Точная инструкция по настройке Spec-Kit и MemoryBank для Cursor

Структура файлов в проекте

Создайте следующую структуру каталогов в корне вашего проекта:

```
ваш-проект/
├── .cursor/
│   └── rules/
│       ├── memory-bank.mdc
│       ├── project-constitution.mdc
│       └── experiment-protocol.mdc
├── memory-bank/
│   ├── 00-project-overview.md
│   ├── 01-architecture.md
│   ├── 02-current-experiments.md
│   ├── 03-best-results.md
│   ├── 04-failed-attempts.md
│   └── 05-progress-log.md
├── experiments/
│   └── [автоматически создаваемые папки]
└── .speckit/
    ├── constitution.md
    └── templates/
```

1. Настройка MemoryBank

Файл: `.cursor/rules/memory-bank.mdc`

```
---
description: "Memory Bank система для сохранения контекста между сессиями"
globs:
alwaysApply: true
priority: 100
---

# Cursor Memory Bank System

## КРИТИЧЕСКИЕ ПРАВИЛА - НАРУШЕНИЕ НЕДОПУСТИМО

1. **НИКОГДА НЕ УДАЛЯЙ И НЕ ПЕРЕЗАПИСЫВАЙ файлы в `/experiments/` и `/memory-bank/`**
2. **ВСЕГДА ЧИТАЙ ВСЕ файлы memory-bank ПЕРЕД началом работы**
3. **ОБЯЗАТЕЛЬНО обновляй memory-bank после каждого значимого результата**
```

```
4. **ЗАПРЕЩЕНО работать без создания нового experiment-ID**

## Обязательная последовательность действий

При каждом запуске:
1. Прочитай ВСЕ файлы из `memory-bank/`
2. Создай новый experiment ID: `exp-YYYYMMDD-HHMM-<краткое_описание>`
3. Работай ТОЛЬКО в новой папке `/experiments/<experiment-ID>/`
4. Логируй ВСЕ действия в `05-progress-log.md`
5. При получении результата - обнови `03-best-results.md` ТОЛЬКО если результат лучше

## Memory Bank файлы

- `00-project-overview.md` - Общее описание проекта, цели, архитектура
- `01-architecture.md` - Техническая архитектура, стек технологий, ключевые решения
- `02-current-experiments.md` - Текущие эксперименты, гипотезы, планы
- `03-best-results.md` - Лучшие достигнутые результаты с путями к файлам и метриками
- `04-failed-attempts.md` - Неудачные попытки и извлеченные уроки
- `05-progress-log.md` - Хронологический лог всех действий

## Правила обновления лучших результатов

Обновляй `03-best-results.md` ТОЛЬКО если:
- Новая метрика СТРОГО лучше предыдущей
- Модель демонстрирует новые возможности
- Найден значительно более эффективный подход

## Формат записи в progress-log
```

[YYYY-MM-DD HH:MM] Experiment: <ID>

Цель: <что делали>
Результат: <что получили>
Метрики: <числовые показатели>
Файлы: <пути к сохраненным артефактам>
Статус: УСПЕХ/НЕУДАЧА/В_ПРОЦЕССЕ
Следующие шаги: <что планируется>

```
```

Файлы Memory Bank (создайте каждый):

`memory-bank/00-project-overview.md`

```
# Обзор проекта

## Цель проекта
[Опишите основную цель вашего проекта]

## Технологии
- CUDA/cuFFT
```

```
- Python/C++
- [другие технологии]

## Ключевые задачи
1. [Основная задача 1]
2. [Основная задача 2]

## Критерии успеха
- [Метрика 1]: [целевое значение]
- [Метрика 2]: [целевое значение]

## Текущий статус
**Дата обновления:** [дата]
**Общий прогресс:** [%]
**Ключевые достижения:** [список]
```

`memory-bank/01-architecture.md`

```
# Архитектура проекта

## Структура кода
```

src/
├── cuda/ # CUDA kernels
├── python/ # Python wrappers
├── experiments/ # Экспериментальный код
└── utils/ # Утилиты

```

## Ключевые компоненты
1. **FFT модуль** - [описание]
2. **Optimization модуль** - [описание]

## Технические решения
- [Решение 1]: [обоснование]
- [Решение 2]: [обоснование]

## Зависимости
- CUDA Toolkit: [версия]
- cuFFT: [версия]
- [другие зависимости]
```

`memory-bank/03-best-results.md`

```
# Лучшие результаты

## Актуальная лучшая модель

**Дата:** [дата]
**Experiment ID:** [ID]
**Путь к файлам:** `/experiments/[ID]/artifacts/`
```

```
### Метрики
- **Основная метрика:** [значение]
- **Скорость:** [значение]
- **Точность:** [значение]

### Конфигурация
```

[конфигурация лучшей модели]

```
### Воспроизведение
```

[команды для воспроизведения результата]

```
## История лучших результатов
[Таблица с предыдущими лучшими результатами]
```

2. Настройка Spec-Kit

Установка Spec-Kit

```
# Установите uv если еще нет
curl -LsSf https://astral.sh/uv/install.sh | sh

# Установите Spec-Kit в проект
uvx specify init .
```

Файл: `.cursor/rules/project-constitution.mdc`

```
---
description: "Конституция проекта - основные принципы разработки"
globs:
alwaysApply: true
priority: 90
---

# Конституция проекта

## Принципы разработки

### 1. Воспроизводимость
- Все эксперименты должны быть полностью воспроизводимыми
- Конфигурации версионируются
- Результаты сохраняются с полными метаданными

### 2. Инкрементальность
- Никогда не удалять предыдущие результаты
```

```
- Только добавлять новые эксперименты
- Сравнивать с предыдущими результатами

### 3. Документирование
- Каждое изменение логируется
- Каждый эксперимент документируется
- Неудачи документируются наравне с успехами

## Workflow правила

### Перед началом работы
1. `/speckit.specify` - определить что строим
2. `/speckit.plan` - создать техплан
3. `/speckit.tasks` - разбить на задачи
4. Создать experiment ID
5. Прочитать memory bank

### В процессе работы
- Логировать каждый шаг
- Сохранять промежуточные результаты
- Обновлять progress log

### После завершения
- Обновить memory bank
- Сравнить с лучшими результатами
- Задокументировать выводы

## Технические стандарты
- Код на Python: PEP 8
- CUDA код: комментарии на русском/английском
- Конфиги: YAML формат
- Логи: структурированный формат

## Запрещенные действия
- Удаление experiment папок
- Перезапись best results без улучшения метрик
- Работа без experiment ID
- Пропуск документирования
```

Файл: `.cursor/rules/experiment-protocol.mdc`

```
---
description: "Протокол проведения экспериментов"
globs:
alwaysApply: true
priority: 80
---

# Протокол экспериментов

## Обязательная последовательность

### 1. Планирование (/speckit.specify)
```

Что мы хотим улучшить?
Какие метрики важны?
Какие ограничения есть?
Критерии успеха?

```
### 2. Техплан (/speckit.plan)
```

Архитектура решения
Выбор алгоритмов
План тестирования
Ожидаемые результаты

```
### 3. Задачи (/speckit.tasks)
```

Пошаговый план
Критерии готовности
Оценка времени
Зависимости между задачами

```
### 4. Реализация (/speckit.implement)
```

Experiment ID: exp-YYYYMMDD-HHMM-<описание>
Папка: /experiments/<experiment-ID>/
Обязательные файлы:

- config.yaml
- run.py/main.py
- results/
- logs/
- README.md

```
## Структура experiment папки
```

/experiments/exp-20241008-1200-fft-optimization/
├── config.yaml # Полная конфигурация
├── src/ # Исходный код эксперимента
├── results/ # Результаты и метрики
│ ├── metrics.json
│ ├── best_model.pt
│ └── plots/

├── logs/ # Логи выполнения
├── artifacts/ # Дополнительные артефакты
└── README.md # Описание эксперимента

```
## Критерии обновления лучших результатов

Обновлять `/memory-bank/03-best-results.md` только если:

1. **Основная метрика улучшилась** (например, accuracy > предыдущий best)
2. **При равной метрике - скорость выше**
3. **Значительное архитектурное улучшение**

## Шаблон README.md для эксперимента
```

Experiment: [ID]

Цель

[Что хотели достичь]

Гипотеза

[Предположение которое проверяли]

Метод

[Как проверяли]

Результаты

- Основная метрика: [значение]
- Время выполнения: [значение]
- Память: [значение]

Выводы

[Что узнали]

Следующие шаги

[Что планируется дальше]

```
## Automation hooks

Всегда вызывать перед работой:
```

def setup_experiment():
# Читаем memory bank
# Создаем experiment ID
# Создаем папки
# Инициализируем логирование

```
Всегда вызывать после работы:
```

def finalize_experiment():
# Сохраняем результаты
# Обновляем memory bank
# Сравниваем с best results
# Логируем в progress log

```

```

3. Spec-Kit команды

После установки используйте эти команды в Cursor:

1. `/speckit.constitution` - создание/обновление принципов проекта
2. `/speckit.specify` - определение что вы хотите построить
3. `/speckit.plan` - создание технического плана
4. `/speckit.tasks` - разбивка на задачи
5. `/speckit.implement` - реализация по плану

4. Инструкции для ИИ в чате

Добавьте эти правила в начале каждого чата с Cursor:

```
ОБЯЗАТЕЛЬНЫЕ ПРАВИЛА:

1. ВСЕГДА сначала прочитай ВСЕ файлы из memory-bank/
2. НИКОГДА не удаляй и не перезаписывай файлы в /experiments/
3. Для КАЖДОГО эксперимента создавай новый ID формата: exp-YYYYMMDD-HHMM-описание
4. ОБЯЗАТЕЛЬНО логируй все действия в 05-progress-log.md
5. Обновляй 03-best-results.md ТОЛЬКО если новый результат СТРОГО лучше

Начни с команды: "Читаю memory bank и готовлюсь к работе..."
```

5. Автоматизационные скрипты

Создайте файл `scripts/setup_experiment.py`:

```
#!/usr/bin/env python3
import os
import json
import yaml
from datetime import datetime

def create_experiment(description):
    timestamp = datetime.now().strftime("%Y%m%d-%H%M")
    exp_id = f"exp-{timestamp}-{description}"

    exp_dir = f"experiments/{exp_id}"
    os.makedirs(exp_dir, exist_ok=True)
    os.makedirs(f"{exp_dir}/src", exist_ok=True)
    os.makedirs(f"{exp_dir}/results", exist_ok=True)
    os.makedirs(f"{exp_dir}/logs", exist_ok=True)
    os.makedirs(f"{exp_dir}/artifacts", exist_ok=True)

    # Создаем базовые файлы
    config = {
        "experiment_id": exp_id,
        "created": datetime.now().isoformat(),
        "description": description,
        "metrics": {},
        "status": "in_progress"
    }

    with open(f"{exp_dir}/config.yaml", "w") as f:
        yaml.dump(config, f, default_flow_style=False)

    with open(f"{exp_dir}/README.md", "w") as f:
        f.write(f"""# Experiment: {exp_id}

## Цель
{description}

## Статус
В процессе

## Результаты
TBD
""")

    print(f"Создан эксперимент: {exp_id}")
    return exp_id

if __name__ == "__main__":
    import sys
    if len(sys.argv) > 1:
        create_experiment(sys.argv[^1])
    else:
        print("Использование: python setup_experiment.py 'описание эксперимента'")
```

6. Проверка настройки

После настройки проверьте:

1. Все папки созданы
2. Файлы .mdc в .cursor/rules/ присутствуют
3. Memory bank файлы заполнены базовой информацией
4. Spec-Kit команды работают в Cursor

Начните тестирование с простой команды:

```
/speckit.specify "Хочу оптимизировать FFT для работы с большими векторами"
```

Cursor должен прочитать ваши правила и следовать структурированному подходу к разработке, сохраняя все результаты и никогда не теряя лучшие модели.

⁂

1. https://apidog.com/blog/cursor-setup-guide/
2. https://www.reddit.com/r/cursor/comments/1jkfp7m/memory_bank_for_cursor/
3. https://cursor.com/learn/context
4. https://dev.to/stamigos/setting-up-cursor-rules-the-complete-guide-to-ai-enhanced-development-24cg
5. https://gist.github.com/ipenywis/1bdb541c3a612dbac4a14e1e3f4341ab
6. https://www.arsturn.com/blog/efficient-memory-management-cursor-context-handling
7. https://github.com/github/spec-kit
8. https://www.youtube.com/watch?v=Uufa6flWid4
9. https://playbooks.com/rules/memory
10. https://cursor.com/docs/configuration/tools/web-development
11. https://forum.cursor.com/t/memory-bank-feature-for-your-cursor/71979
12. https://github.com/vanzan01/cursor-memory-bank
13. https://www.youtube.com/watch?v=azXNHRtzd5s
14. https://github.com/madebyaris/spec-kit-command-cursor
15. https://apidog.com/blog/cline-memory-cursor/
16. https://www.reddit.com/r/cursor/comments/1jdef7p/i_built_a_system_that_gives_cursor_persistent/
17. https://www.hubermann.com/en/blog/mastering-cursor-configuration-a-comprehensive-guide-to-project-rules-and-settings
18. https://www.lullabot.com/articles/supercharge-your-ai-coding-cursor-rules-and-memory-banks
19. https://forum.cursor.com/t/agentic-memory-management-for-cursor/78021