مینی پروژه اول

لیست پیوندی دوطرفه

بخش پیاده‌سازی

برای پیاده‌سازی این پروژه تنها مجاز به استفاده از کتابخانه‌های زیر می‌باشید:

Functions and Classes	Header File
همه توابع و کلاس‌ها	#include <iostream>
assert(predicate);	#include <cassert>

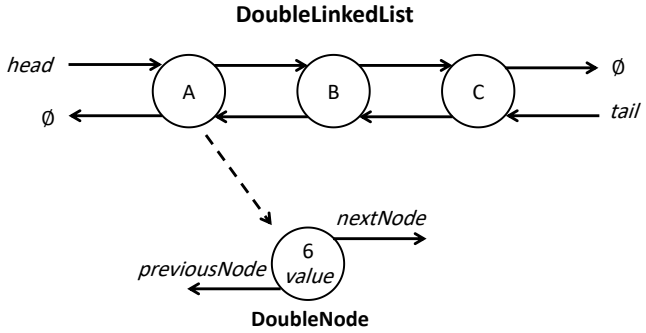
std::min(T, T);
std::max(T, T);
std::swap(T&, T&);
	#include <algorithm>

همه توابع	#include <cmath>
همه ثابت‌ها	#include <cfloat>
همه ثابت‌ها	#include <climits>

برای هر تابع کامنت‌گذاری کرده و موارد زیر را توضیح دهید:

- عملکرد تابع
- کاربرد هر پارامتر
- فرضیات تابع
- خطاهایی که ممکن است رخ دهد و نحوه مدیریت‌شان

نام کلاس شما باید دقیقا DoubleLinkedList بوده و در فایل double_linked_list.h پیاده‌سازی شود. توابع موردنیاز لیست پیوندی دوطرفه در ادامه توضیح داده شده‌اند. فایل امضای توابع ضمیمه تمرین گردیده است. برخی از توابع پیاده‌سازی شده‌اند و برخی باید توسط شما پیاده‌سازی شوند.



شکل ۱: نمایش یک نمونه لیست پیوندی دوطرفه (DoubleLinkedList) و ساختار کلاس هر گره (DoubleNode)

UML Class Diagram

DoubleLinkedList
- listHead:DoubleNode - listTail:DoubleNode - listSize:Integer
+ create():DoubleLinkedList + create(in dll: DoubleLinkedList): DoubleLinkedList + size():Integer + empty():Boolean + front():Integer + back():Integer + begin():DoubleNode + end():DoubleNode + find(in value: Integer): DoubleNode + count(in value: Integer):Integer + swap(inout list: DoubleLinkedList) + pushFront(in value: Integer) + pushBack(in value: Integer) + popFront() + popBack():Integer + erase(in value: Integer):Integer + destroy()

توضیحات فیلدها و توابع

فیلد/تابع	توضیحات
listHead	اشاره‌گر به ابتدای لیست
listTail	اشاره‌گر به انتهای لیست
listSize	تعداد عناصر موجود در لیست را نگه می‌دارد
create()	سازنده پیش‌فرض که یک لیست خالی ایجاد می‌کند. $O(1)$
create(DoubleLinkedList&)	یک لیست پیوندی دریافت کرده و همه عناصر آن را در خود کپی می‌کند. $O(n)$
size()	تعداد عناصر موجود در لیست را برمی‌گرداند. $O(1)$
empty()	در صورتی که لیست خالی باشد، مقدار true و درغیراینصورت false برمیگرداند. $O(1)$
front()	مقداری که در ابتدای لیست قرار دارد را برمی‌گرداند. در صورت خالی بودن لیست خطایی از نوع Underflow پرتاب می‌کند. $O(1)$
back()	مقداری که در انتهای لیست قرار دارد را برمی‌گرداند. در صورت خالی بودن لیست خطایی از نوع Underflow پرتاب می‌کند. $O(1)$
begin()	آدرس گره آغازین (head) در لیست را برمی‌گرداند. $O(1)$
end()	آدرس گره پایانی (tail) در لیست را برمی‌گرداند. $O(1)$
find(Integer)	آدرس اولین گره‌ای که مقدارش با مقدار پارامتر ورودی برابر است را برمی‌گرداند. در صورت یافت نشدن مقدار مربوطه، مقدار nullptr برگردانده می‌شود. $O(n)$
count(Integer)	تعداد عناصری که مقدارشان با مقدار پارامتر ورودی برابر است را برمی‌گرداند. $O(n)$
swap(DoubleLinkedList)	لیست پیوندی ورودی را با خود (this) جابجا می‌کند. $O(1)$
pushFront(Integer)	گره جدیدی با مقدار داده شده در ورودی ایجاد کرده و در ابتدای لیست درج می‌کند. $O(1)$

pushBack(Integer)	گره جدیدی با مقدار داده شده در ورودی ایجاد کرده و در انتهای لیست درج می‌کند. $O(n)$ یا $O(1)$!!!
popFront()	گره آغازین لیست را حذف می‌کند. در صورت خالی بودن لیست خطایی از نوع Underflow پرتاب می‌کند. $O(1)$
popBack()	گره پایانی لیست را حذف می‌کند. در صورت خالی بودن لیست خطایی از نوع Underflow پرتاب می‌کند. $O(1)$
erase(Integer)	همه گره‌هایی که مقدارشان با مقدار پارامتر ورودی برابر است را حذف کرده و تعداد گره‌های حذف شده را برمی‌گرداند. $O(n)$

برای پیمایش لیست پیوندی از ابتدا تا انتها می‌توانید به صورت زیر عمل کنید:

```
DoubleLinkedList list;

list.push_front( 0 );

for ( auto ptr = list.begin(); ptr != nullptr; ptr = ptr->next() ) {

    std::cout << ptr->value() << ' ';

}
```

نکته: در صورتی که از کلمه نوع auto استفاده کنید، کامپایلر به صورت خودکار نوع متغیر را بر اساس عبارت سمت راست تشخیص می‌دهد. بجای "`auto ptr`" می‌توان نوشت "`DoubleLinkedList::DoubleNode *ptr`".

بخش تعاریف ثابت

در این بخش به توضیح کلاس‌ها و تعاریفی که به صورت ثابت در اختیار شما قرار داده شده و باید بدون تغییر از آن‌ها استفاده کنید، پرداخته شده است.

فایل Exception.h شامل استثناهای موردنیاز شما در پیاده‌سازی پروژه است:

استثنا	توضیحات
Underflow	خالی بودن لیست
Overflow	پر بودن لیست
DivisionByZero	تقسیم بر صفر
IllegalArgument	پارامتر ورودی نامعتبر
OutOfRange	اندیس خارج از محدوده

برای پرتاب کردن یک استثنا از موارد بالا به صورت زیر عمل می‌کنیم:

```
#include "exception.h"

...

throw Underflow();
```

فایل tester.cpp برای آزمون پیاده‌سازی شما نوشته شده است. این کلاس، دستوراتی را به شکل خلاصه شده از ورودی دریافت کرده و توابع متناظر از لیست پیوندی پیاده‌سازی شده توسط شما را فراخوانی می‌کند. صحت پیاده‌سازی شما براساس مقادیری که این فایل در خروجی استاندارد چاپ می‌کند، بررسی خواهد شد. شما نیز می‌توانید با استفاده از همین فایل به تست پیاده‌سازی خود بپردازید.

دستور	توضیحات
new	یک لیست پیوندی جدید ایجاد می‌کند
copy	یک لیست پیوندی جدید ایجاد کرده و لیست پیوندی قبلی را به سازنده آن ارسال می‌کند
end	آخرین لیست پیوندی ایجاد شده را حذف می‌کند
exit	تست را خاتمه می‌دهد
size	اندازه آخرین لیست پیوندی ایجاد شده را چاپ می‌کند
print	آخرین لیست پیوندی ایجاد شده را چاپ می‌کند
empty	خالی بودن/نبودن آخرین لیست پیوندی ایجاد شده را بررسی می‌کند
swap	تابع swap برای آخرین لیست پیوندی ایجاد شده و لیست قبل از آن فراخوانی می‌کند
push_front x	تابع pushFront را با پارامتر ورودی x برای آخرین لیست پیوندی ایجاد شده فراخوانی می‌کند
push_back x	تابع pushBack را با پارامتر ورودی x برای آخرین لیست پیوندی ایجاد شده فراخوانی می‌کند
pop_front	تابع popFront را برای آخرین لیست پیوندی ایجاد شده فراخوانی می‌کند
pop_back	تابع popBack را برای آخرین لیست پیوندی ایجاد شده فراخوانی می‌کند
find x	تابع find را با پارامتر ورودی x برای آخرین لیست پیوندی ایجاد شده فراخوانی می‌کند
count x	تابع count را با پارامتر ورودی x برای آخرین لیست پیوندی ایجاد شده فراخوانی می‌کند
erase x	تابع erase را با پارامتر ورودی x برای آخرین لیست پیوندی ایجاد شده فراخوانی می‌کند

یک نمونه ورودی

```
new
 push_front 3
 push_front 2
 push_front 1
 print
 push_back 4
 push_back 5
 print
 find 3
 erase 1
 print
 erase 6
 print
end
```

خروجی نمونه بالا

```
head->1->2->3->0
tail->3->2->1->0
head->1->2->3->4->5->0
tail->5->4->3->2->1->0
3
1
head->2->3->4->5->0
tail->5->4->3->2->0
0
head->2->3->4->5->0
tail->5->4->3->2->0
```