بسم الله الرحمن الرحیم


**دانشگاه صنعتی امیرکبیر**

**(پلی تکنیک تهران)**

**دانشکده مهندسی برق**


# گزارش تمرین سری چهارم

## برنامه نویسی پیشرفته


**دکتر جهانشاهی**

**وحید پوراکبر**

**9523022**

# Git

**https://github.com/vahid-aki/HW4.git**

# C++

## سوال 1:

### Move semantics :

با استفاده از ریفرینس میتوان به جای کپی کردن متغیر میتوان فقط یک متغیر دیگری ایجادکرد که دقیقا به همان آدرس متغیر اول اشاره میکند و در واقع فقط اسم آن متفاوت است. این کار مزایای زیادی دارد برای مثال میتوان ازکپی کردن کد جلوگیری کرد و یا میتوان متغیر دوم را به صورت const تعریف کرد که در این صورت نمیتوان از طریق دومی اولی را نیز تغییر داد و بسیاری کاربردهای دیگر مثلا در ارسال به تابع یا حتی درکلاس ها برای نمونه اگر در داخل کلاس ما متغیر های داینامیک داشته باشیم باید حتما توابع move constructor و destructor و =operator را تعریف کنیم. چون در غیر این صورت متغیر های داینامیک را چند بار حذف می کند و باعث segmentation error میشود.

### Polymorphism :

چند ریختی یعنی این که چند تا نوع یا type متفاوت رابط یکسانی داشته باشند و بشود به شکل یکسانی به آن ها دسترسی داشت.

چندریختی ها به منظور استفاده از توابع کلاسهایی که از هم ارث برده اند استفاده میشود. برای مثال وقتی ما پوینتری ازکلاس والد تعریف میکنیم که به کلاس بچه اشاره میکند در این صورت میتوان از توابع هر دو استفاده کرد که برای کاپپایلر از توابع والد استفاده می کند برای جلوگیری از این موضوع میتوان آن تابعی که

مد نظر هست را به صورت virtual تعریف کرد. همچنین در موقع destruct نیز اشتباها destruct میکند و و فقط والد را destruct میکندکه برای همین باید destructorها را نیز به صورت virtual تعریف کرد.

چندریختی جاهایی استفاده میشود که نیاز باشد بصورت generic کد بنویسیم (یعنی این که یک کد خاص با انواع متقاوت کار بکند)

## pure abstract:

اگر در داخل یک کلاس فقط یا حداقل یکی از توبع virtual را برابر صفر قرار دهیم مانند: virtual void print() = 0; در این صورت آن کلاس abstract میشود که در این صورت به صورت مستقیم نمیشود از آن کلاس آبجکتی درست کرد البته میتوان به صورت پوینتری یا رفرنسی و استفاده از آبجکتهای کلاس مشتق شده(کلاس بچه) استفاده کرد. برای همین اکثر مواقع کلاس والد را که بقیه کلاس ها از آن مشتق گرفته اند را به این صورت تعریف میکنند مانند سوال 2 این سری تمرینات.

## Override :

این کلیدواژه زمانی کاربرد پیدا میکندکه ما کلاس هایی داشته باشیم که از هم ارث ببرند و داخل این کلاس ها توابع virtual باشد در این صورت با گذاشتن کلیدواژه override به کامپایلر یادآوری میکنیم که ما در این توابع از virtual استفاده کرده ایم و در واقع خود کامپایلر این را چک میکند و اگر اشتباهی صورت گفته باشدکامپایلر اررور میدهد. برای مثال اشتباه میتواند به این صورت باشدکه ما درکلاس والد مثلا این تابع را داشته باشیم : virtual void func() و درکلاس مشتق میخوایم تابعی تعریف کنیم که با تابع کلاس والد یکی باشد اما اشتباهی صورت میگیرد مثلا void func(int a) در این صورت کامپایلرکد را درست اجرا میکند ولی این کدی که اجرا میکند منظور ما نیست برای همین به این صورت تعریف میکنیم void func(int a) override تا کامپایلر حواسش باشد.

## Inline :

وقتی که تابع inline تعریف شده باشدکامپایلر به صورت مستقیم کد داخل تابع را داخل main برنامه فرار میدهد و اگرتابع inline نباشد از call برای صدا زدن procedure استفاده میکند ، در این صورت تابع سریعتر اجرا میشود.

باید توجه کرد که در تابعی که inline شده علاوه بر این که عملیات push و pop انجام نمیشود به این دلیل که وقتی دستور call داخل اسمبلی استفاده میشود cpu مجبور به صبر میشود تا دستور کاملا اجرا شده و بعد بقیه دستورات را اجرا کند حالا اگر مثلا مقدار بازگشتی از تابع داخل یک دستور شرطی استفاده شده بود و ما call نداشته باشیم می تواند با استفاده از branch prediction نتیجه دستورات شرطی تابع را پیش بینی کرده و قسمت های بعدی کد را هم همزمان با تابع اجرا کندکه تاثیر خیلی زیادی در سرعت کاپایل خواهد داشت.

البته Inline کردن توابع حجم برنامه را زیاد می کند پس فقط توابعی که طول کوتاهی دارند و دفعات زیادی هم استفاده نمیشنوند را inline میکنند.

## Explicit :

explicit برای سازنده کلاس تعریف میشه که کامپایلر مقید میشود که فقط و فقط همان نوع داده ای را قبول کندکه در تعریف آمده و در غیر اینصورت در هنگام کامپایل خطا صادر می کند. برای مثال:

```
class Complex
{
  Complex(double r = 0.0, double i = 0.0) : real(r), imag(i) {}

  // A method to compare two Complex numbers
  bool operator == (Complex rhs)
};

int main()
{
  // a Complex object
```

```
    Complex com1(3.0, 0.0);

    if (com1 == 3.0)
}
```

درست اجرا میشود اما کد زیر با ارور روبرو میشود:

```
explicit Complex(double r = 0.0, double i = 0.0) : real(r), imag(i) {}
```

ارور:

```
no match for 'operator==' in 'com1 == 3.0e+0'
```

برای رفع این مشکل میتوان به صورت زیر عمل کرد:

```
  if (com1 == (Complex)3.0)
```

## سوال 2:

با اجرای کد مشاهده میشود:

```
Str 0  :  vec's size: 1001   &   vec's capacity: 2000

.

.

.

Str 999  :  vec's size: 2000   &   vec's capacity: 2000
```

همانطورکه مشاهده میشود سایز وکتور از 1000 شروع میشود علت این است‌که هزار تای اول را برابر با صفر یا null قرار داده(بستگی به‌کامپایلر دارد) و بعد از آن را push میکند. ظرفیت نیز با push کردن ارایه 1001 ام به اندازه 1000 تا اضافه میشود.

اما در حالت دوم مشاهده میشود‌که

Str 0 : vec's size: 1 & vec's capacity: 1000

.

.

.

Str 999 : vec's size: 1000 & vec's capacity: 1000

بااستفاده از reserve کاپایلر به اندازه 1000 تا ظرفیت جدا میکند و در اختیار ما قرار میدهد و با هر بار push کردن نیز سایز اضافه میشود.

## سوال3:

در این سوال اکثر مواردکامنت شده اند و بحث در مورد virtualنیز در جوب سوال 4 آمده اند. فقط یک نکته و آن هم اینکه به جای تعریف اپراتور = و + به صورت جدا چون در هنگام جمع شکل با پوینت جمع شده و در داخل خود شکل ریخته میشود برای همین فقط اپراتور =+ است.

## سوال 4:

تابع print چون توسط آبجکت‌کلاس shape صدا زده می شود و از آنجایی‌که

(آبجکت از نوع shape به صورت مستقیم نداریم و به صورت پوینتری که به آبجکت های کلاس مشتق اشاره میکند برای همین کلاس shape به صورت abstract میباشد برای همین یکی از توابع کلاسی که به صورت virtual میباشد را برابر 0 میگذاریم که این تابع تابع print میباشد)

برای همین در واقع print به کلاس های مشتق شده اشاره میکند در حالی که توسط کلاس والد صدا زده شده است برای همین این تابع را به صورت virtual تعریف میکنیم.

اما توابع مساحت و حجم در main نیامده اند و در داخل کلاس ها فقط توسط خود کلاسی که میخواهیم این تابع را روی آن اجرا کنیم صدا زده شده برای همین نیازی به تعریف این دو تابع به صورت virtual نمی باشد.

# سوال 5:

در این سوال از template ها استفاده شده به این علت که کلاس stack بتواندکلاس های مختلف را در خود ذخیره کند. همانطورکه در قسمت main آمده دیده میشود که از shared_ptr ها استفاده شده به همین دلیل در هنگام push کردن از make_shared استفاده شده و با هر بار push کردن این به اول لیست اضافه میشود. بقیه قسمت ها تقریبا مشخص می باشد همچنین بیشتر مواردکامنت شده اند. فقط در هنگام pop کردن خروجی stack.pop() از نوع کلاس ذخیره شده می باشد برای همین getText برای کلاس دومی تعریف شده است.

درکل در هنگام push به اول لیست اضافه شده در هنگام pop از اول لیست برداشته میشود.

# سوال 6:

در هنگام استفاد از remove در واقع 2 را از وکتور حذف نمیکند بلکه فقط آن را در نظر نمیگیرد برای حذف 2 این کار انجام شده که اول iterator مقدار 2 را پیدا کرده با استفاده از swap آنها را به آخر انتقالداده و بعد آن را pop میکند.

برای بقیه موارد از دستورات copy_if و landa function ها استفاده شده است.

فقط در مورد قسمت پیدا کردن مقدار فاصله از میانگین کد نوشته شده به نظر درست می آمد ولی با ارور نامشخصی روبرو شدم برای همین این قسمت کامنت شده است.

## سوال 7:

در این سوال دقیقا منظور سوال مشخص نشده مثلا هدف از آرایه a چیست؟ و یا اگر قرار نیست از حلقه استفاده کنیم پس چگونه مقدار اولیه وکتور b را بدهیم و...

برای همین کد نوشته شده ممکن است با منظور سوال متفاوت باشد.

# Backend

این سوال با استفاده راهنمایی های TA این قسمت نوشته شده است.