

## Программирование на C++ в Unreal Engine 5

Данный курс научит Вас созданию игр на C++ в Unreal Engine 5. Курс состоит из 12 разделов, в которых Вас ждёт теория и практика. Причём, в качестве практики будет создан весьма крупный проект объёмом **свыше 5000 строк качественного кода**, который уже на практике познакомит Вас с принципами создания игр на C++ в Unreal Engine 5.

Параллельно с курсом Вы также будете получать домашние задания, результатом которых станет, в том числе, полноценная серьёзная работа для портфолио. Помимо самого курса Вас ждёт ещё и очень ценный Бонус: «**Тестирование Unreal-проектов на Python**», в рамках которого Вы научитесь писать очень полезные тесты для тестирования самых разных аспектов разработки игр.

[Подробнее](#)

### Уроки и статьи

[Главная](#)[HTML](#)[CSS](#)[JavaScript](#)[PHP](#)[MySQL](#)[XML](#)[Joomla](#)[Регистрация сайта](#)[Раскрутка сайта](#)[Java](#)[Python](#)[Python Основы](#)

Вы здесь: [Главная](#) - [Python](#) - [Основы Python](#) -  
Кэширование функций в Python

## Кэширование функций в Python



В сегодняшней статье мы рассмотрим как оптимизировать время выполнения программы в **Python** с помощью операции **кэширования**. Мы рассмотрим данную операцию на примере рекурсивной функции в **Python**. Но сначала - пара слов о рекурсии. **Рекурсия** — это понятие в программировании, при котором функция вызывает сама себя один или несколько раз. Данные типы функций часто сталкиваются с проблемами скорости, из-за того, что функция постоянно вызывает сама себя. Операция рекурсии занимает достаточно много памяти из-за постоянного повторения одних и тех же шагов. **Кэширование** или **Мемоизация** (англ. **memoization** от англ. memory и англ. optimization) помогает этому процессу, сохраняя значения, которые

C#

C++

Android

Учебники

Видеоуроки

Софт

Форум

Карта сайта

**Видеокурсы**

Видеоуроки (VIP)

## Подписка

Подпишитесь на мой канал на YouTube, где я регулярно публикую новые видео.



Подписаться

Подписавшись по **E-mail**, Вы будете получать уведомления о новых статьях.



Подписаться

Добавляйтесь ко мне в друзья **ВКонтакте!** Отзывы о сайте и обо мне оставляйте в моей группе.



Мой аккаунт

Моя группа

## Опрос

Какая тема Вас интересует больше?

Создание сайтов

Создание игр

Создание приложений

уже были рассчитаны для последующего использования. Давайте сначала вспомним что такое рекурсивные функции. Давайте посмотрим несколько примеров!

### Написание факториальной функции.

Факториалы — один из самых простых примеров рекурсии, и является результатом умножения всех чисел меньших на единицу чем данное:

```
def factorial(n):  
    # установите базовый случай!  
    if n <= 1:  
        return 1  
    else:  
        return factorial( n - 1 ) * n  
  
print( factorial(5) )    # результат от умножения 5 * 4 * 3 * 2 * 1  
  
# вывод  
120
```

### Последовательность Фибоначчи.

Последовательность Фибоначчи — одна из самых известных формул в математике. Это также одна из самых известных рекурсивных функций в программировании. Каждое число в последовательность — это сумма двух предыдущих чисел, таких, что **fib(5) = fib(4) + fib(3)**. Вычислим ее для 20.

```
from datetime import datetime  
import time  
  
def fib(n):  
    if n <= 1:  
        return n  
    else:  
        return fib( n - 1 ) + fib( n - 2 )  
  
start_time = datetime.now()  
  
print(fib(35))  
  
print(datetime.now() - start_time)  
  
# вывод  
9227465  
0:00:17.647056 # время вычисления
```

Как видно на вычисление результата для числа 35, ушло целых 17 секунд.

По мере того, как растет количество передаваемых данных, растет структура и количество рекурсивных вызовов. Он экспоненциальный, что может значительно замедлить работу программы. Даже попытка выполнить fib(40) может занять пару минут, а fib(100) обычно не работает из-за проблем с максимальной глубиной

### Понимание мемоизации.

Когда вы заходите на веб-страницу в первый раз, вашему браузеру требуется некоторое время, чтобы загрузить изображения и файлы, необходимые странице. Когда вы во второй раз зайдете на ту же самую страницу, она обычно загружается намного быстрее. Это связано с тем, что ваш браузер использует технику, известную как «кэширование». В вычислительной технике мемоизация — это метод оптимизации, используемый в первую очередь для ускорения программы, сохраняя результаты ранее вызванных функций и возвращая сохраненный результат при попытке вычислить ту же последовательность. Это просто известно как **кэширование**.

### Использование мемоизации.

Чтобы применить ее к последовательности Фибоначчи, мы должны понять, какой наилучший метод кэширования значений. В **Python** словари дают нам возможность для хранения значений на основе заданного ключа. Благодаря скорости и уникальной ключевой структуре словарей мы можем использовать их для хранения значения каждой последовательности Фибоначчи. Таким образом, как только одна последовательность, такая как **fib(3)**, рассчитывается, его не нужно вычислять снова. Он просто сохраняется в **кэше** и извлекаются по мере необходимости. Давайте попробуем:

```
cache = { }
def fib(n):
    if n in cache:
        return cache[ n ]
    result = 0

    if n <= 1:
        result = n
    else:
        result = fib( n - 1 ) + fib( n -2 )
        cache[ n ] = result
    return result

print( fib(50) )
```

### Использование @lru\_cache

Теперь, когда мы знаем, как самостоятельно создать систему **кэширования**, давайте воспользуемся встроенными средствами **Python**. способ запоминания. Он известен как **lru\_cache**.

```
from functools import lru_cache

@lru_cache( )    # встроенный инструмент кэширования
def fib(n):
    if n <= 1:
        return n
    else:
        return fib( n - 1 ) + fib( n - 2 )
```

Мы получим тот же результат, что и в предыдущем примере, но на этот раз с меньшим количеством строк. Таким образом, язык **Python** предоставляет возможность оптимизации кода библиотекой **functools**.

🕒 Создано 26.04.2022 09:31:08 • 👤 Михаил Русаков

[Предыдущая статья](#)[Следующая статья](#)

*Копирование материалов разрешается только с указанием автора (Михаил Русаков) и индексируемой прямой ссылкой на сайт (<http://myrusakov.ru>)!*

Добавляйтесь ко мне в друзья **ВКонтакте**:  
<http://vk.com/myrusakov>.

Если Вы хотите дать оценку мне и моей работе, то напишите её в моей группе:  
<http://vk.com/rusakovmy>.

Если Вы не хотите пропустить новые материалы на сайте,  
то Вы можете **подписаться на обновления**:  
[Подписаться на обновления](#)

Если у Вас остались какие-либо вопросы, либо у Вас есть желание высказаться по поводу этой статьи, то Вы можете оставить свой комментарий внизу страницы.

**Порекомендуйте эту статью друзьям:**

Если Вам понравился сайт, то разместите ссылку на него (у себя на сайте, на форуме, в контакте):

1. Кнопка:

```
<a href="https://myrusakov.ru" target="_blank"></a>
```

Она выглядит вот так:



2. Текстовая ссылка:

```
<a href="https://myrusakov.ru" target="_blank">Как  
создать свой сайт</a>
```

Она выглядит вот так: [Как создать свой сайт](#)

3. BB-код ссылки для форумов (например, можете поставить её в подписи):



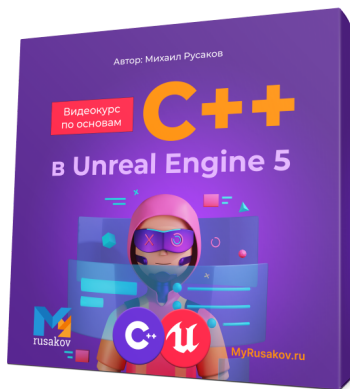
## Комментарии (0):

Добавить комментарий

Для добавления комментариев надо войти в систему.  
Если Вы ещё не зарегистрированы на сайте, то сначала [зарегистрируйтесь](#).

### Бесплатный курс

#### Основы C++ в Unreal Engine 5



#### Особенности курса:

- 5 часов видео
- 53 задания
- Поддержка от автора
- Все исходники приложены

**Чтобы получить  
Видеокурс,  
заполните форму**

Е-mail:

Имя:

Получить курс

Другие курсы

### Бесплатный онлайн-семинар

11 шагов к созданию  
своей Web-студии



### После семинара:

- Вы узнаете главное отличие богатых от бедных.
- Вы увидите разоблачения множества мифов об успешности и о бизнесе.
- Вы получите свой личный финансовый план прямо на семинаре.
- Мы разберём 11 шагов к созданию своей успешной Web-студии.
- Я расскажу о своих личных историях: об успешных и неуспешных бизнесах. Это мой многолетний опыт, которым я поделюсь с Вами.

[Записаться](#)

[Другие курсы](#)

### Умные цитаты

Тот, который передвигает горы,  
сначала убирает маленькие  
камешки.

*Конфуций*

