${c.v} = 0`;
2946         case 'ne0': return `${c.v} != 0`;
2947         case 'cmp': return `${c.l} ${c.op} ${c.r}`;
2948         case 'and': return `(${printCond(c.a)} AND ${printCond(c.b)})`;
2949         case 'or':  return `(${printCond(c.a)} OR ${printCond(c.b)})`;
2950         case 'not': return `NOT(${printCond(c.x)})`;
2951         case 'true': return 'TRUE';
2952         case 'false': return 'FALSE';
2953         default: return '?';
2954     }
2955 }
2956
2957 function printExpr(e) {
2958     if (!e) return '0';
2959
2960     switch (e.type) {
2961         case 'const': return String(e.n);
2962         case 'var': return e.name;
2963         case 'op': return `(${printExpr(e.l)}${e.op}${printExpr(e.r)})`;
2964         case 'when': return `WHEN(${printCond(e.c)}, ${printExpr(e.t)}, $
      {printExpr(e.e)})`;
2965         default: return '?';
2966     }
2967 }
2968
2969 window.Optimizer = {
2970     Eq0, Ne0, Cmp, And, Or, Not, TrueCond, FalseCond,
2971     Const, Var, Op, When,
2972     simplifyCond, simplifyExpr,
2973     printCond, printExpr,
2974     condEq, exprEq
2975 };
2976
2977 codegen.js
2978
2979 // js/codegen.js
2980
2981 const CodeGen = {
2982     _cache: {},
2983     _branchCache: {},
2984     _resolveCache: {},
```

```
    _visiting: new Set(),

    reset() {
        this._cache = {};
        this._branchCache = {};
        this._resolveCache = {};
        this._visiting = new Set();
    },

    toExpr(valueStr) {
        const s = String(valueStr).trim();
        if (s === '0') return Optimizer.Const(0);
        const num = parseFloat(s);
        if (!isNaN(num) && String(num) === s) return Optimizer.Const(num);
        return Optimizer.Var(s);
    },

    exprToName(exprAst) {
        if (!exprAst) return '0';
        if (exprAst.type === 'var') return exprAst.name;
        if (exprAst.type === 'const') return String(exprAst.n);
        return Optimizer.printExpr(exprAst);
    },

    mergeCond(a, b) {
        if (!a && !b) return null;
        if (!a) return b;
        if (!b) return a;
        if (Optimizer.condEq && Optimizer.condEq(a, b)) return a;
        return Optimizer.And(a, b);
    },

    getConn(toId, toPort) {
        return AppState.connections.find(c => c.toElement === toId && c.toPort ===
toPort);
    },

    getConns(toId, prefix) {
        return AppState.connections.filter(c => c.toElement === toId &&
c.toPort.startsWith(prefix));
    },

    buildFormulaExpr(elem) {
        let result = elem.props.expression || '0';

        // 1) Сначала раскрываем шаблоны (h и др.)
        const map = (typeof Settings !== 'undefined' && Settings.getTemplatesMap)
            ? Settings.getTemplatesMap()
            : null;
        result = expandFormulaTemplates(result, map);

        // 2) Потом раскрываем ссылки на формулы
        const formulaRefs = result.match(/formula[_-]\d+/g) || [];
        for (const ref of formulaRefs) {
            const refElem = AppState.elements[ref];
            if (refElem && refElem.type === 'formula') {
                const refExpr = this.buildFormulaExpr(refElem);
                result = result.replace(new RegExp(ref, 'g'), `(${refExpr})`);
            }
        }

        return result;
    },

    // === Получить ЧИСТУЮ логику элемента ===
```

```
    getPureLogic(id) {
        const cacheKey = `logic:${id}`;
        if (cacheKey in this._cache) {
            return this._cache[cacheKey];
        }

        const elem = AppState.elements[id];
        if (!elem) return null;

        let logic = null;

        switch (elem.type) {
            case 'if': {
                const leftConn = this.getConn(id, 'in-0');
                const rightConn = this.getConn(id, 'in-1');

                const leftVal = leftConn ? this.getValue(leftConn.fromElement) :
    Optimizer.Const(0);
                const rightVal = rightConn ? this.getValue(rightConn.fromElement) :
    Optimizer.Const(0);

                const op = (elem.props.operator || '=').trim();
                const leftName = this.exprToName(leftVal);
                const rightName = this.exprToName(rightVal);

                const leftZero = leftVal.type === 'const' && leftVal.n === 0;
                const rightZero = rightVal.type === 'const' && rightVal.n === 0;

                switch (op) {
                    case '=':
                        if (rightZero) {
                            logic = Optimizer.Eq0(leftName);
                        } else if (leftZero) {
                            logic = Optimizer.Eq0(rightName);
                        } else {
                            logic = Optimizer.Cmp(leftName, '=', rightName);
                        }
                        break;
                    case '!=':
                        if (rightZero) {
                            logic = Optimizer.Ne0(leftName);
                        } else if (leftZero) {
                            logic = Optimizer.Ne0(rightName);
                        } else {
                            logic = Optimizer.Cmp(leftName, '!=', rightName);
                        }
                        break;
                    case '>':
                    case '<':
                    case '>=':
                    case '<=':
                        logic = Optimizer.Cmp(leftName, op, rightName);
                        break;
                    default:
                        logic = Optimizer.TrueCond;
                }
                break;
            }

            case 'and':
            case 'or': {
                const isAnd = elem.type === 'and';
                const count = elem.props.inputCount || 2;
                let result = null;

```

```
                for (let i = 0; i < count; i++) {
                    const conn = this.getConn(id, `in-${i}`);
                    if (!conn) continue;

                    const val = this.getPureLogic(conn.fromElement);
                    if (!val) continue;

                    if (result === null) {
                        result = val;
                    } else {
                        result = isAnd ? Optimizer.And(result, val) :
    Optimizer.Or(result, val);
                    }
                }
                logic = result || Optimizer.FalseCond;
                break;
            }

            case 'not': {
                const conn = this.getConn(id, 'in-0');
                const inputLogic = conn ? this.getPureLogic(conn.fromElement) : null;
                logic = Optimizer.Not(inputLogic || Optimizer.FalseCond);
                break;
            }

            case 'separator': {
                const conn = this.getConn(id, 'in-0');
                logic = conn ? this.getPureLogic(conn.fromElement) :
    Optimizer.FalseCond;
                break;
            }

            default:
                logic = null;
        }

        // ↓ новая часть: добавляем контекст с cond-порта для логических элементов
        if (elem.type === 'if' || elem.type === 'and' || elem.type === 'or' ||
    elem.type === 'not') {
            const ctx = this.getConditionFromPort(id);
            if (ctx) {
                if (logic) {
                    logic = Optimizer.And(ctx, logic);
                } else {
                    logic = ctx;
                }
            }
        }

        this._cache[cacheKey] = logic;
        return logic;
    },

    // === Получить значение ===
    getValue(id) {
        const elem = AppState.elements[id];
        if (!elem) return Optimizer.Const(0);

        switch (elem.type) {
            case 'input-signal':
                // Имя сигнала или id как Var(...)
                return this.toExpr(elem.props.name || id);

            case 'const':
                return Optimizer.Const(Number(elem.props.value) || 0);
```

```
3173
3174                case 'formula': {
3175                    // Используем текст формулы как выражение
3176                    const exprStr = this.buildFormulaExpr(elem) || '0';
3177                    return this.toExpr(exprStr);
3178                }
3179
3180                default:
3181                    // На всякий случай — даём символическое имя, а не 0
3182                    if (elem.props && typeof elem.props.name === 'string') {
3183                        return this.toExpr(elem.props.name);
3184                    }
3185                    return this.toExpr(id);
3186            }
3187        },
3188
3189        // === Получить ПОЛНОЕ условие для ветки сепаратора ===
3190        getBranchCondition(sepId, fromPort) {
3191            const cacheKey = `${sepId}:${fromPort}`;
3192            if (cacheKey in this._branchCache) {
3193                return this._branchCache[cacheKey];
3194            }
3195
3196            const sep = AppState.elements[sepId];
3197            if (!sep || sep.type !== 'separator') return null;
3198
3199            const inputLogic = this.getPureLogic(sepId);
3200            const sepContext = this.getConditionFromPort(sepId);
3201
3202            let branchLogic;
3203            if (fromPort === 'out-1') {
3204                branchLogic = inputLogic ? Optimizer.Not(inputLogic) : Optimizer.TrueCond;
3205            } else {
3206                branchLogic = inputLogic || Optimizer.TrueCond;
3207            }
3208
3209            let result;
3210            if (sepContext) {
3211                result = Optimizer.And(sepContext, branchLogic);
3212            } else {
3213                result = branchLogic;
3214            }
3215
3216            this._branchCache[cacheKey] = result;
3217            return result;
3218        },
3219
3220        // === Получить условие от cond-порта ===
3221        getConditionFromPort(id) {
3222            const conn = this.getConn(id, 'cond-0');
3223            if (!conn) return null;
3224
3225            const sourceElem = AppState.elements[conn.fromElement];
3226            if (!sourceElem) return null;
3227
3228            if (sourceElem.type === 'separator') {
3229                return this.getBranchCondition(conn.fromElement, conn.fromPort);
3230            }
3231
3232            return this.getPureLogic(conn.fromElement);
3233        },
3234
3235        // === Основная функция разрешения ===
3236        resolve(id) {
3237            if (id in this._resolveCache) {
```

```
            return this._resolveCache[id];
        }

        if (this._visiting.has(id)) {
            return null;
        }
        this._visiting.add(id);

        const elem = AppState.elements[id];
        if (!elem) {
            this._visiting.delete(id);
            return null;
        }

        let result = null;

        try {
            switch (elem.type) {
                case 'input-signal':
                    result = {
                        isValue: true,
                        cond: null,
                        expr: this.toExpr(elem.props.name || id)
                    };
                    break;

                case 'const': {
                    const cond = this.getConditionFromPort(id);
                    result = {
                        isValue: true,
                        cond: cond,
                        expr: Optimizer.Const(Number(elem.props.value) || 0)
                    };
                    break;
                }

                case 'formula': {
                    let cond = this.getConditionFromPort(id);

                    const inConns = this.getConns(id, 'in-');
                    for (const conn of inConns) {
                        const inputNode = this.resolve(conn.fromElement);
                        if (inputNode && inputNode.cond) {
                            cond = this.mergeCond(cond, inputNode.cond);
                        }
                    }

                    const fullExpr = this.buildFormulaExpr(elem);
                    result = {
                        isValue: true,
                        cond: cond,
                        expr: Optimizer.Var(fullExpr)
                    };
                    break;
                }

                default:
                    result = null;
            }
        } finally {
            this._visiting.delete(id);
        }

        this._resolveCache[id] = result;
        return result;
```

```
    },

    generate() {
        console.log('=== Генерация кода (граф) ===');
        this.reset();

        try {
            const outputs = Object.values(AppState.elements).filter(e => e.type === 'output');

            if (outputs.length === 0) {
                return '/* Нет выходов */';
            }

            const allVariants = [];

            for (const out of outputs) {
                const conns = this.getConns(out.id, 'in-');

                for (const conn of conns) {
                    console.log(`\n=== Обработка выхода ${out.id}, вход от ${conn.fromElement} ===`);
                    const graph = CodeGenGraph.buildDependencyGraph(conn.fromElement);
                    const result = CodeGenGraph.evalGraphValue(graph);
                    console.log(`Результат: cond=${Optimizer.printCond(result.cond)}, expr=${Optimizer.printExpr(result.expr)}`);

                    if (!result || !result.expr) continue;

                    const cond = result.cond ? Optimizer.simplifyCond(result.cond) : null;
                    const isZero = result.expr.type === 'const' && result.expr.n === 0;

                    if (isZero && !cond) continue;

                    allVariants.push({
                        cond,
                        expr: result.expr,
                        isZero
                    });
                }
            }

            console.log('Варианты:', allVariants.map(v => ({
                cond: Optimizer.printCond(v.cond),
                expr: Optimizer.printExpr(v.expr)
            })));

            if (allVariants.length === 0) return '0';

            const valueVariants = allVariants.filter(v => !v.isZero || v.cond);
            if (valueVariants.length === 0) return '0';

            let result = Optimizer.Const(0);

            for (let i = valueVariants.length - 1; i >= 0; i--) {
                const v = valueVariants[i];
                if (v.cond) {
                    result = Optimizer.When(v.cond, v.expr, result);
                } else {
                    result = v.expr;
                }
            }

```

```
3363              const simplified = Optimizer.simplifyExpr(result);
3364              return Optimizer.printExpr(simplified);
3365
3366          } catch (err) {
3367              console.error('Ошибка:', err);
3368              return `/* Ошибка: ${err.message} */`;
3369          }
3370      }
3371  };
3372
3373  window.CodeGen = CodeGen;
3374
3375  config.js:
3376
3377      /**
3378   * Конфигурация приложения
3379   */
3380
3381  // Типы сигналов
3382  const SIGNAL_TYPE = {
3383      NUMERIC: 'numeric',   // Числовой сигнал
3384      LOGIC: 'logic',       // Логический (может быть TRUE или FALSE)
3385      TRUE: 'true',         // Явно ИСТИНА
3386      FALSE: 'false',       // Явно ЛОЖЬ
3387      ANY: 'any'            // Любой тип
3388  };
3389
3390  // Типы проекта
3391  const PROJECT_TYPE = {
3392      PARAMETER: 'parameter',
3393      RULE: 'rule'
3394  };
3395
3396  // Конфигурация элементов
3397  const ELEMENT_TYPES = {
3398      'input-signal': {
3399          name: 'Вход',
3400          inputs: 0,
3401          outputs: 1,
3402          outputLabels: ['out'],
3403          outputTypes: [SIGNAL_TYPE.NUMERIC],
3404          color: '#4a90d9',
3405          hasProperties: true,
3406          defaultProps: { name: 'Сигнал', signalType: SIGNAL_TYPE.NUMERIC },
3407          resizable: true,
3408          minWidth: 150,
3409          minHeight: 50
3410      },
3411      'and': {
3412          name: 'И',
3413          inputs: 2,  // По умолчанию 2, но может быть изменено
3414          outputs: 1,
3415          inputLabels: ['A', 'B'],
3416          inputTypes: [SIGNAL_TYPE.LOGIC, SIGNAL_TYPE.LOGIC],
3417          outputLabels: ['результат'],
3418          outputTypes: [SIGNAL_TYPE.LOGIC],
3419          color: '#a855f7',
3420          hasProperties: true,  // ← Теперь есть свойства (для изменения количества
      входов)
3421          resizable: true,
3422          minWidth: 120,
3423          minHeight: 80,
3424          hasConditionPort: true,
3425          conditionPortType: SIGNAL_TYPE.LOGIC,
3426          defaultProps: {
```

```
            inputCount: 2  // ← Новое свойство
        }
    },
    'or': {
        name: 'ИЛИ',
        inputs: 2,  // По умолчанию 2
        outputs: 1,
        inputLabels: ['A', 'B'],
        inputTypes: [SIGNAL_TYPE.LOGIC, SIGNAL_TYPE.LOGIC],
        outputLabels: ['результат'],
        outputTypes: [SIGNAL_TYPE.LOGIC],
        color: '#a855f7',
        hasProperties: true,  // ← Теперь есть свойства
        resizable: true,
        minWidth: 120,
        minHeight: 80,
        hasConditionPort: true,
        conditionPortType: SIGNAL_TYPE.LOGIC,
        defaultProps: {
            inputCount: 2  // ← Новое свойство
        }
    },
    'not': {
        name: 'НЕ',
        inputs: 1,
        outputs: 1,
        inputLabels: ['A'],
        inputTypes: [SIGNAL_TYPE.LOGIC],
        outputLabels: ['¬A'],
        outputTypes: [SIGNAL_TYPE.LOGIC],
        color: '#a855f7',
        hasProperties: true,
        resizable: true,
        minWidth: 100,
        minHeight: 60,
        hasConditionPort: true,
        conditionPortType: SIGNAL_TYPE.LOGIC
    },
    'if': {
        name: 'ЕСЛИ',
        inputs: 2,
        outputs: 1,  // ← Только один выход!
        inputLabels: ['A', 'B'],
        inputTypes: [SIGNAL_TYPE.ANY, SIGNAL_TYPE.ANY],
        outputLabels: ['результат'],  // ← Просто результат
        outputTypes: [SIGNAL_TYPE.LOGIC],  // ← Выход типа LOGIC
        color: '#e94560',
        hasProperties: true,
        defaultProps: { operator: '=' },
        resizable: true,
        minWidth: 120,
        minHeight: 80,
        hasConditionPort: true,
        conditionPortType: SIGNAL_TYPE.LOGIC
    },
    'separator': {  // ← НОВЫЙ ЭЛЕМЕНТ
        name: 'Сепаратор',
        inputs: 1,
        outputs: 2,
        inputLabels: ['сигнал'],
        inputTypes: [SIGNAL_TYPE.LOGIC],
        outputLabels: ['ИСТИНА', 'ЛОЖЬ'],
        outputTypes: [SIGNAL_TYPE.TRUE, SIGNAL_TYPE.FALSE],  // ← TRUE и FALSE
        color: '#f59e0b',
        hasProperties: true,
```

```
            resizable: true,
            minWidth: 120,
            minHeight: 80,
            hasConditionPort: true,
            conditionPortType: SIGNAL_TYPE.LOGIC
        },
        'const': {
            name: 'Константа',
            inputs: 0,
            outputs: 1,
            outputLabels: ['out'],
            outputTypes: [SIGNAL_TYPE.NUMERIC],
            color: '#3b82f6',
            hasProperties: true,
            defaultProps: { value: 0 },
            resizable: true,
            minWidth: 120,
            minHeight: 60,
            hasConditionPort: true,
            conditionPortType: SIGNAL_TYPE.LOGIC
        },
        'formula': {
            name: 'Формула',
            inputs: 2,
            outputs: 1,
            inputLabels: ['in₁', 'in₂'],
            inputTypes: [SIGNAL_TYPE.ANY, SIGNAL_TYPE.ANY],
            outputLabels: ['результат'],
            outputTypes: [SIGNAL_TYPE.NUMERIC],
            color: '#f59e0b',
            hasProperties: true,
            resizable: true,
            minWidth: 140,
            minHeight: 80,
            defaultProps: {
                expression: '',
                inputCount: 2
            },
            hasConditionPort: true,
            conditionPortType: SIGNAL_TYPE.LOGIC
        },
        'output': {
            name: 'Выход',
            inputs: 1,
            outputs: 0,
            inputLabels: ['сигнал'],
            inputTypes: [SIGNAL_TYPE.ANY],
            color: '#10b981',
            hasProperties: true,
            defaultProps: { label: 'Выход', outputGroup: '' },
            resizable: true,
            minWidth: 150,
            minHeight: 60,
        },  // ← важно, если предыдущий элемент не заканчивается запятой
        'group': {
            name: 'Группа',
            inputs: 0,
            outputs: 0,
            color: '#6b7280',
            resizable: true,
            minWidth: 200,
            minHeight: 120,
            hasProperties: true,
            defaultProps: { title: 'Группа' }
        }
```

```
};

const VIEWPORT_CONFIG = {
    minZoom: 0.1,
    maxZoom: 3,
    zoomStep: 0.1,
    panSpeed: 1,
    canvasWidth: 5000,
    canvasHeight: 5000
};

const MINIMAP_CONFIG = {
    width: 200,
    height: 150,
    padding: 10
};

connections.js:

/**
 * Модуль работы с соединениями
 */

const Connections = {
    /**
     * Настройка обработчиков порта
     */
    setupPortHandlers(port) {
        port.addEventListener('mousedown', (e) => {
            e.stopPropagation();

            if (port.classList.contains('output')) {
                const elemId = port.dataset.element;
                const portName = port.dataset.port;
                const signalType = getOutputPortType(elemId, portName);

                AppState.connectingFrom = {
                    element: elemId,
                    port: portName
                };
                AppState.connectingFromType = signalType;

                this.highlightCompatiblePorts(signalType);

                const svg = document.getElementById('connections-svg');
                const startPos = this._getPortCanvasCenter(port);

                AppState.tempLine = document.createElementNS('http://www.w3.org/2000/svg', 'path');
                AppState.tempLine.setAttribute('class', 'temp-connection');
                AppState.tempLine.setAttribute('d', `M ${startPos.x} ${startPos.y} L ${startPos.x} ${startPos.y}`);
                svg.appendChild(AppState.tempLine);
            }
        });

        port.addEventListener('mouseup', (e) => {
            e.stopPropagation();
            e.preventDefault();

            if (AppState.connectingFrom && port.classList.contains('input')) {
                const toElement = port.dataset.element;
                const toPortName = port.dataset.port;
                const inputType = getInputPortType(toElement, toPortName);

```

```
                if (!areTypesCompatible(AppState.connectingFromType, inputType)) {
                    this.clearConnectionState();
                    return;
                }

                if (AppState.connectingFrom.element !== toElement) {
                    const targetElem = AppState.elements[toElement];
                    const allowMultipleInputs = targetElem?.type === 'output';

                    const exists = AppState.connections.some(c =>
                        c.toElement === toElement && c.toPort === toPortName
                    );

                    if (!exists || allowMultipleInputs) {
                        AppState.connections.push({
                            fromElement: AppState.connectingFrom.element,
                            fromPort: AppState.connectingFrom.port,
                            toElement,
                            toPort: toPortName,
                            signalType: AppState.connectingFromType
                        });

                        port.classList.add('connected');
                        this.drawConnections();
                        this.clearConnectionState();
                        return;
                    }
                }
            }

            this.clearConnectionState();
        });

        port.addEventListener('mouseenter', () => {
            if (AppState.connectingFrom && port.classList.contains('input')) {
                const toPortName = port.dataset.port;
                const inputType = getInputPortType(port.dataset.element, toPortName);

                if (!areTypesCompatible(AppState.connectingFromType, inputType)) {
                    if (AppState.tempLine) {
                        AppState.tempLine.classList.add('invalid');
                    }
                }
            }
        });

        port.addEventListener('mouseleave', () => {
            if (AppState.tempLine) {
                AppState.tempLine.classList.remove('invalid');
            }
        });
    },

    /**
     * Подсветка совместимых портов
     */
    highlightCompatiblePorts(signalType) {
        document.querySelectorAll('.port.input').forEach(port => {
            const inputType = getInputPortType(port.dataset.element,
  port.dataset.port);

            if (areTypesCompatible(signalType, inputType)) {
                port.classList.add('compatible-highlight');
            } else {
                port.classList.add('incompatible');
```

```
            }
        });
    },

    /**
     * Очистка состояния соединения
     */
    clearConnectionState() {
        if (AppState.tempLine) {
            AppState.tempLine.remove();
            AppState.tempLine = null;
        }
        AppState.connectingFrom = null;
        AppState.connectingFromType = null;

        document.querySelectorAll('.port').forEach(port => {
            port.classList.remove('compatible-highlight', 'incompatible');
        });
    },

    /**
     * Отрисовка временной линии соединения
     */
    drawTempConnection(e) {
    if (!AppState.tempLine || !AppState.connectingFrom) return;

    const fromElem = document.getElementById(AppState.connectingFrom.element);
    if (!fromElem) return;

    const fromPort = fromElem.querySelector(`[data-port="${AppState.connectingFrom.port}"]`);
    if (!fromPort) return;

    const startPos = this._getPortCanvasCenter(fromPort);
    const endPos = screenToCanvas(e.clientX, e.clientY);

    const horizontalDist = Math.abs(endPos.x - startPos.x);
    const controlDist = Math.max(horizontalDist * 0.4, 50);

    // Тянем всегда от выхода (вектор 1, 0)
    const cx1 = startPos.x + controlDist;
    const cy1 = startPos.y;

    // Вторая точка контроля для плавности за курсором
    const cx2 = endPos.x - controlDist;
    const cy2 = endPos.y;

    AppState.tempLine.setAttribute('d', `M ${startPos.x} ${startPos.y} C ${cx1} ${cy1}, ${cx2} ${cy2}, ${endPos.x} ${endPos.y}`);
    AppState.tempLine.setAttribute('fill', 'none');
},

    /**
     * Отрисовка всех соединений
     */
    drawConnections() {
    const svg = document.getElementById('connections-svg');

    // 1. Очистка старых линий
    svg.querySelectorAll('path:not(.temp-connection)').forEach(p => p.remove());

    // 2. Сброс визуального состояния портов
    document.querySelectorAll('.port.connected').forEach(port => {
        port.classList.remove('connected');
    });
```

```
3747
3748      // 3. Перебор всех соединений из AppState
3749      AppState.connections.forEach(conn => {
3750          const fromElem = document.getElementById(conn.fromElement);
3751          const toElem = document.getElementById(conn.toElement);
3752
3753          if (!fromElem || !toElem) return;
3754
3755          const fromPort = fromElem.querySelector(`[data-port="${conn.fromPort}"]`);
3756          const toPort = toElem.querySelector(`[data-port="${conn.toPort}"]`);
3757
3758          if (!fromPort || !toPort) return;
3759
3760          fromPort.classList.add('connected');
3761          toPort.classList.add('connected');
3762
3763          const startPos = this._getPortCanvasCenter(fromPort);
3764          const endPos = this._getPortCanvasCenter(toPort);
3765
3766          if (!startPos || !endPos) return;
3767
3768          // Расстояние для изгиба кривой
3769          const horizontalDist = Math.abs(endPos.x - startPos.x);
3770          const verticalDist = Math.abs(endPos.y - startPos.y);
3771          const controlDist = Math.max(horizontalDist * 0.4, 50);
3772
3773          // --- ЛОГИКА ГЕОМЕТРИИ (Вектора касательных) ---
3774          let d;
3775          let cx1 = startPos.x;
3776          let cy1 = startPos.y;
3777          let cx2 = endPos.x;
3778          let cy2 = endPos.y;
3779
3780          // ВЫХОД (Source): Касательная (1, 0) -> Всегда вправо
3781          cx1 = startPos.x + controlDist;
3782          cy1 = startPos.y;
3783
3784          // ВХОД (Target):
3785          if (conn.toPort === 'cond-0') {
3786              // Технический порт: Касательная (0, 1) в декартовой (вверх)
3787              // В экранных координатах Y инвертирован, поэтому отнимаем от Y
3788              cx2 = endPos.x;
3789              cy2 = endPos.y - controlDist; // Линия заходит сверху вертикально
3790          } else {
3791              // Обычный вход: Касательная (-1, 0) -> Слева направо
3792              cx2 = endPos.x - controlDist;
3793              cy2 = endPos.y;
3794          }
3795
3796          d = `M ${startPos.x} ${startPos.y} C ${cx1} ${cy1}, ${cx2} ${cy2}, ${endPos.x}
     ${endPos.y}`;
3797
3798          const path = document.createElementNS('http://www.w3.org/2000/svg', 'path');
3799          path.setAttribute('d', d);
3800          path.setAttribute('fill', 'none'); // Чтобы не было черных полигонов
3801
3802          // --- ЛОГИКА ЦВЕТА (Классы) ---
3803          let cssClass = 'connection';
3804          const type = conn.signalType;
3805
3806          // Приоритет новым типам сигналов
3807          if (type === SIGNAL_TYPE.TRUE) cssClass += ' true-conn';
3808          else if (type === SIGNAL_TYPE.FALSE) cssClass += ' false-conn';
3809          else if (type === SIGNAL_TYPE.LOGIC) cssClass += ' logic-conn';
3810          else if (type === SIGNAL_TYPE.NUMERIC) cssClass += ' numeric-conn';
```

```
3811          else if (type === SIGNAL_TYPE.ANY) cssClass += ' any-conn';
3812
3813          path.setAttribute('class', cssClass);
3814
3815          // Обработчики событий
3816          path.style.pointerEvents = 'stroke';
3817          path.style.cursor = 'pointer';
3818          path.addEventListener('click', () => this.handleConnectionClick(conn));
3819
3820          svg.appendChild(path);
3821      });
3822
3823      if (typeof Outputs !== 'undefined' && Outputs.updateOutputStatus) {
3824          Outputs.updateOutputStatus();
3825      }
3826      Viewport.updateMinimap();
3827  },
3828      /**
3829       * Обработка клика по соединению (удаление)
3830       */
3831      handleConnectionClick(conn) {
3832          if (confirm('Удалить соединение?')) {
3833              AppState.connections = AppState.connections.filter(c =>
3834                  !(c.fromElement === conn.fromElement &&
3835                    c.fromPort === conn.fromPort &&
3836                    c.toElement === conn.toElement &&
3837                    c.toPort === conn.toPort)
3838              );
3839
3840              this.drawConnections();
3841          }
3842      },
3843
3844      /**
3845       * Получение центра порта в координатах Canvas
3846       */
3847      _getPortCanvasCenter(portEl) {
3848          if (!portEl) return null;
3849
3850          const rect = portEl.getBoundingClientRect();
3851          return screenToCanvas(
3852              rect.left + rect.width / 2,
3853              rect.top + rect.height / 2
3854          );
3855      }
3856  };
3857
3858  elements.js:
3859
3860  /**
3861   * Модуль работы с элементами схемы
3862   */
3863
3864  const Elements = {
3865      /**
3866       * Генерация HTML для элемента
3867       */
3868          createElementHTML(elemType, elemId, x, y, props = {}, width, height) {
3869              const config = ELEMENT_TYPES[elemType];
3870              if (!config) throw new Error(`Неизвестный тип элемента: ${elemType}`);
3871
3872              const safe = (value, fallback = '') => (value === null || value ===
      undefined) ? fallback : String(value);
3873              const w = width ?? config.minWidth ?? 120;
3874              const h = height ?? config.minHeight ?? 60;
```

```
3875
3876            const getPortClass = (signalType, direction) => {
3877                const base = direction === 'output' ? 'port output' : 'port input';
3878                if (signalType === SIGNAL_TYPE.LOGIC) return `${base} logic-port`;
3879                if (signalType === SIGNAL_TYPE.NUMBER) return `${base} number-port`;
3880                return `${base} any-port`;
3881            };
3882
3883            // Эта функция buildConditionPort будет вызываться ИНАЧЕ, а не внутри
     innerHTML
3884            // Она тут остается, но ее результат не встраивается в HTML-строку
     напрямую, кроме формулы
3885            const buildConditionPortHTML = () => {
3886                return `
3887                    <div class="condition-port-wrapper">
3888                        <div class="condition-port-label">условие</div>
3889                        <div class="port input condition-port"
3890                            data-port="cond-0"
3891                            data-element="${elemId}"
3892                            data-signal-type="${SIGNAL_TYPE.LOGIC}"
3893                            title="Техническое условие">
3894                        </div>
3895                    </div>`;
3896            };
3897
3898
3899            const buildInputPorts = (count, types = [], labels = []) => {
3900                let html = '';
3901                for (let i = 0; i < count; i++) {
3902                    const type = types[i] ?? types[types.length - 1] ??
     SIGNAL_TYPE.ANY;
3903                    html += `<div class="${getPortClass(type, 'input')}" data-
     port="in-${i}" data-element="${elemId}" data-signal-type="${type}" title="${labels[i]
     || `Вход ${i+1}`}"></div>`;
3904                }
3905                return html;
3906            };
3907
3908            const buildOutputPorts = (count, types = [], labels = []) => {
3909                let html = '';
3910                for (let i = 0; i < count; i++) {
3911                    const type = types[i] ?? types[types.length - 1] ??
     SIGNAL_TYPE.ANY;
3912                    html += `<div class="${getPortClass(type, 'output')}" data-
     port="out-${i}" data-element="${elemId}" data-signal-type="${type}" title="${labels[i]
     || `Выход ${i+1}`}"></div>`;
3913                }
3914                return html;
3915            };
3916
3917            const resizeHandles = config.resizable ? `<div class="resize-handle
     handle-se" data-direction="se"></div><div class="resize-handle handle-e" data-
     direction="e"></div><div class="resize-handle handle-s" data-direction="s"></div>` :
     '';
3918            // hasCondClass будет добавляться в addElement
3919            // const hasCondClass = config.hasConditionPort ? 'has-condition-port' :
     '';
3920
3921            let innerHTML = '';
3922
3923            if (elemType === 'input-signal') {
3924                const name = safe(props.name, 'Сигнал');
3925                const type = props.signalType || SIGNAL_TYPE.NUMBER;
3926                const symbol = type === SIGNAL_TYPE.LOGIC ? '🔀' : '🔢';
3927                innerHTML = `
```

```
                    <div class="element-header" style="background:${config.color};">Источник</div>
                    <div class="element-body">
                        <div class="element-symbol">
                            <span class="input-signal-icon">${symbol}</span>
                            <span class="input-signal-name">${name}</span>
                        </div>
                        <div class="ports-right">
                            ${buildOutputPorts(1, [type], ['Выход'])}
                        </div>
                    </div>`;
            }
            else if (elemType === 'const') {
                innerHTML = `
                    <div class="element-header" style="background:${config.color};">Константа</div>
                    <div class="element-body">
                        <div class="element-symbol">${props.value ?? 0}</div>
                        <div class="ports-right">
                            ${buildOutputPorts(1, [SIGNAL_TYPE.NUMBER], ['Значение'])}
                        </div>
                    </div>`;
            }
            else if (elemType === 'separator') {
                innerHTML = `
                    <div class="element-header" style="background:${config.color};">Сепаратор</div>
                    <div class="element-body">
                        <div class="ports-left">${buildInputPorts(1, config.inputTypes, config.inputLabels)}</div>
                        <div class="element-symbol">✓/✗</div>
                        <div class="ports-right">
                            <div class="port output logic-port true-port" data-port="out-0" data-element="${elemId}" data-signal-type="${SIGNAL_TYPE.TRUE}" title="ИСТИНА"></div>
                            <div class="port output logic-port false-port" data-port="out-1" data-element="${elemId}" data-signal-type="${SIGNAL_TYPE.FALSE}" title="ЛОЖЬ"></div>
                        </div>
                    </div>`;
            }
            else if (elemType === 'and' || elemType === 'or') {
                const gateSymbol = elemType === 'and' ? '∧' : '∨';
                const inputCount = props.inputCount || config.defaultProps?.inputCount || 2;

                // Генерируем динамические входы
                let inputsHTML = '';
                for (let i = 0; i < inputCount; i++) {
                    inputsHTML += `<div class="port input logic-port" data-port="in-${i}" data-element="${elemId}" data-signal-type="${SIGNAL_TYPE.LOGIC}" title="Вход ${i+1}"></div>`;
                }

                innerHTML = `
                    <div class="element-header" style="background:${config.color};">${config.name}</div>
                    <div class="element-body">
                        <div class="ports-left">
                            ${inputsHTML}
                        </div>
                        <div class="element-symbol">${gateSymbol}</div>
                        <div class="ports-right">
                            <div class="port output logic-port" data-port="out-0" data-element="${elemId}" data-signal-type="${SIGNAL_TYPE.LOGIC}" title="Результат"></
```

```
      div>
3980                          </div>
3981                      </div>`;
3982              }
3983              else if (elemType === 'if') {
3984                  const op = safe(props.operator, '=');
3985                  innerHTML = `
3986                      <div class="element-header" style="background:$
      {config.color};">Условие</div>
3987                      <div class="element-body">
3988                          <div class="ports-left">${buildInputPorts(2,
      config.inputTypes, config.inputLabels)}</div>
3989                          <div class="element-symbol">${op}</div>
3990                          <div class="ports-right">
3991                              ${buildOutputPorts(1, [SIGNAL_TYPE.LOGIC], ['результат'])}
3992                          </div>
3993                      </div>`;
3994              }
3995              else if (elemType === 'not') {
3996                  innerHTML = `
3997                      <div class="element-header" style="background:$
      {config.color};">НЕ</div>
3998                      <div class="element-body">
3999                          <div class="ports-left">${buildInputPorts(1,
      [SIGNAL_TYPE.LOGIC], ['A'])}</div>
4000                          <div class="element-symbol">¬</div>
4001                          <div class="ports-right">
4002                              ${buildOutputPorts(1, [SIGNAL_TYPE.LOGIC], ['¬A'])}
4003                          </div>
4004                      </div>`;
4005              }
4006              else if (elemType === 'formula') {
4007                  const inputCount = props.inputCount || config.defaultProps?.inputCount
      || config.inputs || 2;
4008                  const expression = safe(props.expression);
4009                  const displayExpression = expression
4010                      ? (expression.length > 12 ? `${expression.slice(0, 12)}…` :
      expression)
4011                      : 'f(x)';
4012
4013                  innerHTML = `
4014                      ${buildConditionPortHTML()}
4015                      <div class="element-header" style="background:$
      {config.color};">Формула</div>
4016                      <div class="element-body">
4017                          <div class="ports-left">${buildInputPorts(inputCount,
      config.inputTypes, config.inputLabels)}</div>
4018                          <div class="element-symbol">${displayExpression}</div>
4019                          <div class="ports-right">
4020                              ${buildOutputPorts(1, [SIGNAL_TYPE.NUMBER],
      ['Результат'])}
4021                          </div>
4022                      </div>`;
4023              }
4024              else if (elemType === 'output') {
4025                  innerHTML = `
4026                      <div class="element-header" style="background:$
      {config.color};">Выход</div>
4027                      <div class="element-body">
4028                          <div class="ports-left">
4029                              ${buildInputPorts(1, [SIGNAL_TYPE.ANY], ['сигнал'])}
4030                          </div>
4031                          <div class="element-symbol">${safe(props.label, 'Выход')}</
      div>
4032                          <div class="ports-right"></div>
```

```
                    </div>`;

            }
            else if (elemType === 'group') {
                const title = props.title || 'Группа';
                innerHTML = `
                    <div class="group-content">
                    <div class="group-title">${title}</div>
                    </div>`;
            }

            else { // Для любых других (fallback)
                innerHTML = `
                    <div class="element-header" style="background:${config.color};">${config.name}</div>
                    <div class="element-body">
                        <div class="ports-left">${buildInputPorts(config.inputs || 0, config.inputTypes, config.inputLabels)}</div>
                        <div class="element-symbol">${config.name}</div>
                        <div class="ports-right">
                            ${buildOutputPorts(config.outputs || 0, config.outputTypes, config.outputLabels)}
                        </div>
                    </div>`;
            }
            const commentHtml = `<div class="element-comment">${safe(props.comment, '')}</div>`;


            const html = `
                <div class="element ${elemType}" id="${elemId}"
                    style="left:${x}px; top:${y}px; width:${w}px; height:${h}px;" data-type="${elemType}">
                    ${innerHTML}
                    ${commentHtml}
                    ${resizeHandles}
                </div>`;

            return { html, width: w, height: h };
        },

    /**
     * Добавление элемента
     */
        addElement(elemType, x, y, props = {}, elemId = null, customWidth = null, customHeight = null) {
            const config = ELEMENT_TYPES[elemType];
            if (!config) {
                console.error(`Неизвестный тип элемента: ${elemType}`);
                return null;
            }

            if (!elemId) {
                elemId = `${elemType}_${++AppState.elementCounter}`;
            }

            let width = customWidth;
            let height = customHeight;

            if (width === null || width === undefined) {
                width = config.minWidth || 140;
            }
            if (height === null || height === undefined) {
                height = config.minHeight || 70;
            }
```

```
4092
4093              try {
4094                  const result = this.createElementHTML(elemType, elemId, x, y, props,
      width, height);
4095                  if (!result || !result.html) {
4096                      console.error('createElementHTML вернул пустой результат');
4097                      return null;
4098                  }
4099
4100                  const workspace = document.getElementById('workspace');
4101                  const wrapper = document.createElement('div');
4102                  wrapper.innerHTML = result.html.trim();
4103                  const element = wrapper.firstElementChild;
4104                  if (!element) {
4105                      console.error('Не удалось создать DOM элемент из HTML');
4106                      return null;
4107                  }
4108
4109                  // Добавляем класс для отступа
4110                  if (config.hasConditionPort) {
4111                      element.classList.add('has-condition-port');
4112                  }
4113
4114                  workspace.appendChild(element);
4115
4116                  AppState.elements[elemId] = {
4117                      id: elemId,
4118                      type: elemType,
4119                      x,
4120                      y,
4121                      width: result.width || width,
4122                      height: result.height || height,
4123                      props: { ...(config.defaultProps || {}), ...(props || {}) }
4124                  };
4125
4126                  // ЕСЛИ У ЭЛЕМЕНТА ЕСТЬ COND-ПОРТ (И ОН НЕ ФОРМУЛА, КОТОРАЯ УЖЕ ИМЕЕТ
      ЕГО В HTML)
4127                  if (config.hasConditionPort && elemType !== 'formula') {
4128                      const condPortWrapper = document.createElement('div');
4129                      condPortWrapper.innerHTML = `
4130                          <div class="condition-port-wrapper">
4131                              <div class="condition-port-label">условие</div>
4132                              <div class="port input condition-port"
4133                                  data-port="cond-0"
4134                                  data-element="${elemId}"
4135                                  data-signal-type="${SIGNAL_TYPE.LOGIC}"
4136                                  title="Техническое условие">
4137                              </div>
4138                          </div>`;
4139                      element.prepend(condPortWrapper.firstElementChild); // Вставляем в
      самое начало элемента
4140                  }
4141
4142
4143                  this.setupElementHandlers(elemId); // Передаем ID элемента
4144
4145                  // Порты инициализируются внутри setupElementHandlers, нет нужды здесь
4146                  // element.querySelectorAll('.port').forEach(port => {
4147                  //     Connections.setupPortHandlers(port);
4148                  // });
4149
4150                  Connections.drawConnections(); // Перерисовываем соединения, чтобы
      учесть новые порты
4151                  Viewport.updateMinimap();
4152                  return elemId;
```

```
            } catch (err) {
                console.error(`Ошибка при добавлении элемента ${elemType}:`, err);
                return null;
            }
        },

    /**
     * Обновление входов логического элемента (AND, OR)
     */
    updateLogicGateInputs(elemId, inputCount) {
        const elem = document.getElementById(elemId);
        if (!elem) return;

        const portsLeft = elem.querySelector('.ports-left');
        if (!portsLeft) return;

        // Удаляем соединения к портам, которые больше не существуют
        AppState.connections = AppState.connections.filter(c => {
            if (c.toElement === elemId && c.toPort.startsWith('in-')) {
                const portNum = parseInt(c.toPort.split('-')[1], 10);
                return portNum < inputCount;
            }
            return true;
        });

        // Генерируем новые входы
        let inputsHTML = '';
        for (let i = 0; i < inputCount; i++) {
            inputsHTML += `
                <div class="port input logic-port"
                    data-port="in-${i}"
                    data-element="${elemId}"
                    data-signal-type="${SIGNAL_TYPE.LOGIC}"
                    title="Вход ${i+1}">
                </div>
            `;
        }
        portsLeft.innerHTML = inputsHTML;

        // Переподключаем обработчики
        portsLeft.querySelectorAll('.port').forEach(port =>
            Connections.setupPortHandlers(port)
        );

        Connections.drawConnections();
    },

    /**
     * Удаление элемента
     */
    deleteElement(elemId) {
        AppState.connections = AppState.connections.filter(c =>
            c.fromElement !== elemId && c.toElement !== elemId
        );

        const elem = document.getElementById(elemId);
        if (elem) elem.remove();

        delete AppState.elements[elemId];

        if (AppState.selectedElement === elemId) {
            AppState.selectedElement = null;
        }

        Connections.drawConnections();
```

```
        Viewport.updateMinimap();
    },

    /**
     * Выделение элемента
     */
    selectElement(elemId) {
        if (AppState.selectedElement) {
            const oldElem = document.getElementById(AppState.selectedElement);
            if (oldElem) oldElem.classList.remove('selected');
        }

        AppState.selectedElement = elemId;
        const elem = document.getElementById(elemId);
        if (elem) elem.classList.add('selected');

        const elemData = AppState.elements[elemId];
        if (elemData) {
            document.getElementById('selection-info').textContent =
                `Выбрано: ${ELEMENT_TYPES[elemData.type]?.name || elemData.type}`;
        }
    },

    /**
     * Снять выделение
     */
    deselectAll() {
        if (AppState.selectedElement) {
            const elem = document.getElementById(AppState.selectedElement);
            if (elem) elem.classList.remove('selected');
            AppState.selectedElement = null;
        }
        document.getElementById('selection-info').textContent = '';
    },

    /**
     * Настройка обработчиков элемента
     */
    setupElementHandlers(elemId) {
        try {
            const elem = document.getElementById(elemId);
            if (!elem) return;

            elem.addEventListener('mousedown', (e) => {
                if (e.target.classList.contains('port')) return;
                if (e.target.classList.contains('resize-handle')) return;

                e.preventDefault();
                e.stopPropagation();

                this.selectElement(elemId);

                AppState.draggingElement = elemId;
                const canvasPos = screenToCanvas(e.clientX, e.clientY);
                const elemData = AppState.elements[elemId];
                AppState.dragOffset.x = canvasPos.x - elemData.x;
                AppState.dragOffset.y = canvasPos.y - elemData.y;
            });

            elem.addEventListener('dblclick', (e) => {
                if (e.target.classList.contains('port')) return;
                const config = ELEMENT_TYPES[AppState.elements[elemId].type];
                if (config?.hasProperties) {
                    Modal.showPropertiesModal(elemId);
                }
```

```
            });

            elem.addEventListener('contextmenu', (e) => {
                e.preventDefault();
                this.showContextMenu(e.clientX, e.clientY, elemId);
            });

            const handles = elem.querySelectorAll('.resize-handle');
            handles.forEach(handle => this.setupResizeHandlers(handle, elemId));

            const ports = elem.querySelectorAll('.port');
            ports.forEach(port => Connections.setupPortHandlers(port));

        } catch (err) {
            console.error('setupElementHandlers error for', elemId, err);
        }
    },

    /**
     * Контекстное меню
     */
    showContextMenu(x, y, elemId) {
        const menu = document.getElementById('context-menu');
        menu.style.left = `${x}px`;
        menu.style.top = `${y}px`;
        menu.style.display = 'block';
        menu.dataset.elementId = elemId;
    },

    /**
     * Настройка resize
     */
    setupResizeHandlers(handle, elemId) {
        handle.addEventListener('mousedown', (e) => {
            e.stopPropagation();
            e.preventDefault();

            const elemData = AppState.elements[elemId];

            AppState.resizing = {
                elemId: elemId,
                handle: handle.dataset.direction,
                startX: e.clientX,
                startY: e.clientY,
                startWidth: elemData.width,
                startHeight: elemData.height,
                startLeft: elemData.x,
                startTop: elemData.y
            };
        });
    },

    /**
     * Обработка resize
     */
    handleResize(e) {
        if (!AppState.resizing) return;

        const { elemId, handle, startX, startY, startWidth, startHeight, startLeft,
    startTop } = AppState.resizing;
        const elem = document.getElementById(elemId);
        const elemData = AppState.elements[elemId];
        const config = ELEMENT_TYPES[elemData.type];

        const dx = (e.clientX - startX) / AppState.viewport.zoom;
```

```
4347          const dy = (e.clientY - startY) / AppState.viewport.zoom;
4348
4349          let newWidth = startWidth;
4350          let newHeight = startHeight;
4351          let newLeft = startLeft;
4352          let newTop = startTop;
4353
4354          if (handle.includes('e')) {
4355              newWidth = Math.max(config.minWidth, startWidth + dx);
4356          }
4357          if (handle.includes('w')) {
4358              newWidth = Math.max(config.minWidth, startWidth - dx);
4359              newLeft = startLeft + (startWidth - newWidth);
4360          }
4361          if (handle.includes('s')) {
4362              newHeight = Math.max(config.minHeight, startHeight + dy);
4363          }
4364          if (handle.includes('n')) {
4365              newHeight = Math.max(config.minHeight, startHeight - dy);
4366              newTop = startTop + (startHeight - newHeight);
4367          }
4368
4369          elem.style.width = `${newWidth}px`;
4370          elem.style.height = `${newHeight}px`;
4371          elem.style.left = `${newLeft}px`;
4372          elem.style.top = `${newTop}px`;
4373
4374          elemData.width = newWidth;
4375          elemData.height = newHeight;
4376          elemData.x = newLeft;
4377          elemData.y = newTop;
4378
4379          Connections.drawConnections();
4380      },
4381
4382      /**
4383       * Обработка перетаскивания элемента
4384       */
4385      handleDrag(e) {
4386          if (!AppState.draggingElement) return;
4387
4388          const canvasPos = screenToCanvas(e.clientX, e.clientY);
4389          const elemId = AppState.draggingElement;
4390          const elemData = AppState.elements[elemId];
4391          if (!elemData) return;
4392
4393          const newX = canvasPos.x - AppState.dragOffset.x;
4394          const newY = canvasPos.y - AppState.dragOffset.y;
4395          const dx = newX - elemData.x;
4396          const dy = newY - elemData.y;
4397
4398          // если выделено несколько
4399          const group = AppState.selectedElements && AppState.selectedElements.length >
      1
4400              ? AppState.selectedElements
4401              : [elemId];
4402
4403          for (const id of group) {
4404              const elData = AppState.elements[id];
4405              if (!elData) continue;
4406              elData.x += dx;
4407              elData.y += dy;
4408              const el = document.getElementById(id);
4409              if (el) {
4410              el.style.left = elData.x + 'px';
```

```
            el.style.top = elData.y + 'px';
            }
        }

        Connections.drawConnections();
    },

    /**
     * Обновление входов формулы
     */
    updateFormulaInputs(elemId, inputCount) {
        const elem = document.getElementById(elemId);
        if (!elem) return;

        const portsLeft = elem.querySelector('.ports-left');
        if (!portsLeft) return;

        AppState.connections = AppState.connections.filter(c => {
            if (c.toElement === elemId && c.toPort.startsWith('in-')) {
                const portNum = parseInt(c.toPort.split('-')[1], 10);
                return portNum < inputCount;
            }
            return true;
        });

        let inputsHTML = '';
        for (let i = 0; i < inputCount; i++) {
            inputsHTML += `
                <div class="port input any-port"
                     data-port="in-${i}"
                     data-element="${elemId}"
                     data-signal-type="${SIGNAL_TYPE.ANY}"
                     title="in${i} (Любой)">
                </div>
            `;
        }
        portsLeft.innerHTML = inputsHTML;

        portsLeft.querySelectorAll('.port').forEach(port =>
            Connections.setupPortHandlers(port)
        );

        Connections.drawConnections();
    },

    /**
     * Рассчитать оптимальный размер элемента на основе количества портов
     */
    calculateOptimalHeight(elemId, inputCount, outputCount = 1) {
        const elem = AppState.elements[elemId];
        if (!elem) return null;

        const config = ELEMENT_TYPES[elem.type];
        if (!config || !config.resizable) return null;

        // Базовая высота
        let baseHeight = config.minHeight || 60;

        // Каждый порт требует примерно 25-30px высоты
        const portSpacing = 28;
        const maxPorts = Math.max(inputCount, outputCount);

        // Добавляем высоту для портов (кроме первого, который уже в baseHeight)
        const additionalHeight = (maxPorts - 1) * portSpacing;
        const newHeight = Math.max(baseHeight, baseHeight + additionalHeight);
```

```
4476
4477          return newHeight;
4478      },
4479
4480      /**
4481       * Обновление размера элемента при изменении портов
4482       */
4483      updateElementSize(elemId) {
4484          const elem = document.getElementById(elemId);
4485          const elemData = AppState.elements[elemId];
4486
4487          if (!elem || !elemData) return;
4488
4489          const config = ELEMENT_TYPES[elemData.type];
4490          if (!config || !config.resizable) return;
4491
4492          let inputCount = 0;
4493          let outputCount = config.outputs || 1;
4494
4495          // Определяем количество входов
4496          if (elemData.type === 'and' || elemData.type === 'or' || elemData.type ===
      'formula') {
4497              inputCount = elemData.props.inputCount || config.inputs || 2;
4498          } else {
4499              inputCount = config.inputs || 0;
4500          }
4501
4502          // Рассчитываем новую высоту
4503          const newHeight = this.calculateOptimalHeight(elemId, inputCount,
      outputCount);
4504
4505          if (newHeight && newHeight !== elemData.height) {
4506              elemData.height = newHeight;
4507              elem.style.height = `${newHeight}px`;
4508
4509              // Перерисовываем соединения, т.к. изменился размер элемента
4510              Connections.drawConnections();
4511              Viewport.updateMinimap();
4512          }
4513      }
4514
4515
4516  };
4517
4518  modal.js:
4519  /**
4520   * Модуль модальных окон
4521   */
4522
4523  const Modal = {
4524      /**
4525       * Инициализация модальных окон
4526       */
4527      init() {
4528          // Модальное окно свойств элемента
4529          document.getElementById('modal-save').addEventListener('click', () => {
4530              this.saveElementProperties();
4531          });
4532
4533          document.getElementById('modal-cancel').addEventListener('click', () => {
4534              this.hideModal('modal-overlay');
4535          });
4536
4537          document.getElementById('modal-overlay').addEventListener('click', (e) => {
4538              if (e.target.id === 'modal-overlay') {
```

```
                this.hideModal('modal-overlay');
            }
        });

        // Модальное окно свойств проекта
        document.getElementById('project-modal-save').addEventListener('click', () => {
            this.saveProjectProperties();
        });

        document.getElementById('project-modal-cancel').addEventListener('click', () => {
            this.hideModal('project-modal-overlay');
        });

        document.getElementById('project-modal-overlay').addEventListener('click', (e) => {
            if (e.target.id === 'project-modal-overlay') {
                this.hideModal('project-modal-overlay');
            }
        });
    },

    /**
     * Показать модальное окно
     */
    showModal(modalId) {
        document.getElementById(modalId).style.display = 'flex';
    },

    /**
     * Скрыть модальное окно
     */
    hideModal(modalId) {
        document.getElementById(modalId).style.display = 'none';
    },

    /**
     * Показать свойства элемента
     */
    showPropertiesModal(elemId) {
        const elemData = AppState.elements[elemId];
        const elemType = elemData.type;
        const props = elemData.props;
        const config = ELEMENT_TYPES[elemType];

        const modalOverlay = document.getElementById('modal-overlay');
        const modalTitle = document.getElementById('modal-title');
        const modalContent = document.getElementById('modal-content');

        modalTitle.textContent = `Свойства: ${config.name}`;

        let contentHTML = '';

 if (elemType === 'input-signal') {
  const signalType = props.signalType || SIGNAL_TYPE.NUMBER;

  contentHTML = `
    <div class="modal-row">
      <label>Название сигнала:</label>
      <input type="text" id="prop-name" value="${props.name || ''}" placeholder="Например: 10LBA..." />
      <small style="color:#999;">
        Поиск по маске через * (например: *MAA*CP*)
      </small>
```

```
4600        <div id="signal-filter-results"
4601             style="max-height:160px; overflow-y:auto; background:#0f3460; border-
      radius:5px; margin-top:6px; display:none;">
4602        </div>
4603      </div>
4604
4605      <div class="modal-row">
4606        <label>Описание сигнала:</label>
4607        <textarea id="prop-description" readonly>${props.description || ''}</textarea>
4608      </div>
4609
4610      <div class="modal-row">
4611        <label>Тип сигнала:</label>
4612        <select id="prop-signal-type">
4613          <option value="${SIGNAL_TYPE.NUMBER}" ${signalType === SIGNAL_TYPE.NUMBER ?
      'selected' : ''}>Числовой</option>
4614          <option value="${SIGNAL_TYPE.LOGIC}" ${signalType === SIGNAL_TYPE.LOGIC ?
      'selected' : ''}>Логический</option>
4615        </select>
4616      </div>
4617    `;
4618
4619    // ВАЖНО: обработчики можно навесить только после того, как модалка вставила HTML в
      DOM.
4620    // Поэтому ниже мы добавим "хуки" после того, как modalContent.innerHTML применится.
4621    // (Смотри пункт 2 — небольшая вставка в конце showPropertiesModal)
4622  } else if (elemType === 'if') {
4623              contentHTML = `
4624                  <div class="modal-row">
4625                      <label>Оператор сравнения:</label>
4626                      <select id="prop-operator">
4627                          <option value="=" ${props.operator === '=' ? 'selected' : ''}
      >=  (равно)</option>
4628                          <option value=">" ${props.operator === '>' ? 'selected' : ''}
      >>  (больше)</option>
4629                          <option value="<" ${props.operator === '<' ? 'selected' : ''}
      ><  (меньше)</option>
4630                          <option value=">=" ${props.operator === '>=' ? 'selected' :
      ''}>= (больше или равно)</option>
4631                          <option value="<=" ${props.operator === '<=' ? 'selected' :
      ''}>= (меньше или равно)</option>
4632                          <option value="!=" ${props.operator === '!=' ? 'selected' :
      ''}>!= (не равно)</option>
4633                      </select>
4634                  </div>
4635              `;
4636          } else if (elemType === 'and' || elemType === 'or') {
4637              contentHTML = `
4638                  <div class="modal-row">
4639                      <label>Количество входов:</label>
4640                      <input type="number" id="prop-input-count" value="$
      {props.inputCount || 2}" min="2" max="10">
4641                  </div>
4642                  <div class="modal-row">
4643                      <p style="color: #aaa; font-size: 12px;">
4644                          Измените количество входных портов для этого логического
      элемента.
4645                          Лишние соединения будут автоматически удалены.
4646                      </p>
4647                  </div>
4648              `;
4649          } else if (elemType === 'const') {
4650              contentHTML = `
4651                  <div class="modal-row">
4652                      <label>Значение:</label>
```

```
                    <input type="number" id="prop-value" value="${props.value ?? 0}" step="any">
                </div>
            `;
        }
        else if (elemType === 'group') {
            contentHTML = `
            <div class="modal-row">
                <label>Название группы:</label>
                <input type="text" id="prop-title" value="${props.title || 'Группа'}">
            </div>`;
        }

        else if (elemType === 'formula') {
            let signalsHTML = '';
            AppState.connections.forEach(conn => {
                if (conn.toElement === elemId) {
                    const fromElem = AppState.elements[conn.fromElement];
                    if (fromElem) {
                        const signalName = fromElem.props?.name || fromElem.id;
                        signalsHTML += `<div class="signal-item" data-signal="${signalName}">${signalName} (${conn.toPort})</div>`;
                    }
                }
            });

            // ... (где-то выше код сбора signalsHTML) ...

            contentHTML = `
                <div class="modal-row">
                    <label>Количество входов:</label>
                    <input type="number" id="prop-input-count" value="${props.inputCount || 2}" min="1" max="10">
                </div>

                <!-- Верхний блок: Две колонки (Сигналы и Шаблоны) -->
                <div style="display: flex; gap: 15px; margin-bottom: 15px; height: 140px;">
                    <!-- Левая колонка: Сигналы -->
                    <div style="flex: 1; display: flex; flex-direction: column;">
                        <label style="margin-bottom: 5px; display:block;">Входные сигналы:</label>
                        <div class="signal-list" id="signal-list" style="flex: 1; overflow-y: auto; background: #0f3460; padding: 5px; border-radius: 4px; border: 1px solid #4a90d9;">
                            ${signalsHTML || '<div style="color:#888;padding:5px;">Нет сигналов</div>'}
                        </div>
                    </div>

                    <!-- Правая колонка: Шаблоны -->
                    <div style="flex: 1; display: flex; flex-direction: column;">
                        <label style="margin-bottom: 5px; display:block;">Шаблоны:</label>
                        <div class="signal-list" id="template-list" style="flex: 1; overflow-y: auto; background: #0f3460; padding: 5px; border-radius: 4px; border: 1px solid #4a90d9;">
                            <div style="color:#888;padding:5px;">Загрузка…</div>
                        </div>
                    </div>
                </div>

                <!-- Нижний блок: Поле формулы (во всю ширину) -->
                <div class="modal-row">
                    <label>Выражение формулы:</label>
```

```
                    <textarea id="prop-expression"
                            style="width: 100%; min-height: 80px; font-family:
    monospace; font-size: 14px; line-height: 1.4;"
                            spellcheck="false">${props.expression || ''}</textarea>
                    <small style="color:#999; display:block; margin-top:4px;">
                        Двойной клик по сигналу или шаблону вставит его в позицию
    курсора (или заменит выделенный текст).
                    </small>
                </div>
            `;
        }
        if (!contentHTML) {
            contentHTML = `<div style="color:#aaa; font-size:12px;">Нет специальных
    свойств.</div>`;
            }
        contentHTML += `
            <div class="modal-row">
                <label>Комментарий:</label>
                <textarea id="prop-comment" placeholder="Комментарий к элементу...">$
    {props.comment || ''}</textarea>
            </div>
            `;


        modalContent.innerHTML = contentHTML;
        if (elemType === 'formula') {
            const listEl = document.getElementById('template-list');
            (async () => {
                try {
                const data = await Settings.fetchFormulaTemplates();
                const items = data.templates || [];
                if (!items.length) {
                    listEl.innerHTML = '<div style="color:#888;padding:5px;">Нет
    шаблонов</div>';
                    return;
                }
                listEl.innerHTML = items.map(t => {
                    const sig = `${t.name}(${(t.args || []).join(', ')})`;
                    return `<div class="signal-item template-item" data-insert="$
    {sig}">${sig}</div>`;
                }).join('');

                listEl.querySelectorAll('.template-item').forEach(div => {
                    div.addEventListener('dblclick', () => {
                        const insert = div.dataset.insert;
                        const textarea = document.getElementById('prop-expression');

                        // БЫЛО: textarea.value += ...;
                        // СТАЛО:
                        insertAtCursor(textarea, insert);
                    });
                });
                } catch (e) {
                console.error(e);
                listEl.innerHTML = '<div style="color:#888;padding:5px;">Ошибка
    загрузки</div>';
                }
            })();
        }


        // --- post init handlers (когда DOM модалки уже существует) ---
        if (elemType === 'input-signal') {
            const input = document.getElementById('prop-name');
```

```
            const results = document.getElementById('signal-filter-results');
            const descField = document.getElementById('prop-description');

            let timer = null;

            const renderList = (items) => {
                if (!items || items.length === 0) {
                results.innerHTML = '<div style="color:#666;padding:6px;">Нет совпадений</div>';
                results.style.display = 'block';
                return;
                }

                results.innerHTML = items.map(s => `
                <div class="signal-result-item"
                    style="padding:6px 8px; cursor:pointer; border-bottom:1px solid rgba(255,255,255,0.08);">
                    <div style="font-weight:600;">${s.Tagname}</div>
                    <div style="color:#aaa; font-size:11px;">${s.Description || ''}</div>
                </div>
                `).join('');

                results.style.display = 'block';

                results.querySelectorAll('.signal-result-item').forEach((div, i) => {
                div.addEventListener('click', () => {
                    const chosen = items[i];
                    input.value = chosen.Tagname;
                    descField.value = chosen.Description || '';
                    results.style.display = 'none';
                });
                });
            };

            const search = async () => {
                const mask = (input.value || '').trim();

                // Показываем список только если пользователь реально использует маску
                if (!mask.includes('*')) {
                results.style.display = 'none';
                return;
                }

                results.innerHTML = '<div style="color:#666;padding:6px;">Поиск...</div>';
                results.style.display = 'block';

                try {
                // В settings.js должен быть метод Settings.fetchSignals(mask, limit)
                const data = await Settings.fetchSignals(mask, 50);
                renderList(data.items || []);
                } catch (e) {
                results.innerHTML = '<div style="color:#666;padding:6px;">Ошибка загрузки сигналов</div>';
                results.style.display = 'block';
                console.error(e);
                }
            };

            input.addEventListener('input', () => {
                clearTimeout(timer);
                timer = setTimeout(search, 200); // debounce
            });

```

```
            // опционально: закрывать список кликом вне
            document.addEventListener('mousedown', (e) => {
                if (!results.contains(e.target) && e.target !== input) {
                results.style.display = 'none';
                }
            }, { once: true });
        }
        modalOverlay.dataset.elementId = elemId;
        this.showModal('modal-overlay');

        // Функция для умной вставки текста в позицию курсора
        const insertAtCursor = (field, text) => {
            if (!field) return;

            // Получаем позиции выделения
            const startPos = field.selectionStart;
            const endPos = field.selectionEnd;
            const currentValue = field.value;

            // Вставляем текст: (текст до) + (новый текст) + (текст после)
            field.value = currentValue.substring(0, startPos) +
                        text +
                        currentValue.substring(endPos, currentValue.length);

            // Возвращаем фокус и ставим курсор сразу после вставленного текста
            field.focus();
            const newCursorPos = startPos + text.length;
            field.setSelectionRange(newCursorPos, newCursorPos);
        };

        // Обработчик вставки сигналов для формулы
        if (elemType === 'formula') {
            document.querySelectorAll('.signal-item').forEach(item => {
            item.addEventListener('dblclick', () => {
                const signal = item.dataset.signal;
                const textarea = document.getElementById('prop-expression');

                // БЫЛО: textarea.value += signal;
                // СТАЛО:
                insertAtCursor(textarea, signal);
                });
            });
        }
    },

    /**
     * Сохранить свойства элемента
     */
/**
 * Сохранить свойства элемента
 */
    saveElementProperties() {
        try {
            const modalOverlay = document.getElementById('modal-overlay');
            const elemId = modalOverlay.dataset.elementId;
            const elemData = AppState.elements[elemId];
            const elem = document.getElementById(elemId);
            if (!elemData) {
                alert('⚠ Элемент не найден — возможно, он был удалён или
переименован.');
                console.warn(`saveElementProperties: элемент ${elemId} не найден.`);
                this.hideModal('modal-overlay');
                return;
            }

```

```
            const elemType = elemData.type;

            if (elemType === 'input-signal') {
                const name = document.getElementById('prop-name').value || 'Сигнал';
                const description = document.getElementById('prop-description').value || '';
                const signalType = document.getElementById('prop-signal-type').value;

                const oldSignalType = elemData.props.signalType;
                elemData.props.name = name;
                elemData.props.description = description;
                elemData.props.signalType = signalType;

                if (oldSignalType !== signalType) {
                    AppState.connections = AppState.connections.filter(conn => {
                    if (conn.fromElement === elemId) {
                        const toPortIndex = parseInt(conn.toPort.split('-')[1]);
                        const inputType = getInputPortType(conn.toElement, toPortIndex);
                        return areTypesCompatible(signalType, inputType);
                    }
                    return true;
                    });
                }

                const { html } = Elements.createElementHTML(
                    elemType, elemId, elemData.x, elemData.y, elemData.props, elemData.width, elemData.height
                );
                elem.outerHTML = html;

                Elements.setupElementHandlers(elemId);
                Connections.drawConnections();
            } else if (elemType === 'if') {
                const operator = document.getElementById('prop-operator').value;
                elemData.props.operator = operator;
                const symbol = elem.querySelector('.element-symbol');
                if (symbol) symbol.textContent = operator;

            } else if (elemType === 'const') {
                const value = parseFloat(document.getElementById('prop-value').value) || 0;
                elemData.props.value = value;
                const symbol = elem.querySelector('.element-symbol');
                if (symbol) symbol.textContent = String(value);

            } else if (elemType === 'formula') {
                const expression = document.getElementById('prop-expression').value;
                const inputCount = parseInt(document.getElementById('prop-input-count').value) || 2;

                elemData.props.expression = expression;
                elemData.props.inputCount = inputCount;

                const symbol = elem.querySelector('.element-symbol');
                if (symbol) {
                    symbol.textContent = expression.length > 12 ? `${expression.slice(0, 12)}…` : (expression || 'f(x)');
                }

                Elements.updateFormulaInputs(elemId, inputCount);
                Elements.updateElementSize(elemId);  // ← Добавляем это
            } else if (elemType === 'and' || elemType === 'or') {
                const inputCount = parseInt(document.getElementById('prop-input-count').value) || 2;
```

```
                elemData.props.inputCount = inputCount;

                Elements.updateLogicGateInputs(elemId, inputCount);
                Elements.updateElementSize(elemId);  // ← Добавляем это

                const symbol = elem.querySelector('.element-symbol');
                if (symbol) {
                    symbol.textContent = elemType === 'and' ? '∧' : '∨';
                }

        } else if (elemType === 'output') {
                const label = document.getElementById('prop-label').value || 'Выход';
                const outputGroup = document.getElementById('prop-output-group').value
    || '';

                elemData.props.label = label;
                elemData.props.outputGroup = outputGroup;

                const symbol = elem.querySelector('.element-symbol');
                if (symbol) symbol.textContent = label;
            }
            else if (elemType === 'group') {
                const title = document.getElementById('prop-title').value || 'Группа';
                elemData.props.title = title;
                const titleEl = elem.querySelector('.group-title');
                if (titleEl) titleEl.textContent = title;
                }
            const commentEl = document.getElementById('prop-comment');
            if (commentEl) elemData.props.comment = commentEl.value || '';

            this.hideModal('modal-overlay');

        } catch (error) {
            console.error('❌ Ошибка при сохранении свойств:', error);
            alert('Ошибка сохранения: ' + error.message);
        }
    },

    /**
     * Показать свойства проекта
     */
    showProjectPropertiesModal() {
        const content = document.getElementById('project-modal-content');
        const project = AppState.project;

        // Генерируем HTML для списка выходов только если модуль загружен
        let outputsHtml = '';
        if (typeof Outputs !== 'undefined' && AppState.outputs) {
            const logicalOutputsHtml = AppState.outputs.logical.length > 0
                ? AppState.outputs.logical.map(output => `
                    <div class="output-item"
                        data-element-id="${output.elementId}"
                        onmouseenter="Outputs.highlightOutput('${output.elementId}',
    true)"
                        onmouseleave="Outputs.highlightOutput('${output.elementId}',
    false)"
                        onclick="Outputs.navigateToOutput('${output.elementId}');
    Modal.hideModal('project-modal-overlay');">
                        <span class="output-icon">${output.portLabel === 'Да' ? '✅' :
    '❌'}</span>
                        <span class="output-name">${output.elementName}</span>
                        <span class="output-port">→ ${output.portLabel}</span>
                    </div>
                `).join('')
                : '<div class="no-outputs">Нет логических выходов</div>';
```

```
5007
5008            const numericOutputsHtml = AppState.outputs.numeric.length > 0
5009                ? AppState.outputs.numeric.map(output => `
5010                    <div class="output-item numeric"
5011                        data-element-id="${output.elementId}"
5012                        onmouseenter="Outputs.highlightOutput('${output.elementId}',
    true)"
5013                        onmouseleave="Outputs.highlightOutput('${output.elementId}',
    false)"
5014                        onclick="Outputs.navigateToOutput('${output.elementId}');
    Modal.hideModal('project-modal-overlay');">
5015                        <span class="output-icon">🔢</span>
5016                        <span class="output-name">${output.elementName}</span>
5017                        <span class="output-port">→ значение</span>
5018                    </div>
5019                `).join('')
5020                : '<div class="no-outputs">Нет числовых выходов</div>';
5021
5022            outputsHtml = `
5023                <div class="modal-row">
5024                    <label>Выходные сигналы схемы:</label>
5025                    <div class="outputs-container">
5026                        <div class="outputs-section">
5027                            <div class="outputs-section-title">
5028                                <span class="section-icon">🔀</span>
5029                                Логические выходы (${AppState.outputs.logical.length})
5030                            </div>
5031                            <div class="outputs-list">
5032                                ${logicalOutputsHtml}
5033                            </div>
5034                        </div>
5035                        <div class="outputs-section">
5036                            <div class="outputs-section-title">
5037                                <span class="section-icon">📐</span>
5038                                Числовые выходы (${AppState.outputs.numeric.length})
5039                            </div>
5040                            <div class="outputs-list">
5041                                ${numericOutputsHtml}
5042                            </div>
5043                        </div>
5044                    </div>
5045                    <div class="outputs-hint">
5046                        💡 Выходами автоматически становятся элементы, чьи выходные
    порты не подключены к другим элементам.
5047                        Кликните на выход, чтобы перейти к нему на схеме.
5048                    </div>
5049                </div>
5050            `;
5051        }
5052
5053        content.innerHTML = `
5054            <div class="modal-row">
5055                <label>Код проекта:</label>
5056                <input type="text" id="project-code" value="${project.code || ''}"
    placeholder="Уникальный идентификатор">
5057            </div>
5058
5059            <div class="modal-row">
5060                <label>Тип проекта:</label>
5061                <div class="project-type-selector">
5062                    <div class="project-type-btn ${project.type ===
    PROJECT_TYPE.PARAMETER ? 'active' : ''}" data-type="${PROJECT_TYPE.PARAMETER}">
5063                        <div class="type-icon">📊</div>
5064                        <div class="type-name">Параметр</div>
5065                        <div class="type-desc">Вычисляемое значение</div>
```

```
                    </div>
                    <div class="project-type-btn ${project.type === 
PROJECT_TYPE.RULE ? 'active' : ''}" data-type="${PROJECT_TYPE.RULE}">
                        <div class="type-icon">📋</div>
                        <div class="type-name">Правило</div>
                        <div class="type-desc">Логическое условие</div>
                    </div>
                </div>
            </div>

            <div id="parameter-fields" class="conditional-fields ${project.type === 
PROJECT_TYPE.PARAMETER ? 'visible' : ''}">
                <div class="modal-row">
                    <label>Размерность:</label>
                    <input type="text" id="project-dimension" value="$
{project.dimension || ''}" placeholder="Например: м/с, кг, °C">
                </div>
            </div>

            <div id="rule-fields" class="conditional-fields ${project.type === 
PROJECT_TYPE.RULE ? 'visible' : ''}">
                <div class="modal-row">
                    <label>Возможная причина:</label>
                    <textarea id="project-possible-cause" placeholder="Описание 
возможной причины срабатывания правила">${project.possibleCause || ''}</textarea>
                </div>
                <div class="modal-row">
                    <label>Методические указания:</label>
                    <textarea id="project-guidelines" placeholder="Инструкции и 
рекомендации при срабатывании правила">${project.guidelines || ''}</textarea>
                </div>
            </div>

            ${outputsHtml}
        `;

        // Обработчики переключения типа
        content.querySelectorAll('.project-type-btn').forEach(btn => {
            btn.addEventListener('click', () => {
                content.querySelectorAll('.project-type-btn').forEach(b => 
b.classList.remove('active'));
                btn.classList.add('active');

                const type = btn.dataset.type;
                document.getElementById('parameter-
fields').classList.toggle('visible', type === PROJECT_TYPE.PARAMETER);
                document.getElementById('rule-fields').classList.toggle('visible', 
type === PROJECT_TYPE.RULE);
            });
        });

        this.showModal('project-modal-overlay');
    },

    /**
     * Сохранить свойства проекта
     */
    saveProjectProperties() {
        const activeTypeBtn = document.querySelector('.project-type-btn.active');
        const type = activeTypeBtn ? activeTypeBtn.dataset.type : 
PROJECT_TYPE.PARAMETER;

        AppState.project.code = document.getElementById('project-code').value;
        AppState.project.type = type;

```

```
        if (type === PROJECT_TYPE.PARAMETER) {
            AppState.project.dimension = document.getElementById('project-
dimension').value;
            AppState.project.possibleCause = '';
            AppState.project.guidelines = '';
        } else {
            AppState.project.dimension = '';
            AppState.project.possibleCause = document.getElementById('project-
possible-cause').value;
            AppState.project.guidelines = document.getElementById('project-
guidelines').value;
        }

        this.hideModal('project-modal-overlay');
    }
};

output.js:

/**
 * Модуль управления выходными сигналами
 */

const Outputs = {
    /**
     * Обновление статуса выходных элементов
     * Вызывается при каждом изменении схемы
     */
    updateOutputStatus() {
        this.clearAllOutputHighlights();
        AppState.outputs.logical = [];
        AppState.outputs.numeric = [];
        updateFrameChildren();

        // Обработка элементов-выходов
        Object.values(AppState.elements).forEach(elem => {
            if (!elem || elem.type !== 'output') return;

            // Проверяем, к чему подключен вход этого выхода
            const inputConns = AppState.connections.filter(c =>
                c.toElement === elem.id && c.toPort === 'in-0'
            );

            // Каждое соединение к выходу — это отдельный выход
            inputConns.forEach((conn, index) => {
                const fromElem = AppState.elements[conn.fromElement];
                if (!fromElem) return;

                const outputType = conn.signalType;
                const outputInfo = {
                    id: `${elem.id}_conn_${index}`,
                    elementId: elem.id,
                    sourceElementId: conn.fromElement,
                    sourcePort: conn.fromPort,
                    portIndex: 0,
                    portId: 'in-0',
                    type: outputType,
                    label: elem.props?.label || 'Выход',
                    elementType: 'output',
                    elementName: elem.props?.label || 'Выход',
                    name: elem.props?.label || 'Выход'
                };

                if (outputType === SIGNAL_TYPE.LOGIC) {
                    AppState.outputs.logical.push(outputInfo);
```

```
                } else if (outputType === SIGNAL_TYPE.NUMBER) {
                    AppState.outputs.numeric.push(outputInfo);
                }

                // Подсветим входной порт
                this.highlightOutputPort(elem.id, 0, outputType);
            });
        });

        this.updateOutputCounter();
    },

    /**
     * Очистка всех выделений выходов
     */
    clearAllOutputHighlights() {
        document.querySelectorAll('.port.output-active').forEach(port => {
            port.classList.remove('output-active');
        });

        document.querySelectorAll('.element.has-output').forEach(elem => {
            elem.classList.remove('has-output');
        });

        document.querySelectorAll('.element.output-ambiguous').forEach(el =>
    el.classList.remove('output-ambiguous'));
        document.querySelectorAll('.element.output-missing').forEach(el =>
    el.classList.remove('output-missing'));
    },

    /**
     * Выделение выходного порта
     */
    highlightOutputPort(elemId, portIndex, portType) {
        const elem = document.getElementById(elemId);
        if (!elem) return;

        const port = elem.querySelector(`.port.output[data-port="out-${portIndex}"]`);
        if (port) {
            port.classList.add('output-active');
        }

        // Добавляем класс элементу (даёт общий визуал)
        elem.classList.add('has-output');
    },

    /**
     * Обновление счётчика выходов в меню
     */
    updateOutputCounter() {
        const counter = document.getElementById('output-counter');
        if (counter) {
            const total = AppState.outputs.logical.length +
    AppState.outputs.numeric.length;
            counter.textContent = total;
            counter.style.display = total > 0 ? 'inline-block' : 'none';
        }
    },

    /**
     * Получить все выходы для сохранения в проект
     */
    getOutputsForSave() {
        // Сохраняем информацию о frame/inner для рамок
        return {
```

```
            logical: AppState.outputs.logical.map(o => ({
                id: o.id,
                elementId: o.elementId,
                frameId: o.frameId || null,
                innerElementId: o.innerElementId || null,
                portIndex: o.portIndex ?? o.innerPortIndex ?? null,
                portLabel: o.label
            })),
            numeric: AppState.outputs.numeric.map(o => ({
                id: o.id,
                elementId: o.elementId,
                frameId: o.frameId || null,
                innerElementId: o.innerElementId || null,
                portIndex: o.portIndex ?? o.innerPortIndex ?? null,
                portLabel: o.label
            }))
        };
    },

    /**
     * Подсветить конкретный выход (при наведении в списке)
     */
    highlightOutput(elementId, highlight = true) {
        const elem = document.getElementById(elementId);
        if (elem) {
            if (highlight) {
                elem.classList.add('output-highlighted');
            } else {
                elem.classList.remove('output-highlighted');
            }
        }
    },

    /**
     * Перейти к элементу выхода на схеме (elementId — фокусируемый элемент; для рамок это id рамки)
     */
    navigateToOutput(elementId) {
        const elemData = AppState.elements[elementId];
        if (!elemData) return;

        // Центрируем viewport на элементе
        const container = document.getElementById('workspace-container');
        const rect = container.getBoundingClientRect();

        const centerX = elemData.x + elemData.width / 2;
        const centerY = elemData.y + elemData.height / 2;

        AppState.viewport.panX = rect.width / 2 - centerX * AppState.viewport.zoom;
        AppState.viewport.panY = rect.height / 2 - centerY * AppState.viewport.zoom;

        Viewport.updateTransform();

        // Выделяем элемент
        Elements.selectElement(elementId);

        // Временная подсветка
        this.highlightOutput(elementId, true);
        setTimeout(() => this.highlightOutput(elementId, false), 2000);
    }
};

project.js:

/**
```

```
5309   * Модуль управления проектом (сохранение, загрузка)
5310   */
5311
5312  // --- миграция id: '-' -> '_' с обновлением всех ссылок ---
5313  function migrateIdsDashToUnderscore() {
5314    const map = {};
5315
5316    // 1) собрать map старых id → новых
5317    Object.values(AppState.elements).forEach(el => {
5318      if (typeof el.id === 'string' && el.id.includes('-')) {
5319        map[el.id] = el.id.replace(/-/g, '_');
5320      }
5321    });
5322
5323    if (!Object.keys(map).length) return;
5324
5325    // 2) DOM id + data-element
5326    Object.entries(map).forEach(([oldId, newId]) => {
5327      const dom = document.getElementById(oldId);
5328      if (dom) dom.id = newId;
5329
5330      if (dom) {
5331        dom.querySelectorAll('[data-element]').forEach(p => {
5332          if (p.dataset.element === oldId) p.dataset.element = newId;
5333        });
5334      }
5335    });
5336
5337    // 3) AppState.elements ключи
5338    Object.entries(map).forEach(([oldId, newId]) => {
5339      const el = AppState.elements[oldId];
5340      if (!el) return;
5341      el.id = newId;
5342      AppState.elements[newId] = el;
5343      delete AppState.elements[oldId];
5344    });
5345
5346    // 4) connections
5347    AppState.connections.forEach(c => {
5348      if (map[c.fromElement]) c.fromElement = map[c.fromElement];
5349      if (map[c.toElement]) c.toElement = map[c.toElement];
5350    });
5351
5352    // 5) формулы
5353    const escapeRegex = s => s.replace(/[.*+?^${}()|[\]\\]/g, '\\$&');
5354    Object.values(AppState.elements).forEach(el => {
5355      if (el.type === 'formula' && el.props?.expression) {
5356        let expr = el.props.expression;
5357        Object.entries(map).forEach(([oldId, newId]) => {
5358          const re = new RegExp(`(^|[^A-Za-z0-9_])${escapeRegex(oldId)}(?![A-Za-
      z0-9_])`, 'g');
5359          expr = expr.replace(re, (m, p1) => `${p1}${newId}`);
5360        });
5361        el.props.expression = expr;
5362      }
5363    });
5364
5365    // 6) selected + modal
5366    if (map[AppState.selectedElement]) AppState.selectedElement =
      map[AppState.selectedElement];
5367    const modal = document.getElementById('modal-overlay');
5368    if (modal && map[modal.dataset.elementId]) modal.dataset.elementId =
      map[modal.dataset.elementId];
5369  }
5370
```

```
const Project = {
    /**
     * Инициализация
     */
        /**
     * Инициализация
     */
init() {
    document.getElementById('btn-new').addEventListener('click', () =>
this.newProject());
    document.getElementById('btn-save').addEventListener('click', () =>
this.saveProject());
    document.getElementById('btn-load').addEventListener('click', () =>
this.openProjectListModal());
    document.getElementById('btn-project-settings').addEventListener('click', () => {
        Modal.showProjectPropertiesModal();
    });

    // Работа с модалкой выбора проекта
    this.projectList = [];
    this.filteredProjectList = [];
    this.selectedProjectFilename = null;

    document.getElementById('project-cancel').addEventListener('click', () =>
this.closeProjectListModal());
    document.getElementById('project-refresh').addEventListener('click', () =>
this.refreshProjectList());

    document.getElementById('project-load').addEventListener('click', () => {
        if (this.selectedProjectFilename) {
            this.loadProjectFromList(this.selectedProjectFilename);
        }
    });

    document.getElementById('project-search').addEventListener('input', (event) => {
        this.filterProjectList(event.target.value);
    });
},

    /**
     * Новый проект
     */
    newProject() {
        if (Object.keys(AppState.elements).length > 0) {
            if (!confirm('Создать новый проект? Несохранённые изменения будут
потеряны.')) {
                return;
            }
        }

        document.getElementById('workspace').innerHTML = '';
        document.getElementById('connections-svg').innerHTML = '';

        resetState();
        Viewport.updateTransform();
    },

        /**
     * Запрос имени файла и загрузка с сервера
     */
    async loadProjectPrompt() {
        const filename = window.prompt(
            "Введите имя файла проекта для загрузки (с сервера). Пример:
scheme_logic.json",
            AppState.project.code ? `${AppState.project.code}_$
```

```
      {AppState.project.type}.json` : "scheme_type.json"
5429          );
5430
5431          if (!filename) return; // Отмена
5432
5433          try {
5434              // Используем обертку из Settings.js для запроса к /api/project/load
5435              const data = await Settings.loadProject(filename);
5436
5437              // Если загрузка успешна, вызываем основную функцию обработки данных
5438              this._processLoadedData(data);
5439              alert(`Проект "${filename}" успешно загружен с сервера.`);
5440
5441          } catch (error) {
5442              console.error('Ошибка загрузки проекта:', error);
5443              alert(`Ошибка загрузки проекта: ${error.message}`);
5444          }
5445      },
5446
5447      /**
5448       * Сохранение проекта
5449       */
5450          async saveProject() { // !!! Сделать функцию асинхронной (async) !!!
5451          // 1. Проверяем свойства проекта
5452          if (!AppState.project.code) {
5453              Modal.showProjectPropertiesModal();
5454              alert('Пожалуйста, укажите код проекта перед сохранением.');
5455              return;
5456          }
5457
5458          // Обновляем размеры рамок перед сохранением
5459          updateFrameChildren();
5460          // ✅ нормализуем, даже если проект был открыт до фикса
5461          migrateIdsDashToUnderscore();
5462
5463          // ✅ подчистим связи прямо перед сохранением
5464          const exists = (id) => !!AppState.elements[id];
5465          AppState.connections = (AppState.connections || [])
5466            .map(c => ({
5467              ...c,
5468              fromElement: exists(c.fromElement) ? c.fromElement :
      c.fromElement.replace(/-/g, '_'),
5469              toElement: exists(c.toElement) ? c.toElement : c.toElement.replace(/-/g,
      '_')
5470            }))
5471            .filter(c => exists(c.fromElement) && exists(c.toElement))
5472            .filter((c, idx, arr) => {
5473              const key = `${c.fromElement}|${c.fromPort}|${c.toElement}|${c.toPort}`;
5474              return arr.findIndex(x =>
5475                `${x.fromElement}|${x.fromPort}|${x.toElement}|${x.toPort}` === key
5476              ) === idx;
5477            });
5478          // ✅ 1. Генерируем код заранее
5479          let generatedCode = '';
5480          if (typeof CodeGen !== 'undefined' && typeof CodeGen.generate === 'function')
      {
5481              try {
5482                  generatedCode = CodeGen.generate() || '';
5483              } catch (err) {
5484                  console.error('Code generation failed:', err);
5485              }
5486          }
5487
5488          // 2. Сборка объекта проекта
5489          const project = {
```

```
            version: '1.0',
            project: AppState.project,
            elements: AppState.elements,
            connections: AppState.connections,
            counter: AppState.elementCounter,
            viewport: {
                zoom: AppState.viewport.zoom,
                panX: AppState.viewport.panX,
                panY: AppState.viewport.panY
            },
            code: generatedCode
        };

        const filename = `${AppState.project.code || 'scheme'}_${AppState.project.type}.json`;

        // 3. Сохранение на сервер
        try {
            await Settings.saveProject(filename, project);
            alert(`Проект успешно сохранен на сервере как: ${filename}`);
        } catch (error) {
            console.error('Ошибка сохранения проекта:', error);
            alert(`Ошибка сохранения проекта: ${error.message}`);
        }
    },

    async showProjectList() {
        try {
            const result = await Settings.listProjects(); // нужно реализовать в settings.js
            const list = result.projects || [];

            if (list.length === 0) {
                alert('Проекты в папке не найдены.');
                return;
            }

            const choice = window.prompt(
                'Список проектов:\n' + list.map((p, i) => `${i + 1}. ${p.code || p.filename} — ${p.description}`).join('\n') +
                '\n\nВведите номер проекта для загрузки:',
                '1'
            );
            const index = parseInt(choice, 10) - 1;
            if (isNaN(index) || !list[index]) return;

            await this.loadProjectByFilename(list[index].filename);
        } catch (error) {
            console.error(error);
            alert('Не удалось получить список проектов: ' + error.message);
        }
    },

    async loadProjectByFilename(filename) {
        try {
            const data = await Settings.loadProject(filename);
            this._processLoadedData(data);
            alert(`Проект "${filename}" загружен.`);
        } catch (error) {
            console.error(error);
            alert('Ошибка загрузки проекта: ' + error.message);
        }
    },

    openProjectListModal() {
```

```
5552    const modal = document.getElementById('modal-project-list');
5553    modal.classList.remove('hidden');
5554    document.body.classList.add('modal-open'); // если есть такой класс для блокировки
      скролла
5555    this.refreshProjectList();
5556  },
5557
5558  closeProjectListModal() {
5559    const modal = document.getElementById('modal-project-list');
5560    modal.classList.add('hidden');
5561    document.body.classList.remove('modal-open');
5562  },
5563
5564  async refreshProjectList() {
5565    const tbody = document.getElementById('project-list-body');
5566    tbody.innerHTML = `<tr><td colspan="4" class="project-list__empty">Загрузка…</td></
      tr>`;
5567    try {
5568      const result = await Settings.listProjects();
5569      this.projectList = result.projects || [];
5570      this.filteredProjectList = [...this.projectList];
5571      this.renderProjectList();
5572    } catch (err) {
5573      console.error(err);
5574      tbody.innerHTML = `<tr><td colspan="4" class="project-list__empty">Ошибка: $
      {err.message}</td></tr>`;
5575    }
5576  },
5577
5578  renderProjectList() {
5579    const tbody = document.getElementById('project-list-body');
5580    const loadBtn = document.getElementById('project-load');
5581    loadBtn.disabled = true;
5582    this.selectedProjectFilename = null;
5583
5584    if (!this.filteredProjectList.length) {
5585      tbody.innerHTML = `<tr><td colspan="4" class="project-list__empty">Ничего не
      найдено</td></tr>`;
5586      return;
5587    }
5588
5589    tbody.innerHTML = '';
5590    this.filteredProjectList.forEach((item) => {
5591      const tr = document.createElement('tr');
5592      tr.innerHTML = `
5593        <td>${item.filename}</td>
5594        <td>${item.code || ''}</td>
5595        <td>${item.description || ''}</td>
5596        <td>${item.type || ''}</td>
5597      `;
5598      tr.addEventListener('click', () => {
5599        this.highlightRow(tr);
5600        this.selectedProjectFilename = item.filename;
5601        loadBtn.disabled = false;
5602      });
5603      tr.addEventListener('dblclick', () => {
5604        this.highlightRow(tr);
5605        this.selectedProjectFilename = item.filename;
5606        loadBtn.disabled = false;
5607        this.loadProjectFromList(item.filename);
5608      });
5609      tbody.appendChild(tr);
5610    });
5611  },
5612
```

```
  highlightRow(row) {
    const tbody = row.parentElement;
    [...tbody.children].forEach((tr) => tr.classList.remove('selected'));
    row.classList.add('selected');
  },


  // Фильтр по поисковой строке
  filterProjectList(query) {
    const q = (query || '').trim().toLowerCase();
    if (!q) {
      this.filteredProjectList = [...this.projectList];
    } else {
      this.filteredProjectList = this.projectList.filter((item) => {
        return [
          item.filename,
          item.code,
          item.description,
          item.type
        ].some((field) => (field || '').toLowerCase().includes(q));
      });
    }
    this.renderProjectList();
  },

  async loadProjectFromList(filename) {
    try {
      const data = await Settings.loadProject(filename);
      this._processLoadedData(data);
      this.closeProjectListModal();
      alert(`Проект "${filename}" успешно загружен.`);
    } catch (error) {
      console.error(error);
      alert('Ошибка загрузки проекта: ' + error.message);
    }
  },









      /**
       * Загрузка проекта
       */
   _processLoadedData(data) {
    try {
      document.getElementById('workspace').innerHTML = '';
      document.getElementById('connections-svg').innerHTML = '';
      resetState();

      if (data.project) {
        AppState.project = { ...AppState.project, ...data.project };
      }

      AppState.elementCounter = data.counter || 0;

      if (data.viewport) {
        AppState.viewport.zoom = data.viewport.zoom || 1;
        AppState.viewport.panX = data.viewport.panX || 0;
        AppState.viewport.panY = data.viewport.panY || 0;
      }

      const elements = data.elements || {};
      Object.values(elements)
```

```
5678        .filter(e => e.type === 'output-frame')
5679        .forEach(elemData => {
5680          Elements.addElement(
5681            elemData.type,
5682            elemData.x,
5683            elemData.y,
5684            elemData.props,
5685            elemData.id,
5686            elemData.width,
5687            elemData.height
5688          );
5689        });
5690
5691      Object.values(elements)
5692        .filter(e => e.type !== 'output-frame')
5693        .forEach(elemData => {
5694          Elements.addElement(
5695            elemData.type,
5696            elemData.x,
5697            elemData.y,
5698            elemData.props,
5699            elemData.id,
5700            elemData.width,
5701            elemData.height
5702          );
5703        });
5704
5705      AppState.connections = data.connections || [];
5706
5707      // ✅ ВСТАВЬ ЭТОТ БЛОК СРАЗУ ЗДЕСЬ (до вычисления счётчика)
5708      //Object.values(AppState.elements).forEach(e => {
5709      //  if (typeof e.id === 'string') {
5710      //    e.id = e.id.replace(/-/g, '_');
5711      //  }
5712      //  if (e.props?.name) {
5713      //    e.props.name = e.props.name.replace(/-/g, '_');
5714      //  }
5715      //});
5716      // ✅ конец добавленной секции
5717      // ✅ Миграция id: '-' -> '_'
5718      migrateIdsDashToUnderscore();
5719      // ✅ очистка соединений: удалить битые и дубликаты
5720      const exists = (id) => !!AppState.elements[id];
5721
5722      AppState.connections = (AppState.connections || [])
5723        // оставить только те, где оба конца реально существуют
5724        .filter(c => exists(c.fromElement) && exists(c.toElement))
5725        // убрать дубликаты
5726        .filter((c, idx, arr) => {
5727          const key = `${c.fromElement}|${c.fromPort}|${c.toElement}|${c.toPort}`;
5728          return arr.findIndex(x =>
5729            `${x.fromElement}|${x.fromPort}|${x.toElement}|${x.toPort}` === key
5730          ) === idx;
5731        });
5732
5733
5734      // корректно восстанавливаем счётчик
5735      const counterFromFile = Number(data.counter);
5736      AppState.elementCounter = Number.isFinite(counterFromFile) ? counterFromFile : 0;
5737
5738      const maxIdSuffix = Object.values(AppState.elements).reduce((max, el) => {
5739          if (!el?.id) return max;
5740          const match = String(el.id).match(/_(\d+)$/);   // теперь хвост по
      подчёркиванию
5741          const num = match ? parseInt(match[1], 10) : NaN;
```

```
5742          return Number.isFinite(num) ? Math.max(max, num) : max;
5743      }, 0);
5744
5745      AppState.elementCounter = Math.max(AppState.elementCounter, maxIdSuffix);
5746
5747      Viewport.updateTransform();
5748      Connections.drawConnections();
5749      updateFrameChildren();
5750
5751    } catch (e) {
5752      alert('Ошибка обработки данных проекта: ' + e.message);
5753      console.error(e);
5754    }
5755  }
5756  };
5757
5758  settings.js:
5759
5760  const Settings = {
5761    config: null,
5762    templates: null,
5763
5764    async init() {
5765      // тянем настройки (не обязательно, но полезно)
5766      try {
5767        const r = await fetch('/api/settings');
5768        if (r.ok) this.config = await r.json();
5769      } catch (e) {
5770        console.warn('Settings load failed:', e);
5771      }
5772      try {
5773        const t = await this.fetchFormulaTemplates();
5774        this.templates = t.templates || [];
5775      } catch (e) {
5776        this.templates = [];
5777      }
5778    },
5779
5780    getTemplatesMap() {
5781      const map = {};
5782      (this.templates || []).forEach(t => { if (t?.name) map[t.name] = t; });
5783      return map;
5784    },
5785
5786    async fetchSignals(mask, limit = 50) {
5787      const url = `/api/signals?q=${encodeURIComponent(mask || '')}&limit=$
      {encodeURIComponent(limit)}`;
5788      const r = await fetch(url);
5789      if (!r.ok) throw new Error('Failed to fetch signals');
5790      return await r.json(); // {items, total}
5791    },
5792    // ... в объекте Settings
5793
5794    async saveProject(filename, projectData) {
5795      if (!filename.endsWith('.json')) {
5796        filename += '.json';
5797      }
5798      const r = await fetch('/api/project/save', {
5799        method: 'POST',
5800        headers: { 'Content-Type': 'application/json' },
5801        body: JSON.stringify({
5802          filename: filename,
5803          content: projectData
5804        })
5805      });
```

```
      if (!r.ok) throw new Error('Failed to save project');
      return r.json();
    },

    async listProjects() {
      const r = await fetch('/api/project/list');
      if (!r.ok) throw new Error('Failed to list projects');
      return r.json();
    },

    async fetchFormulaTemplates() {
        const r = await fetch('/api/formula-templates');
        if (!r.ok) throw new Error('Failed to fetch formula templates');
        return await r.json(); // {templates:[...]}
    },

    async loadProject(filename) {
      if (!filename.endsWith('.json')) {
        filename += '.json';
      }
      const r = await fetch(`/api/project/load/${encodeURIComponent(filename)}`);
      if (!r.ok) {
          if (r.status === 404) {
               throw new Error(`Project "${filename}" not found (404)`);
          }
          throw new Error('Failed to load project');
      }
      return r.json();
    }

  // ...
  };

  state.js:
  /**
   * Глобальное состояние приложения
   */

  const AppState = {
      // Элементы схемы
      elements: {},
      connections: [],
      elementCounter: 0,

      // Выделение
      selectedElement: null,

      // Перетаскивание
      draggingElement: null,
      dragOffset: { x: 0, y: 0 },
      isDraggingFromPalette: false,
      dragPreview: null,
      dragType: null,

      // Соединения
      connectingFrom: null,
      connectingFromType: null,
      tempLine: null,

      // Resize
      resizing: null,

      // Viewport (масштабирование и перемещение)
      viewport: {
          zoom: 1,
```

```
        panX: 0,
        panY: 0,
        isPanning: false,
        lastMouseX: 0,
        lastMouseY: 0
    },

    // Свойства проекта
    project: {
        code: '',
        type: PROJECT_TYPE.PARAMETER,
        // Для параметра
        dimension: '',
        // Для правила
        possibleCause: '',
        guidelines: ''
    },

    // Выходные сигналы (автоматически определяются)
    outputs: {
        logical: [],    // Логические выходы [{elementId, portIndex, portLabel, ...}]
        numeric: []     // Числовые выходы (формулы)
    }
};

/**
 * Сброс состояния
 */
function resetState() {
    AppState.elements = {};
    AppState.connections = [];
    AppState.elementCounter = 0;
    AppState.selectedElement = null;
    AppState.draggingElement = null;
    AppState.connectingFrom = null;
    AppState.tempLine = null;
    AppState.resizing = null;

    AppState.viewport = {
        zoom: 1,
        panX: 0,
        panY: 0,
        isPanning: false,
        lastMouseX: 0,
        lastMouseY: 0
    };

    AppState.project = {
        code: '',
        type: PROJECT_TYPE.PARAMETER,
        dimension: '',
        possibleCause: '',
        guidelines: ''
    };

    AppState.outputs = {
        logical: [],
        numeric: []
    };
}

utils.js:
/**
 * Вспомогательные функции
 */
```

```
5936
5937  /**
5938   * Генерация уникального ID
5939   */
5940  function generateId() {
5941      AppState.elementCounter++;
5942      return `elem_${AppState.elementCounter}`;
5943  }
5944
5945  function getInputPortType(elementId, portIdentifier) {
5946      const element = AppState.elements[elementId];
5947      if (!element) return SIGNAL_TYPE.ANY;
5948
5949      const config = ELEMENT_TYPES[element.type];
5950      if (!config) return SIGNAL_TYPE.ANY;
5951
5952      let portIndex = portIdentifier;
5953
5954      // Обработка технического порта условия
5955      if (typeof portIdentifier === 'string') {
5956          if (portIdentifier === 'cond-0' && config.hasConditionPort) {
5957              return config.conditionPortType || SIGNAL_TYPE.LOGIC;
5958          }
5959
5960          if (portIdentifier.startsWith('in-')) {
5961              portIndex = parseInt(portIdentifier.split('-')[1], 10);
5962          }
5963      }
5964
5965      if (Number.isNaN(portIndex) || portIndex === null || portIndex === undefined) {
5966          portIndex = 0;
5967      }
5968
5969      // Динамические входы для AND/OR берут тип из конфига
5970      if ((element.type === 'and' || element.type === 'or')) {
5971          return SIGNAL_TYPE.LOGIC;  // Логические элементы всегда ожидают LOGIC на
      входе
5972      }
5973
5974      if (element.type === 'formula') {
5975          return SIGNAL_TYPE.ANY;
5976      }
5977
5978      const types = config.inputTypes || [];
5979      if (types.length === 0) return SIGNAL_TYPE.ANY;
5980
5981      if (portIndex < types.length) {
5982          return types[portIndex] || SIGNAL_TYPE.ANY;
5983      }
5984
5985      return types[types.length - 1] || SIGNAL_TYPE.ANY;
5986  }
5987
5988  function getOutputPortType(elementId, portIdentifier) {
5989      const element = AppState.elements[elementId];
5990      if (!element) return SIGNAL_TYPE.ANY;
5991
5992      const config = ELEMENT_TYPES[element.type];
5993      if (!config) return SIGNAL_TYPE.ANY;
5994
5995      let portIndex = portIdentifier;
5996
5997      if (typeof portIdentifier === 'string') {
5998          if (portIdentifier.startsWith('out-')) {
5999              portIndex = parseInt(portIdentifier.split('-')[1], 10);
```

```
        }
    }

    if (Number.isNaN(portIndex) || portIndex === null || portIndex === undefined) {
        portIndex = 0;
    }

    const types = config.outputTypes || [];
    if (types.length === 0) return SIGNAL_TYPE.ANY;

    if (portIndex < types.length) {
        return types[portIndex] || SIGNAL_TYPE.ANY;
    }

    return types[types.length - 1] || SIGNAL_TYPE.ANY;
}
/**
 * Проверка совместимости типов сигналов
 *
 * Новая логика:
 * - ANY совместим со всем
 * - TRUE совместим с LOGIC, TRUE, ANY
 * - FALSE совместим с LOGIC, FALSE, ANY
 * - LOGIC совместим с LOGIC, TRUE, FALSE, ANY
 * - NUMERIC совместим с NUMERIC, ANY
 */
function areTypesCompatible(outputType, inputType) {
    // Если один из типов ANY - совместимы
    if (outputType === SIGNAL_TYPE.ANY || inputType === SIGNAL_TYPE.ANY) {
        return true;
    }

    // Если типы одинаковые - совместимы
    if (outputType === inputType) {
        return true;
    }

    // TRUE/FALSE совместимы с LOGIC
    if ((outputType === SIGNAL_TYPE.TRUE || outputType === SIGNAL_TYPE.FALSE) &&
        inputType === SIGNAL_TYPE.LOGIC) {
        return true;
    }

    // LOGIC совместим с TRUE/FALSE (в случае если ожидается конкретный тип)
    if (outputType === SIGNAL_TYPE.LOGIC &&
        (inputType === SIGNAL_TYPE.TRUE || inputType === SIGNAL_TYPE.FALSE)) {
        return true;
    }

    return false;
}

/**
 * Проверка, находится ли элемент внутри рамки
 */
function isInsideFrame(elemId, frameId) {
    const elem = AppState.elements[elemId];
    const frame = AppState.elements[frameId];

    if (!elem || !frame || frame.type !== 'output-frame') return false;

    const elemCenterX = elem.x + elem.width / 2;
    const elemCenterY = elem.y + elem.height / 2;

    return elemCenterX > frame.x &&
```

```
            elemCenterX < frame.x + frame.width &&
            elemCenterY > frame.y &&
            elemCenterY < frame.y + frame.height;
}

/**
 * Обновить принадлежность элементов к рамкам
 */
function updateFrameChildren() {
    // Сначала очистим children у рамок и parentFrame у всех элементов
    Object.values(AppState.elements).forEach(elem => {
        if (elem.type === 'output-frame') {
            elem.children = [];
        } else {
            // удаляем parentFrame по умолчанию (пересчитаем ниже)
            if (elem.parentFrame) delete elem.parentFrame;
        }
    });

    // Назначаем принадлежность: для каждого элемента ищем рамку, в которую он
попадает
    Object.values(AppState.elements).forEach(elem => {
        if (!elem || elem.type === 'output-frame') return;

        Object.values(AppState.elements).forEach(frame => {
            if (!frame || frame.type !== 'output-frame') return;

            if (isInsideFrame(elem.id, frame.id)) {
                // добавляем в массив детей рамки
                frame.children.push(elem.id);
                // отмечаем у элемента родительскую рамку
                if (AppState.elements[elem.id]) {
                    AppState.elements[elem.id].parentFrame = frame.id;
                }
            }
        });
    });
}

/**
 * Преобразование координат экрана в координаты холста
 */
function screenToCanvas(screenX, screenY) {
    const container = document.getElementById('workspace-container');
    const rect = container.getBoundingClientRect();

    const x = (screenX - rect.left - AppState.viewport.panX) / AppState.viewport.zoom;
    const y = (screenY - rect.top - AppState.viewport.panY) / AppState.viewport.zoom;

    return { x, y };
}

/**
 * Преобразование координат холста в координаты экрана
 */
function canvasToScreen(canvasX, canvasY) {
    const container = document.getElementById('workspace-container');
    const rect = container.getBoundingClientRect();

    const x = canvasX * AppState.viewport.zoom + AppState.viewport.panX + rect.left;
    const y = canvasY * AppState.viewport.zoom + AppState.viewport.panY + rect.top;

    return { x, y };
}

```

```
6129  /**
6130   * Проверка, является ли порт выходным (не подключен к другим элементам)
6131   */
6132  function isOutputPort(elemId, portIndex) {
6133      const portKey = `out-${portIndex}`;
6134
6135      // Проверяем, есть ли соединения от этого порта
6136      const hasConnection = AppState.connections.some(conn =>
6137          conn.fromElement === elemId && conn.fromPort === portKey
6138      );
6139
6140      return !hasConnection;
6141  }
6142
6143  /**
6144   * Получить информацию о выходном порте
6145   */
6146  function getOutputPortInfo(elemId, portIndex) {
6147      const elem = AppState.elements[elemId];
6148      if (!elem) return null;
6149
6150      const config = ELEMENT_TYPES[elem.type];
6151      if (!config) return null;
6152
6153      return {
6154          elementId: elemId,
6155          elementType: elem.type,
6156          elementName: config.name,
6157          portIndex: portIndex,
6158          portLabel: config.outputLabels?.[portIndex] || `out${portIndex}`,
6159          portType: config.outputTypes?.[portIndex] || SIGNAL_TYPE.ANY,
6160          // Дополнительная информация для идентификации
6161          displayName: `${config.name} → ${config.outputLabels?.[portIndex] || `out$
      {portIndex}`}`
6162      };
6163  }
6164
6165  function splitArgsTopLevel(argStr) {
6166    const out = [];
6167    let cur = '';
6168    let depth = 0;
6169    for (let i = 0; i < argStr.length; i++) {
6170      const ch = argStr[i];
6171      if (ch === '(') depth++;
6172      if (ch === ')') depth--;
6173      if (ch === ',' && depth === 0) {
6174        out.push(cur.trim());
6175        cur = '';
6176      } else {
6177        cur += ch;
6178      }
6179    }
6180    if (cur.trim()) out.push(cur.trim());
6181    return out;
6182  }
6183
6184  function expandFormulaTemplates(expr, templatesMap) {
6185    if (!expr) return expr;
6186    if (!templatesMap) return expr;
6187
6188    // несколько проходов на случай вложенных шаблонов
6189    for (let pass = 0; pass < 10; pass++) {
6190      let changed = false;
6191
6192      expr = expr.replace(/([A-Za-z_]\w*)\s*\(([^()]|\([^()]*\))*\)/g, (match, name) =>
```

```
    {
6193        const tpl = templatesMap[name];
6194        if (!tpl) return match;
6195
6196        // вытащим аргументы вручную: name(....)
6197        const open = match.indexOf('(');
6198        const close = match.lastIndexOf(')');
6199        const inside = match.slice(open + 1, close);
6200
6201        const args = splitArgsTopLevel(inside);
6202        const formal = tpl.args || [];
6203        let body = String(tpl.body || '0');
6204
6205        // если количество аргументов не совпало — не трогаем (лучше так, чем сломать)
6206        if (args.length !== formal.length) return match;
6207
6208        formal.forEach((f, i) => {
6209          const re = new RegExp(`\\b${f}\\b`, 'g');
6210          body = body.replace(re, `(${args[i]})`);
6211        });
6212
6213        changed = true;
6214        return `(${body})`;
6215      });
6216
6217      if (!changed) break;
6218    }
6219
6220    return expr;
6221  }
6222
6223  viewport.js:
6224  /**
6225   * Модуль управления viewport (масштабирование и перемещение)
6226   */
6227
6228  const Viewport = {
6229      /**
6230       * Инициализация viewport
6231       */
6232      init() {
6233          this.setupZoomControls();
6234          this.setupPanning();
6235          this.setupMouseWheel();
6236          this.setupMinimap();
6237          this.setupCursorPosition();
6238          this.updateTransform();
6239          const container = document.getElementById('workspace-container');
6240          const rect = container.getBoundingClientRect();
6241          AppState.viewport.panX = 100; // немного отступить от левого края
6242          AppState.viewport.panY = (rect.height / 2) - 2500 * 0.5 *
    AppState.viewport.zoom;
6243          this.updateTransform();
6244      },
6245
6246      /**
6247       * Настройка кнопок масштабирования
6248       */
6249      setupZoomControls() {
6250          document.getElementById('btn-zoom-in').addEventListener('click', () => {
6251              this.setZoom(AppState.viewport.zoom + VIEWPORT_CONFIG.zoomStep);
6252          });
6253
6254          document.getElementById('btn-zoom-out').addEventListener('click', () => {
6255              this.setZoom(AppState.viewport.zoom - VIEWPORT_CONFIG.zoomStep);
```

```
        });

        document.getElementById('btn-zoom-reset').addEventListener('click', () => {
            this.setZoom(1);
            this.setPan(0, 0);
        });

        document.getElementById('btn-zoom-fit').addEventListener('click', () => {
            this.fitToContent();
        });
    },

    /**
     * Настройка перемещения (pan)
     */
    setupPanning() {
        const container = document.getElementById('workspace-container');

        container.addEventListener('mousedown', (e) => {
            // Средняя кнопка мыши или пробел + левая кнопка
            if (e.button === 1 || (e.button === 0 && e.target === container)) {
                e.preventDefault();
                AppState.viewport.isPanning = true;
                AppState.viewport.lastMouseX = e.clientX;
                AppState.viewport.lastMouseY = e.clientY;
                container.style.cursor = 'grabbing';
            }
        });

        document.addEventListener('mousemove', (e) => {
            if (AppState.viewport.isPanning) {
                const dx = e.clientX - AppState.viewport.lastMouseX;
                const dy = e.clientY - AppState.viewport.lastMouseY;

                this.setPan(
                    AppState.viewport.panX + dx,
                    AppState.viewport.panY + dy
                );

                AppState.viewport.lastMouseX = e.clientX;
                AppState.viewport.lastMouseY = e.clientY;
            }
        });

        document.addEventListener('mouseup', (e) => {
            if (AppState.viewport.isPanning) {
                AppState.viewport.isPanning = false;
                document.getElementById('workspace-container').style.cursor = '';
            }
        });

        // Клавиша пробел для режима перемещения
        document.addEventListener('keydown', (e) => {
            if (e.code === 'Space' && !e.repeat) {
                document.getElementById('workspace-container').style.cursor = 'grab';
            }
        });

        document.addEventListener('keyup', (e) => {
            if (e.code === 'Space') {
                document.getElementById('workspace-container').style.cursor = '';
            }
        });
    },

```

```
6321      /**
6322       * Настройка масштабирования колесом мыши
6323       */
6324      setupMouseWheel() {
6325          const container = document.getElementById('workspace-container');
6326
6327          container.addEventListener('wheel', (e) => {
6328              e.preventDefault();
6329
6330              const rect = container.getBoundingClientRect();
6331              const mouseX = e.clientX - rect.left;
6332              const mouseY = e.clientY - rect.top;
6333
6334              // Позиция мыши на холсте до масштабирования
6335              const canvasPosBeforeX = (mouseX - AppState.viewport.panX) /
      AppState.viewport.zoom;
6336              const canvasPosBeforeY = (mouseY - AppState.viewport.panY) /
      AppState.viewport.zoom;
6337
6338              // Новый масштаб
6339              const delta = e.deltaY > 0 ? -VIEWPORT_CONFIG.zoomStep :
      VIEWPORT_CONFIG.zoomStep;
6340              const newZoom = Math.max(
6341                  VIEWPORT_CONFIG.minZoom,
6342                  Math.min(VIEWPORT_CONFIG.maxZoom, AppState.viewport.zoom + delta)
6343              );
6344
6345              // Корректируем pan, чтобы точка под курсором осталась на месте
6346              const newPanX = mouseX - canvasPosBeforeX * newZoom;
6347              const newPanY = mouseY - canvasPosBeforeY * newZoom;
6348
6349              AppState.viewport.zoom = newZoom;
6350              AppState.viewport.panX = newPanX;
6351              AppState.viewport.panY = newPanY;
6352
6353              this.updateTransform();
6354          }, { passive: false });
6355      },
6356
6357      /**
6358       * Установить масштаб
6359       */
6360      setZoom(zoom) {
6361          const container = document.getElementById('workspace-container');
6362          const rect = container.getBoundingClientRect();
6363
6364          // Центр экрана
6365          const centerX = rect.width / 2;
6366          const centerY = rect.height / 2;
6367
6368          // Позиция центра на холсте
6369          const canvasCenterX = (centerX - AppState.viewport.panX) /
      AppState.viewport.zoom;
6370          const canvasCenterY = (centerY - AppState.viewport.panY) /
      AppState.viewport.zoom;
6371
6372          // Новый масштаб
6373          const newZoom = Math.max(
6374              VIEWPORT_CONFIG.minZoom,
6375              Math.min(VIEWPORT_CONFIG.maxZoom, zoom)
6376          );
6377
6378          // Корректируем pan
6379          AppState.viewport.panX = centerX - canvasCenterX * newZoom;
6380          AppState.viewport.panY = centerY - canvasCenterY * newZoom;
```

```
        AppState.viewport.zoom = newZoom;

        this.updateTransform();
    },

    /**
     * Установить смещение
     */
    setPan(x, y) {
        AppState.viewport.panX = x;
        AppState.viewport.panY = y;
        this.updateTransform();
    },

    /**
     * Вписать содержимое в экран
     */
    fitToContent() {
        const elements = Object.values(AppState.elements);
        if (elements.length === 0) {
            this.setZoom(1);
            this.setPan(0, 0);
            return;
        }

        // Находим границы содержимого
        let minX = Infinity, minY = Infinity;
        let maxX = -Infinity, maxY = -Infinity;

        elements.forEach(elem => {
            minX = Math.min(minX, elem.x);
            minY = Math.min(minY, elem.y);
            maxX = Math.max(maxX, elem.x + elem.width);
            maxY = Math.max(maxY, elem.y + elem.height);
        });

        const contentWidth = maxX - minX;
        const contentHeight = maxY - minY;

        const container = document.getElementById('workspace-container');
        const rect = container.getBoundingClientRect();

        const padding = 50;
        const availableWidth = rect.width - padding * 2;
        const availableHeight = rect.height - padding * 2;

        const zoomX = availableWidth / contentWidth;
        const zoomY = availableHeight / contentHeight;
        const newZoom = Math.min(zoomX, zoomY, 1);

        AppState.viewport.zoom = Math.max(VIEWPORT_CONFIG.minZoom, newZoom);
        AppState.viewport.panX = padding - minX * AppState.viewport.zoom +
    (availableWidth - contentWidth * AppState.viewport.zoom) / 2;
        AppState.viewport.panY = padding - minY * AppState.viewport.zoom +
    (availableHeight - contentHeight * AppState.viewport.zoom) / 2;

        this.updateTransform();
    },

    /**
     * Обновить трансформацию
     */
    updateTransform() {
        const workspace = document.getElementById('workspace');
        const svg = document.getElementById('connections-svg');
```

```
6444
6445          const transform = `translate(${AppState.viewport.panX}px, $
      {AppState.viewport.panY}px) scale(${AppState.viewport.zoom})`;
6446
6447          workspace.style.transform = transform;
6448          svg.style.transform = transform;
6449
6450          // Обновляем отображение масштаба
6451          document.getElementById('zoom-level').textContent = `$
      {Math.round(AppState.viewport.zoom * 100)}%`;
6452
6453          // Обновляем мини-карту
6454          this.updateMinimap();
6455      },
6456
6457      /**
6458       * Настройка мини-карты
6459       */
6460      setupMinimap() {
6461          const minimap = document.getElementById('minimap');
6462          const canvas = document.getElementById('minimap-canvas');
6463
6464          canvas.width = MINIMAP_CONFIG.width;
6465          canvas.height = MINIMAP_CONFIG.height;
6466
6467          // Клик по мини-карте для перемещения
6468          minimap.addEventListener('click', (e) => {
6469              const rect = minimap.getBoundingClientRect();
6470              const x = e.clientX - rect.left;
6471              const y = e.clientY - rect.top;
6472
6473              this.navigateToMinimapPosition(x, y);
6474          });
6475      },
6476
6477      /**
6478       * Обновить мини-карту
6479       */
6480      updateMinimap() {
6481          const canvas = document.getElementById('minimap-canvas');
6482          const ctx = canvas.getContext('2d');
6483          const viewportEl = document.getElementById('minimap-viewport');
6484
6485          // Очищаем
6486          ctx.fillStyle = '#0a0a1a';
6487          ctx.fillRect(0, 0, canvas.width, canvas.height);
6488
6489          // Масштаб мини-карты
6490          const scale = Math.min(
6491              canvas.width / VIEWPORT_CONFIG.canvasWidth,
6492              canvas.height / VIEWPORT_CONFIG.canvasHeight
6493          );
6494
6495          // Рисуем элементы
6496          Object.values(AppState.elements).forEach(elem => {
6497              const x = elem.x * scale;
6498              const y = elem.y * scale;
6499              const w = Math.max(elem.width * scale, 2);
6500              const h = Math.max(elem.height * scale, 2);
6501
6502              ctx.fillStyle = ELEMENT_TYPES[elem.type]?.color || '#4a90d9';
6503              ctx.fillRect(x, y, w, h);
6504          });
6505
6506          // Рисуем viewport
```

```
        const container = document.getElementById('workspace-container');
        const rect = container.getBoundingClientRect();

        const vpX = (-AppState.viewport.panX / AppState.viewport.zoom) * scale;
        const vpY = (-AppState.viewport.panY / AppState.viewport.zoom) * scale;
        const vpW = (rect.width / AppState.viewport.zoom) * scale;
        const vpH = (rect.height / AppState.viewport.zoom) * scale;

        viewportEl.style.left = `${vpX}px`;
        viewportEl.style.top = `${vpY}px`;
        viewportEl.style.width = `${vpW}px`;
        viewportEl.style.height = `${vpH}px`;
    },

    /**
     * Перейти к позиции на мини-карте
     */
    navigateToMinimapPosition(minimapX, minimapY) {
        const canvas = document.getElementById('minimap-canvas');
        const container = document.getElementById('workspace-container');
        const rect = container.getBoundingClientRect();

        const scale = Math.min(
            canvas.width / VIEWPORT_CONFIG.canvasWidth,
            canvas.height / VIEWPORT_CONFIG.canvasHeight
        );

        const canvasX = minimapX / scale;
        const canvasY = minimapY / scale;

        // Центрируем viewport на этой точке
        AppState.viewport.panX = rect.width / 2 - canvasX * AppState.viewport.zoom;
        AppState.viewport.panY = rect.height / 2 - canvasY * AppState.viewport.zoom;

        this.updateTransform();
    },

    /**
     * Отслеживание позиции курсора
     */
    setupCursorPosition() {
        const container = document.getElementById('workspace-container');

        container.addEventListener('mousemove', (e) => {
            const pos = screenToCanvas(e.clientX, e.clientY);
            document.getElementById('cursor-pos').textContent =
                `X: ${Math.round(pos.x)}, Y: ${Math.round(pos.y)}`;
        });
    }
};

styles.css:

* {
    margin: 0;
    padding: 0;
    box-sizing: border-box;
}

body {
    font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
    background: #1a1a2e;
    color: #eee;
    overflow: hidden;
}
```

```

#app {
    display: flex;
    flex-direction: column;
    height: 100vh;
}

/* ====== МЕНЮ ====== */
#menu {
    background: #16213e;
    padding: 10px 20px;
    display: flex;
    gap: 10px;
    align-items: center;
    border-bottom: 2px solid #0f3460;
    z-index: 100;
    flex-wrap: wrap;
}

.menu-btn {
    background: #0f3460;
    color: #eee;
    border: none;
    padding: 8px 16px;
    border-radius: 5px;
    cursor: pointer;
    transition: background 0.3s;
    font-size: 13px;
}

.menu-btn:hover {
    background: #e94560;
}

.menu-separator {
    width: 1px;
    height: 30px;
    background: #0f3460;
    margin: 0 10px;
}

.zoom-controls {
    display: flex;
    align-items: center;
    gap: 8px;
    background: #0a0a1a;
    padding: 5px 10px;
    border-radius: 5px;
}

.zoom-btn {
    width: 30px;
    height: 30px;
    padding: 0;
    font-size: 18px;
    font-weight: bold;
}

#zoom-level {
    min-width: 50px;
    text-align: center;
    font-size: 12px;
    color: #aaa;
}

```

```
/* ====== ОСНОВНАЯ ОБЛАСТЬ ====== */
#main {
    display: flex;
    flex: 1;
    overflow: hidden;
}

/* ====== ПАЛИТРА ====== */
#palette {
    width: 200px;
    background: #16213e;
    padding: 15px;
    border-right: 2px solid #0f3460;
    overflow-y: auto;
    z-index: 10;
    flex-shrink: 0;
}

#palette h3 {
    margin-bottom: 15px;
    color: #e94560;
    text-align: center;
    font-size: 14px;
}

.palette-section {
    margin-bottom: 15px;
}

.palette-section-title {
    font-size: 11px;
    color: #888;
    margin-bottom: 8px;
    padding-bottom: 3px;
    border-bottom: 1px solid #333;
}

.palette-item {
    background: #0f3460;
    padding: 8px;
    margin-bottom: 6px;
    border-radius: 8px;
    cursor: grab;
    text-align: center;
    transition: all 0.3s;
    border: 2px solid transparent;
    user-select: none;
}

.palette-item:hover {
    border-color: #e94560;
    transform: scale(1.02);
}

.palette-item:active {
    cursor: grabbing;
}

.palette-item svg {
    width: 50px;
    height: 32px;
    margin-bottom: 2px;
    pointer-events: none;
}

```

```
.palette-item-name {
    font-size: 10px;
    color: #aaa;
    pointer-events: none;
}

.type-legend {
    margin-top: 15px;
    padding-top: 10px;
    border-top: 1px solid #333;
    font-size: 10px;
}

.type-legend-item {
    display: flex;
    align-items: center;
    gap: 8px;
    margin-bottom: 5px;
}

.type-legend-dot {
    width: 12px;
    height: 12px;
    border-radius: 50%;
    border: 2px solid #fff;
}
.type-legend-dot.logic { background: #a855f7; }
.type-legend-dot.number { background: #3b82f6; }

/* ====== РАБОЧАЯ ОБЛАСТЬ ====== */
#workspace-container {
    flex: 1;
    position: relative;
    overflow: hidden;
    background-color: #0a0a1a;
    background-image:
        linear-gradient(rgba(255,255,255,0.04) 1px, transparent 1px),
        linear-gradient(90deg, rgba(255,255,255,0.04) 1px, transparent 1px);
    background-size: 25px 25px;
}

#workspace {
    position: absolute;
    transform-origin: 0 0;
    width: 5000px;
    height: 5000px;
}

#connections-svg {
    position: absolute;
    transform-origin: 0 0;
    pointer-events: none;
    z-index: 5;
    width: 5000px;
    height: 5000px;
}

#connections-svg path {
    pointer-events: stroke;
}

/* ====== ЭЛЕМЕНТЫ ====== */
.element {
    position: absolute;
    background: #0f3460;
```

```
6767      border: 2px solid #4a90d9;
6768      border-radius: 8px;
6769      cursor: move;
6770      user-select: none;
6771      z-index: 10;
6772      display: flex;
6773      flex-direction: column;
6774  }
6775
6776  .element.selected {
6777      border-color: #e94560;
6778      box-shadow: 0 0 15px rgba(233, 69, 96, 0.5);
6779  }
6780
6781  .element-header {
6782      background: #4a90d9;
6783      padding: 5px 10px;
6784      border-radius: 5px 5px 0 0;
6785      font-size: 11px;
6786      font-weight: bold;
6787      text-align: center;
6788      white-space: nowrap;
6789      overflow: hidden;
6790      text-overflow: ellipsis;
6791  }
6792
6793  .element-body {
6794      padding: 10px;
6795      display: flex;
6796      justify-content: space-between;
6797      align-items: center;
6798      flex: 1;
6799      gap: 8px;
6800  }
6801
6802  .element-symbol {
6803      font-size: 16px;
6804      font-weight: bold;
6805      flex: 1;
6806      text-align: center;
6807      padding: 0 5px;
6808      word-break: break-all;
6809      color: #eee;
6810  }
6811
6812  /* ====== ПОРТЫ ====== */
6813  .ports-left, .ports-right {
6814      display: flex;
6815      flex-direction: column;
6816      justify-content: space-around;
6817      gap: 10px;
6818      height: 100%;
6819  }
6820
6821  .port {
6822      width: 14px;
6823      height: 14px;
6824      border-radius: 50%;
6825      border: 2px solid #fff;
6826      cursor: crosshair;
6827      transition: all 0.2s;
6828      position: relative;
6829      flex-shrink: 0;
6830  }
6831
```

```
.port:hover { transform: scale(1.3); }
.port.input { margin-left: -8px; }
.port.output { margin-right: -8px; }
.port.connected { background: #4ade80; }

/* Типы портов */
.port.logic-port { background: #a855f7; border-color: #e9d5ff; }
.port.logic-port:hover { background: #c084fc; }
.port.logic-port.connected { background: #7c3aed; }

.port.number-port { background: #3b82f6; border-color: #bfdbfe; }
.port.number-port:hover { background: #60a5fa; }
.port.number-port.connected { background: #2563eb; }

.port.any-port { background: #6b7280; border-color: #d1d5db; }
.port.any-port:hover { background: #9ca3af; }
.port.any-port.connected { background: #4b5563; }

.port.output.yes-port { background: #4ade80 !important; border-color: #bbf7d0 !important; }
.port.output.no-port { background: #f87171 !important; border-color: #fecaca !important; }

.port.incompatible { opacity: 0.3; cursor: not-allowed; }
.port.compatible-highlight { box-shadow: 0 0 10px 3px #4ade80; }

/* ====== RESIZE HANDLES ====== */
.resize-handle {
    position: absolute;
    width: 12px;
    height: 12px;
    background: #e94560;
    border: 1px solid #fff;
    border-radius: 3px;
    z-index: 20;
    opacity: 0;
    transition: opacity 0.2s;
}
.element.selected .resize-handle { opacity: 0.8; }
.resize-handle:hover { opacity: 1; }
.resize-handle.handle-se { bottom: -6px; right: -6px; cursor: se-resize; }
.resize-handle.handle-e { top: 50%; right: -6px; transform: translateY(-50%); cursor: ew-resize; }
.resize-handle.handle-s { bottom: -6px; left: 50%; transform: translateX(-50%); cursor: ns-resize; }


/* ====== ВХОДНОЙ СИГНАЛ (ТРАПЕЦИЯ) ====== */
.element.input-signal {
    background: transparent;
    border: none;
}

.element.input-signal .element-header {
    display: none; /* У трапеции нет заголовка */
}

.element.input-signal .element-body {
    padding: 0;
    background: #0f3460;
    border: 2px solid #4a90d9;
    clip-path: polygon(0 0, 80% 0, 100% 50%, 80% 100%, 0 100%);
    display: flex;
    justify-content: space-between;
    align-items: center;
```

```
    padding-left: 15px;
    padding-right: 25px;
}

.element.input-signal .element-symbol {
    text-align: left;
    color: #eee;
}

.element.input-signal.selected .element-body {
    border-color: #e94560;
}

/* ====== ЭЛЕМЕНТ ВЫХОДА (ПУНКТИР) ====== */
.element.output {
    background: rgba(16, 185, 129, 0.1);
    border: 2px dashed #10b981;
}

.element.output .element-header {
    display: none; /* У выхода нет заголовка */
}

.element.output .element-body {
    padding-left: 20px;
}

.element.output .element-symbol {
    color: #10b981;
    font-size: 14px;
}

.element.output.selected {
    border-color: #e94560;
    border-style: dashed;
}


/* Formula condition port */
/* Универсальный стиль для технического порта (сверху) */
.element.has-condition-port {
    margin-top: 30px; /* Даем место порту над элементом */
}

.condition-port-wrapper {
    position: absolute;
    top: -28px;
    left: 50%;
    transform: translateX(-50%);
    display: flex;
    flex-direction: column;
    align-items: center;
    gap: 4px;
    pointer-events: none;
    z-index: 21;
}

.condition-port-label {
    font-size: 10px;
    color: #f59e0b;
    font-weight: 600;
    white-space: nowrap;
}

.port.condition-port {
```

```
    pointer-events: auto;
    width: 16px;
    height: 16px;
    border-radius: 50%;
    border: 2px solid #f59e0b;
    background: #fff7ed;
    margin: 0; /* Сбрасываем лишние отступы */
}
.element.formula .condition-port:hover { background: #fde68a; }


/* ====== СОЕДИНЕНИЯ ====== */
.connection {
    fill: none !important;  /* ← добавляем !important */
    stroke: #4a90d9;
    stroke-width: 2.5;
}
.connection:hover {
    stroke: #e94560;
    stroke-width: 4;
}

.connection.logic-conn { stroke: #a855f7; }
.connection.numeric-conn { stroke: #3b82f6; }
.connection.any-conn { stroke: #6b7280; }
.connection.true-conn { stroke: #4ade80; }
.connection.false-conn { stroke: #f87171; }

.connection.yes-conn { stroke: #4ade80; }
.connection.no-conn { stroke: #f87171; }

.temp-connection {
    fill: none !important;  /* ← добавляем !important */
    stroke: #e94560;
    stroke-width: 2;
    stroke-dasharray: 5, 5;
}
.temp-connection.invalid { stroke: #ef4444; }

/* ====== ПРОЧЕЕ ====== */
.drag-preview {
    position: fixed;
    pointer-events: none;
    opacity: 0.8;
    z-index: 1000;
    background: #0f3460;
    border: 2px solid #e94560;
    border-radius: 8px;
    padding: 10px 15px;
    color: #fff;
    font-size: 12px;
}

#minimap {
    position: absolute;
    bottom: 20px;
    right: 20px;
    width: 200px;
    height: 150px;
    background: #16213e;
    border: 2px solid #0f3460;
    border-radius: 8px;
    overflow: hidden;
    z-index: 50;
}
```

```
7023
7024  #minimap-canvas { width: 100%; height: 100%; }
7025  #minimap-viewport {
7026      position: absolute;
7027      border: 2px solid #e94560;
7028      background: rgba(233, 69, 96, 0.2);
7029      pointer-events: none;
7030  }
7031
7032  #viewport-info {
7033      position: absolute;
7034      bottom: 20px;
7035      left: 20px;
7036      background: rgba(22, 33, 62, 0.9);
7037      padding: 8px 12px;
7038      border-radius: 5px;
7039      font-size: 11px;
7040      color: #888;
7041      z-index: 50;
7042      display: flex;
7043      gap: 15px;
7044  }
7045  #selection-info { color: #e94560; }
7046
7047  #modal-overlay, .modal-overlay-class {
7048      display: none;
7049      position: fixed;
7050      top: 0; left: 0;
7051      width: 100%; height: 100%;
7052      background: rgba(0, 0, 0, 0.7);
7053      z-index: 1000;
7054      justify-content: center;
7055      align-items: center;
7056  }
7057
7058  #modal, .modal-class {
7059      background: #16213e;
7060      border-radius: 10px;
7061      padding: 20px;
7062      min-width: 400px;
7063      max-width: 600px;
7064      max-height: 80vh;
7065      overflow-y: auto;
7066      border: 2px solid #0f3460;
7067  }
7068
7069  #modal h3, .modal-class h3 { margin-bottom: 15px; color: #e94560; }
7070  .modal-row { margin-bottom: 15px; }
7071  .modal-row label { display: block; margin-bottom: 5px; color: #aaa; font-size: 13px; }
7072  .modal-row input, .modal-row select, .modal-row textarea {
7073      width: 100%;
7074      padding: 10px;
7075      background: #0f3460;
7076      border: 1px solid #4a90d9;
7077      border-radius: 5px;
7078      color: #eee;
7079      font-size: 14px;
7080  }
7081  .modal-row input:focus, .modal-row select:focus, .modal-row textarea:focus { outline:
      none; border-color: #e94560; }
7082  .modal-row textarea { min-height: 80px; font-family: inherit; resize: vertical; }
7083  .signal-list { max-height: 100px; overflow-y: auto; background: #0f3460; border-
      radius: 5px; padding: 5px; margin-top: 5px; }
7084  .signal-item { padding: 5px 10px; cursor: pointer; border-radius: 3px; font-size:
      12px; }
```

```
.signal-item:hover { background: #4a90d9; }
.modal-buttons { display: flex; gap: 10px; justify-content: flex-end; margin-top: 20px; }
.modal-btn { padding: 10px 25px; border: none; border-radius: 5px; cursor: pointer; font-size: 14px; transition: background 0.3s; }
.modal-btn.save { background: #4ade80; color: #000; }
.modal-btn.save:hover { background: #22c55e; }
.modal-btn.cancel { background: #6b7280; color: #fff; }
.modal-btn.cancel:hover { background: #4b5563; }

#context-menu {
    display: none;
    position: fixed;
    background: #16213e;
    border: 1px solid #0f3460;
    border-radius: 5px;
    padding: 5px 0;
    z-index: 1001;
    min-width: 150px;
    box-shadow: 0 5px 20px rgba(0,0,0,0.3);
}
.context-item { padding: 10px 15px; cursor: pointer; font-size: 13px; transition: background 0.2s; }
.context-item:hover { background: #0f3460; }

#file-input { display: none; }

.project-type-selector { display: flex; gap: 10px; margin-bottom: 15px; }
.project-type-btn { flex: 1; padding: 15px; background: #0f3460; border: 2px solid #4a90d9; border-radius: 8px; color: #eee; cursor: pointer; text-align: center; transition: all 0.3s; }
.project-type-btn:hover { border-color: #e94560; }
.project-type-btn.active { background: #4a90d9; border-color: #4a90d9; }
.project-type-btn .type-icon { font-size: 24px; margin-bottom: 5px; }
.project-type-btn .type-name { font-weight: bold; }
.project-type-btn .type-desc { font-size: 11px; color: #aaa; margin-top: 3px; }

.conditional-fields { display: none; padding: 15px; background: #0a0a1a; border-radius: 8px; margin-top: 10px; }
.conditional-fields.visible { display: block; }

::-webkit-scrollbar { width: 8px; height: 8px; }
::-webkit-scrollbar-track { background: #0a0a1a; }
::-webkit-scrollbar-thumb { background: #4a90d9; border-radius: 4px; }
::-webkit-scrollbar-thumb:hover { background: #e94560; }

/* Стили для выходов */
.output-btn { position: relative; }
.output-counter { display: inline-block; background: #e94560; color: white; font-size: 11px; font-weight: bold; padding: 2px 6px; border-radius: 10px; margin-left: 5px; min-width: 18px; text-align: center; }
.output-counter:empty, .output-counter[style*="display: none"] { display: none; }
.element.has-output { box-shadow: 0 0 10px rgba(16, 185, 129, 0.3); }
.element.output-highlighted { box-shadow: 0 0 20px rgba(251, 191, 36, 0.6) !important; border-color: #fbbf24 !important; }
.port.output-active { box-shadow: 0 0 8px 2px rgba(16, 185, 129, 0.8); animation: pulse-output 1.5s infinite; }
@keyframes pulse-output {
    0%, 100% { box-shadow: 0 0 8px 2px rgba(16, 185, 129, 0.8); }
    50% { box-shadow: 0 0 12px 4px rgba(16, 185, 129, 1); }
}

.outputs-container { background: #0a0a1a; border-radius: 8px; padding: 15px; max-height: 250px; overflow-y: auto; }
.outputs-section { margin-bottom: 15px; }
```

```
.outputs-section:last-child { margin-bottom: 0; }
.outputs-section-title { color: #10b981; font-weight: bold; font-size: 13px; margin-bottom: 10px; padding-bottom: 5px; border-bottom: 1px solid #333; display: flex; align-items: center; gap: 8px; }
.outputs-section-title .section-icon { font-size: 16px; }
.outputs-list { display: flex; flex-direction: column; gap: 5px; }
.output-item { display: flex; align-items: center; gap: 10px; padding: 8px 12px; background: rgba(16, 185, 129, 0.1); border: 1px solid rgba(16, 185, 129, 0.3); border-radius: 5px; cursor: pointer; transition: all 0.2s; }
.output-item:hover { background: rgba(16, 185, 129, 0.2); border-color: #10b981; transform: translateX(5px); }
.output-item.numeric { background: rgba(59, 130, 246, 0.1); border-color: rgba(59, 130, 246, 0.3); }
.output-item.numeric:hover { background: rgba(59, 130, 246, 0.2); border-color: #3b82f6; }
.output-icon { font-size: 14px; }
.output-name { font-weight: bold; color: #eee; }
.output-port { color: #888; font-size: 12px; margin-left: auto; }
.no-outputs { color: #666; font-style: italic; padding: 10px; text-align: center; }
.outputs-hint { margin-top: 10px; padding: 10px; background: rgba(59, 130, 246, 0.1); border-radius: 5px; font-size: 12px; color: #888; line-height: 1.4; }
.element.output-ambiguous { box-shadow: 0 0 18px 4px rgba(240, 80, 80, 0.55); border-color: rgba(240,80,80,0.8) !important; }
.element.output-missing { box-shadow: 0 0 14px 3px rgba(250, 200, 30, 0.5); border-color: rgba(250,200,30,0.8) !important; }
/* TRUE/FALSE порты (для сепаратора) */
.port.true-port {
    background: #4ade80 !important;
    border-color: #bbf7d0 !important;
}
.port.true-port:hover {
    background: #22c55e !important;
}
.port.true-port.connected {
    background: #16a34a !important;
}

.port.false-port {
    background: #f87171 !important;
    border-color: #fecaca !important;
}
.port.false-port:hover {
    background: #ef4444 !important;
}
.port.false-port.connected {
    background: #dc2626 !important;
}

/* Сепаратор стиль */
.element.separator {
    background: #0f3460;
    border: 2px solid #f59e0b;
}

.element.separator.selected {
    border-color: #e94560;
    box-shadow: 0 0 15px rgba(233, 69, 96, 0.5);
}

/* === Выделение рамкой === */
#selection-rect {
  position: absolute;
  border: 1px dashed #e94560;
  background: rgba(233, 69, 96, 0.1);
  pointer-events: none;
```

```
  display: none;
  z-index: 200;
}

/* === Кастомный элемент “Группа” === */
.element.group {
  background: rgba(107, 114, 128, 0.12);
  border: 2px dashed #6b7280;
  border-radius: 8px;
  position: absolute;
  z-index: 1;              /* ниже обычных элементов (у них z-index: 10) */
}

.element.group .group-title {
  pointer-events: auto;
}

.group-title {
  position: absolute;
  top: -20px;
  left: 5px;
  font-size: 11px;
  color: #ccc;
  background: #16213e;
  padding: 2px 6px;
  border-radius: 4px;
  pointer-events: auto; /* можно кликнуть для выбора */
}

.modal.hidden { display: none; }
.modal { position: fixed; inset: 0; display: flex; align-items: center; justify-content: center; background: rgba(0,0,0,0.4); z-index: 1000; }
.modal__content { background: #fff; padding: 24px; border-radius: 8px; width: 640px; max-height: 80vh; display: flex; flex-direction: column; gap: 16px; overflow: hidden; }
.modal__content--wide { width: 800px; }
.modal__title { margin: 0; }

.project-list__toolbar { display: flex; gap: 12px; }
.project-list__toolbar input { flex: 1; padding: 6px 10px; }
.project-list__table-container { flex: 1; overflow: auto; border: 1px solid #ddd; border-radius: 6px; }
.project-list__table { width: 100%; border-collapse: collapse; }
.project-list__table th, .project-list__table td { padding: 8px 12px; border-bottom: 1px solid #eee; }
.project-list__table tbody tr { cursor: pointer; transition: background 0.15s ease; }
.project-list__table tbody tr:hover { background: #f0f6ff; }
.project-list__empty { text-align: center; color: #888; padding: 16px; }
.modal__actions { display: flex; justify-content: flex-end; gap: 12px; }
.project-list__table th,
.project-list__table td {
  color: #111;          /* насыщённый чёрный текст */
  padding: 8px 12px;
  border-bottom: 1px solid #eee;
}
.modal__content--wide {
  width: 860px;
  max-height: 90vh;    /* занимает 90% экрана */
}

.project-list__table-container {
  flex: 1;
  overflow: auto;
  border: 1px solid #ddd;
  border-radius: 6px;
```

```
    max-height: 60vh;     /* много строк */
  }

  .element-comment {
    padding: 6px 10px 10px;
    font-size: 11px;
    color: #cbd5e1;
    opacity: 0.9;
    border-top: 1px solid rgba(255,255,255,0.08);
    white-space: pre-wrap;
    word-break: break-word;
  }

  .element-comment:empty { display: none; }

```