به نام خدا

گزارش تمرین سری 5

برنامه نویسی پیشرفته(1)

رضا مرادی 9623100

- **Apmaxheap.hpp**

دیفالت کانستراکتور و دیستراکتور را به صورت دیفالت در نظر میگیریم نیازی به دیلیت کردن وکتور arr نداریم زیرا حتی اگر ورودی وکتور از نوع پوینتر باشد باز هم در کلاس وکتور دیلیت میشود در صورت دیلیت کردن با ارور ()free روبرو میشیم. (ینی فضای حافظه قبلا از کامپایلر گرفته شده و نیازی به پاک کردن ندارد.)

خط 1 تا 4- در این کانستراکتور ورودی به صورت رفرنس و کانست هست زیرا فقط میخواهیم آبجکت ap را داخل this بریزیم ینی داخل کلاسی که کال شده از طرفی با اینکار از کپی شدن اضافه ی ap جلوگیری میکنیم برای اینکار فقط کافیست متغیر arr آبجکت ap را داخل متغیر arr آبجکت this بریزیم با اینکار محتویات ap به this انتقال داده شده است در حالی که ارتباطی بین این دو آبجکت وجود ندارد.

خط 6 تا 24- چون ورودی این تابع rvalue میباشد و ما دنبال این هستیم که آدرس ورودی را ذخیره کنیم اگه ورودی تابع را به صورت &node_value در نظر بگیریم همیشه ورودی تابع ثابت خواهد (ینی مثلا به ازای ورودی های 4 و 5 یا 8 همواره یک آدرس مساوی ذخیره میشود چون &node_value آدرس lvalue هست) ابتدا متغیر i را میسازیم که از سایز arr یکی بیشتر است چونکه i قرار است از یک شروع شود همچنین یک متغیر *T به نام temp میسازیم که آدرس node_value را دارد از آنجایی که node_value داخل فضای تابع به صورت lvalue هست پس باید داخل آن &node_value را بریزیم در نهایت temp برای swap کردن استفاده میشود. در خط 10 بررسی میکنیم که اگر arr خالی هست آدرس node_value داخل arr، push_back بشود اگر اینطور نبود وارد else میشود سپس push_back را انجام میدهیم بعد بررسی میکنیم که هیچ نودی از Parent خودش بزرگتر نباشد برای اینکار داخل while این شرط را بررسی میکنیم اندیس arr از i یکی کمتر است داخل while ، swap را انجام میدهیم سپس به نود parent میرویم (i=i/2) و دوباره شرط را بررسی میکنیم تاز مانی که کوچکترین اندیس ورودی arr منفی نباشد در غیر صورت به segmentation fault میرسیم. در آخر this* را return میکنیم (خروجی تابع باید به صورت & باشد تا اگر تابع push به صورت زنجیری کال شود مشکلی ایجاد نشود و خروجی هر push بتواند به صورت ورودی push دیگر داده شود.)

خط 26 تا 49-برای حالتی که ورودی push از نوع Student است باید تابع را به صورت خاص تعریف کنیم که طبق قاعده ی Class template specialization این کار را انجام داده ایم طبق این قاعده لازم نیس prototype جدید تعریف کنیم و T را برابر Student قرار میدهیم و داخل تگ template را به صورت خالی رها میکنیم چون نیازی به استفاده از T نداریم مانند بخش قبل عمل میکند فقط عامل مقایسه

روی معدل صورت میگیرد.(لازم به توضیح است که نیازی نداریم کلاس Student را اینکلود کنیم زیرا یه نوع متغیر template است مانند حالتی که template مثلا به جای اینکه از نوع int باشد از نوع list باشد و خبری نداریم که قرار از از list استفاده بشود که بخواهیم این کلاس را اینکلود کنیم.)

فقط برای حالتی که ورودی از نوع lvalue است باید از &node_value برای ورودی استفاده کنیم که چون یک تابع جدید حساب میشه لازمه که prototype جدید را نیز بنویسیم برای حالتی که ورودی &&node_value است میتوانیم تابع قبل را کال کنیم زیرا node_value داخل فضای تابع lvalue حساب میشود.

خط 51 تا 115- برای حالتی که arr خالیست فقط return میکنیم زیرا لازم نیست چیزی را پاک کنیم و برای حالتی که arr فقط یک عضو دارد همان را پاک میکنیم برای حالت دیگر ابتدا در خط 61 پرنت اصلی یا ریشه را برابر اخرین عضو arr قرار میدهیم سپس arr را ()pop_back میکنیم مانند این است که ریشه پاک شده حال باید بررسی کنیم که همه ی parent ها از child ها بزرگتر باشند اینکار تحت یک while و در دو بخش انجام شده است باید بچه ای جای پرنت قرار بگیرد که خودش نسبت به بچه ی دیگر بزرگتر است برای همین خط 65 به این صورت نوشته شده است سپس اگر left child جایگزین parent شد باید i=2*i شود و اگر right شد i=2*i+1 مطابق فرمول ها اگر هیچ کدوم بزرگتر نبود که break میکنیم سپس بررسی میکنیم که بزرگترین order داخل while ینی i*2 از سایز arr بیشتر نشود که به segmentation fault بر نخوریم سپس همینکار را برای student بر اساس معدل انجام میدهیم.

خط 117 تا 122- در اینجا وقتی مثلا دو آبجکت ap1 و ap2 داریم و {ap1{ap2+95 را اجرا میکنیم توقع داریم ap2 تغییری نکند ولی ap1 برابر ap2+95 بشود پس یک آبجکت از کلاس داخل scope تابع میسازیم و آنرا بعد از push کردن return میکنیم علاوه براین در خط 120 از std::forward استفاده میکنیم زیرا داخل تابع node_value به صورت lvalue است اما ورودی push ،rvalue است پس با کمک این سینتکس biding را انجام میدهیم.

خط 124 تا 128- نیاز به توضحی خاصی ندارد میتوانستیم تابع را به صورت void تعریف کنیم اما با توجه به این تعریفی که داریم مثلا برای 3 آبجکت ap1, ap2, ap3 کد {ap1{ap2=ap3 نیز به درستی انجام میشود.

خط 130 تا 158- نیاز به توضیح خاصی ندارد فقط توابع show به صورت const تعریف شدند که چیزی را تغییر ندهند و این تابع نیز برای کلاس student به صورت درست specialize شده است.

- **Student.cpp**

خط 1 تا 18- نیاز به توضیح خاصی ندارد مطابق مطلوب کد ها نوشته شده است.

خط 20 تا 23- تابع operator<< به صورت کلی تعریف شده که چون متد کلاس نیست برای اینکه داخل فضای کلاس تعریف شود باید friend بشود و طبیعتا ورودی تابع باید ostream و کلاس student باشد که student را به صورت const میدهیم که داخل تابع تغییری نکند و برای جلوگیری از کپی شدن هردو به صورت رفرنسی وارد تابع میکنیم و برای اینکه به صورت زنجیری نیز کار بکند خروجی نیز باید رفرنسی باشد.

در نهایت همه ی تست ها پاس میشوند:

```
RUNNING TESTS ...
[==========] Running 13 tests from 1 test suite.
[----------] Global test environment set-up.
[----------] 13 tests from APMidTest
[ RUN      ] APMidTest.Test0
APMaxHeap:
20, 12, 5, 10, 8,
[       OK ] APMidTest.Test0 (1 ms)
[ RUN      ] APMidTest.Test1
APMaxHeap:
20, 12, 5, 10, 8,
[       OK ] APMidTest.Test1 (0 ms)
[ RUN      ] APMidTest.Test2
APMaxHeap:
20, 12, 5, 10, 8,
[       OK ] APMidTest.Test2 (0 ms)
[ RUN      ] APMidTest.Test3
APMaxHeap:
10,
[       OK ] APMidTest.Test3 (3 ms)
[ RUN      ] APMidTest.Test4
APMaxHeap:
10,
[       OK ] APMidTest.Test4 (0 ms)
[ RUN      ] APMidTest.Test5
APMaxHeap:
28, 10, 20,
APMaxHeap:
20, 10,
[       OK ] APMidTest.Test5 (0 ms)
```

```
[ RUN      ] APMidTest.Test5
APMaxHeap:
28, 10, 20,
APMaxHeap:
20, 10,
[       OK ] APMidTest.Test5 (0 ms)
[ RUN      ] APMidTest.Test6
APMaxHeap:
28, 10, 20,
APMaxHeap:
95, 28, 20, 10,
[       OK ] APMidTest.Test6 (0 ms)
[ RUN      ] APMidTest.Test7
APMaxHeap:
110, 28, 20, 10,
APMaxHeap:
110, 28, 20, 10,
[       OK ] APMidTest.Test7 (1 ms)
[ RUN      ] APMidTest.Test8
APMaxHeap:
9423037, 9423102, 9423091, 9423013,
[       OK ] APMidTest.Test8 (0 ms)
[ RUN      ] APMidTest.Test9
APMaxHeap:
9423091, 9423102, 9423013,
[       OK ] APMidTest.Test9 (0 ms)
[ RUN      ] APMidTest.Test10
APMaxHeap:
9423037, 9423013, 9423091,
APMaxHeap:
9423091, 9423013,
[       OK ] APMidTest.Test10 (0 ms)
[ RUN      ] APMidTest.Test11
APMaxHeap:
9423037, 9423013, 9423091,
Student: 9423013, 18.1
[       OK ] APMidTest.Test12 (0 ms)
[----------] 13 tests from APMidTest (6 ms total)

[----------] Global test environment tear-down
[==========] 13 tests from 1 test suite ran. (8 ms total)
[  PASSED  ] 13 tests.
<<<SUCCESS>>>
```