Сортиране на единично свързан списък чрез алгоритмите Quick sort и Merge sort

еодор Мангъров

4.04.2025г.

Генериран от Doxygen 1.13.2

1 Класове Указател 1
1.1 Класове Списък . . . 1

2 Файлове Списък 3
2.1 Файлове Списък . . . 3

3 Класове Документация 5
3.1 LinkedList Клас Препратка . . . 5
3.1.1 Подробно описание . . . 6
3.1.2 Конструктор & Деструктор Документация . . . 6
3.1.2.1 LinkedList() . . . 6
3.1.2.2 ~LinkedList() . . . 6
3.1.3 Членове Функции(методи) Документация . . . 7
3.1.3.1 clear() . . . 7
3.1.3.2 copyFrom() . . . 7
3.1.3.3 fillRandom() . . . 7
3.1.3.4 frontBackSplit() . . . 8
3.1.3.5 merge() . . . 8
3.1.3.6 mergeSort() . . . 9
3.1.3.7 print() . . . 9
3.1.3.8 push() . . . 9
3.1.3.9 quickSort() . . . 9
3.1.3.10 quickSortRecur() . . . 10
3.1.3.11 sortWithMergeSort() . . . 11
3.1.3.12 sortWithQuickSort() . . . 11
3.1.4 Член данни Документация . . . 11
3.1.4.1 head . . . 11
3.2 Node Структура Препратка . . . 11
3.2.1 Подробно описание . . . 11
3.2.2 Конструктор & Деструктор Документация . . . 11
3.2.2.1 Node() . . . 11
3.2.3 Член данни Документация . . . 12
3.2.3.1 data . . . 12
3.2.3.2 next . . . 12

4 Файлове Документация 13
4.1 LinkedList.cpp Файл Справка . . . 13
4.2 LinkedList.h Файл Справка . . . 13
4.3 LinkedList.h . . . 13
4.4 main.cpp Файл Справка . . . 14
4.4.1 Функции Документация . . . 14
4.4.1.1 main() . . . 14
4.4.2 Променливи Документация . . . 14
4.4.2.1 size . . . 14

Азбучен указател 15

Глава 1

Класове Указател

1.1 Класове Списък

Класове, структури, обединения и интерфейси с кратко описание:

LinkedList 5
Node 11

Глава 2

Файлове Списък

2.1 Файлове Списък

Пълен списък с файлове с кратко описание:

LinkedList.cpp 13
LinkedList.h 13
main.cpp 14

Глава 3

Класове Документация

3.1 LinkedList Клас Препратка

#include <LinkedList.h>

Общодостъпни членове функции

- void clear ()

  Деалокира динамичната памет на целия списък. Функцията е тясно свързана с деструктора на класа.
- void copyFrom (const LinkedList &other)

  Копиращ конструктор. Нуждаем се от него, защото ще сравняваме времето за сортиране на един и същ масив от случайни числа. Улеснява копирането на дефинирания тип, само чрез подаване на указателя сочещ началото му.
- void fillRandom (int count, int lower, int upper)

  Функция, която запълва списъка със случайни стойности. В дефиницията на функцията се създават числа в интервала от lower до upper.
- LinkedList ()

  Създава обект без параметри. Заделя динамична памет.
- void print () const

  Функция за принтиране на списъка. Имаме добавен const, за да запазим цялостта на данните в списъка.
- void push (int data)

  Функция за вкарване на елемент.
- void sortWithMergeSort ()

  Функция за сортиране на обект от тип единично свързан списък чрез алгоритъма Mergesort.
- void sortWithQuickSort ()

  Функция за сортиране на обект от тип единично свързан списък чрез алгоритъма Quicksort.
- ~LinkedList ()

  Разрушава обект от типа.

Частни членове функции

- void frontBackSplit (Node *head, Node **frontRef, Node **backRef)

  Разделя списъка, спомагайки процедурата по сортиране.
- Node * merge (Node *a, Node *b)

  Обединява два списъка. (Довършителната стъпка от алгоритъма Merge sort)
- Node * mergeSort (Node *head)

  Функция сортираща единично свързан списък чрез алгоритъма Mergesort.
- Node * quickSort (Node *head)

  Функцията обръщаща се към quickSortRecur, за реализиране на алгоритъма Quicksort.
- Node * quickSortRecur (Node *head, Node **newHead, Node **newEnd)

  Функция сортираща единично свързан списък чрез алгоритъма Quicksort.

Частни атрибути

- Node * head

  Указателя, който сочи към началния елемент в списъка.

3.1.1 Подробно описание

!

LinkedList е структурен клас на единично свързан списък с включени функции за сортиране чрез алгоритмите Merge sort и Quick sort.

3.1.2 Конструктор & Деструктор Документация

3.1.2.1 LinkedList()

LinkedList::LinkedList ()

Дефиниция конструктора за LinkedList.

!

Аргументи

head	Стойност по подразбиране - нулевия указател 0x0.

3.1.2.2 ~LinkedList()

LinkedList::~LinkedList ()

Дестуктора на типа.

!

Извиква clear()

3.1.3 Членове Функции(методи) Документация

3.1.3.1 clear()

void LinkedList::clear ()

Изчиства данните от списъка. Деалокира заделената динамична памет.

!

Извиква се в деструктора на типа.

3.1.3.2 copyFrom()

void LinkedList::copyFrom (
const LinkedList & other)

Дефиниция копиращ конструктор за LinkedList.

!

Аргументи

other	Обект от тип LinkedList взет по референция (без копиране) с ключовата дума const, подсигуряваща го от евентуални промени.
other	Константен указател от тип LinkedList по референция (чистия указател, без да се създават скрити копия)

1. Изчиства данните от конкретния (this) списък.
2. Запазва началната точка на other.
3. Обхожда списъка, започвайки от отправната точка other.
4. Презаписва с данните копирани от подадения списък.

3.1.3.3 fillRandom()

void LinkedList::fillRandom (
int count,
int lower,
int upper)

Запълва писъка със случайни числа.

!

Аргументи

count	Желаната бройка от елементи за запълване.
lower	Долна граница на интервала от числа за запълване.
upper	Горна граница на интервала от числа за запълване.
count	Броя на тези числа (определя размера на списъка)
lower	Долна граница (от)
lower	Горна граница (до)

Използва rand().

3.1.3.4 frontBackSplit()

```
void LinkedList::frontBackSplit (
                Node * head,
                Node ** frontRef,
                Node ** backRef)   [private]
```

a.k.a "Split the front and back!!!" [Разделяй] и владей! Юлий Цезар

!

Аргументи

head	Корена на списъка.
frontRef	Началото на 1-вия дял.
backRef	Началото на 2-рип дял.
head	Указател сочещ към корена на списъка.
frontRef	Указател сочещ началото на първия списък.
backRef	Указател сочещ началото на втория списък.

Връща

Нищо. void е. Презаписва frontRef и backRef.

3.1.3.5 merge()

```
Node * LinkedList::merge (
                Node * a,
                Node * b)   [private]
```

Процедурата по обединение.

!

Аргументи

a	1-ви списък
b	2-ри списък

Връща

Началото на обединения списък. Това е и крайната инструкция на mergeSort(Node*);

Аргументи

a	Първия половин списък.
b	Втория половин списък.

Връща

Указател към началото на обединените два списъка в един цял.

3.1.3.6 mergeSort()

Node * LinkedList::mergeSort (
Node * head) [private]

Дефиниция на алгоритъма Merge sort.

!

Аргументи

head	Корена на списъка. (началото)

Връща

Указател от тип Node, тоест един елемент на списъка. (корена)

1. Разделяме списъка на две части.
2. Процедурата се изпълнява рекурсивно за всяка една част до пълно потъване.
3. След достигане на два сортирани списъка се прилага процедура на обединение върху тях, а резултата от самата процедура е началото на вече наредения единично свързан списък!

3.1.3.7 print()

void LinkedList::print () const

Принтира списък.

!

3.1.3.8 push()

void LinkedList::push (
int data)

Добавя елемент в списъка

!

Аргументи

data	Стойността, която елементът ще съдържа.
data	Стойността на елемента.

3.1.3.9 quickSort()

Node * LinkedList::quickSort (
Node * head) [private]

Функцията достъпваща се от обект от тип LinkedList водеща до реализацията на алгоритъма Quick sort.

!

Аргументи

head	Корена (началото) на списъка

Връща

Указател от тип Node, тоест един елемент на списъка (корена).

Пренасочва действието си към функция quickSortRecur(Node*, Node**, Node**);

Връща

Указател тип Node от функцията quickSortRecur(Node*, Node**, Node**), сочещ началото на списъка.

3.1.3.10 quickSortRecur()

```
Node * LinkedList::quickSortRecur (
            Node * head,
            Node ** newHead,
            Node ** newEnd)   [private]
```

Реализацията на Quick sort.

!

Аргументи

Корена,новото	начало, новия край от указатели след сортиране.

Връща

Указател от тип Node, тоест един елемент на списъка.

Аргументи

head	началото на списъка
newHead	Новото началото на списъка след извършване на сортирането.
newEnd	Новият край на списъка след извършване на сортирането.

1. В началото се извършва условна проверна на това дали подадения указател към началото и неговия следващ са валидни. Ако не са, се пренасочват новите начало и край, към това което всъщност сочи.
2. Ако проверката не се осъществи и имаме подаден валиден указател, то се продължава към избирането на ос (pivot), която е всъщност корена на списъка.
3. Избира се следващия елемент на оста, с който след това се обхожда целия списък, сравнявайки се стойностите на конкретния елемент с неговия следващ.
4. Пренареждат се при неспазена наредба, указана във вписаното условие на зависимост, след което се продължава към следващия елемент от списъка.
5. Процедурата се изпълнява рекурсивно, за да се извадят указателите сочещи началото на списъците съответно наредените стойности по-малки от оста и по-големи от оста.
6. Обединяват се и се извежда указателя сочещ началото на вече наредения списък.

Връща

Указател тип Node, сочещ началото на списъка.

3.1.3.11 sortWithMergeSort()

void LinkedList::sortWithMergeSort ()

!

3.1.3.12 sortWithQuickSort()

void LinkedList::sortWithQuickSort ()

!

3.1.4 Член данни Документация

3.1.4.1 head

Node* LinkedList::head [private]

Документация за клас генериран от следните файлове:

- LinkedList.h
- LinkedList.cpp

3.2 Node Структура Препратка

#include <LinkedList.h>

Общодостъпни членове функции

- Node (int val)
  Това е конструкторът на елемент.

Общодостъпни атрибути

- int data
  Данните, които се съдържат в елемента.
- Node * next
  Указателя, който сочи към следващия елемент в списъка.

3.2.1 Подробно описание

!

Node е структура описваща един елемент на единично свързан списък.

3.2.2 Конструктор & Деструктор Документация

3.2.2.1 Node()

Node::Node (
int val)

Дефиниция конструктора за Node.

!

Аргументи

val	Стойността, с която се инициализира параметъра data в елемента.
val	Стойност за аргумент, която да се подава към конструктора.
data	Копира стойността на val.
next	Стойност по подразбиране - нулевия указател 0x0.

3.2.3 Член данни Документация

3.2.3.1 data

int Node::data

3.2.3.2 next

Node* Node::next

Документация за структура генериран от следните файлове:

- LinkedList.h
- LinkedList.cpp

Глава 4

Файлове Документация

4.1 LinkedList.cpp Файл Справка

#include "LinkedList.h"

4.2 LinkedList.h Файл Справка

#include <iostream>
#include <cstdlib>

Класове

- class LinkedList
- struct Node

4.3 LinkedList.h

Вижте документацията за този файл.

```
00001 #pragma once
00002 #include <iostream>
00003 #include <cstdlib>
00004
00008 struct Node {
00009     int data;
00010     Node* next;
00011
00016     Node(int val);
00017 };
00018
00022 class LinkedList {
00023 private:
00024     Node* head;
00025
00026     // Mergesort функции
00032     Node* mergeSort(Node* head);
00033
00034
00041     Node* merge(Node* a, Node* b);
00042
```

```
00043
00050     void frontBackSplit(Node* head, Node** frontRef, Node** backRef);
00051
00052
00053
00054     // Quicksort функции
00060     Node* quickSort(Node* head);
00061
00062
00068     Node* quickSortRecur(Node* head, Node** newHead, Node** newEnd);
00069
00070
00071 public:
00075     LinkedList();
00076
00082     void copyFrom(const LinkedList& other);
00083
00087     ~LinkedList();
00088
00093     void clear();
00094
00099     void push(int data);
00100
00105     void print() const;
00106
00114     void fillRandom(int count, int lower, int upper);
00115
00119     void sortWithMergeSort();
00120
00124     void sortWithQuickSort();
00125 };
```

4.4 main.cpp Файл Справка

```
#include <iostream>
#include <iomanip>
#include <chrono>
#include "LinkedList.h"
```

Функции

- int main ()

Променливи

- const int size = 20

4.4.1 Функции Документация

4.4.1.1 main()

int main ()

4.4.2 Променливи Документация

4.4.2.1 size

const int size = 20

Азбучен указател

~LinkedList
    LinkedList, 6

clear
    LinkedList, 7
copyFrom
    LinkedList, 7

data
    Node, 12

fillRandom
    LinkedList, 7
frontBackSplit
    LinkedList, 7

head
    LinkedList, 11

LinkedList, 5
    ~LinkedList, 6
    clear, 7
    copyFrom, 7
    fillRandom, 7
    frontBackSplit, 7
    head, 11
    LinkedList, 6
    merge, 8
    mergeSort, 8
    print, 9
    push, 9
    quickSort, 9
    quickSortRecur, 10
    sortWithMergeSort, 11
    sortWithQuickSort, 11
LinkedList.cpp, 13
LinkedList.h, 13

main
    main.cpp, 14
main.cpp, 14
    main, 14
    size, 14
merge
    LinkedList, 8
mergeSort
    LinkedList, 8

next
    Node, 12

Node, 11
    data, 12
    next, 12
    Node, 11

print
    LinkedList, 9
push
    LinkedList, 9

quickSort
    LinkedList, 9
quickSortRecur
    LinkedList, 10

size
    main.cpp, 14
sortWithMergeSort
    LinkedList, 11
sortWithQuickSort
    LinkedList, 11