

(Q1)

(کد 1)

این کد کامپایل می شود. ابتدا متغیری از نوع `int` به نام `a` تعریف شده است. سپس یک پوینتر از نوع `int*` تعریف شده است. از نوع تعریف پوینتر میتوان فهمید که خود این پوینتر `const` است و نمیتوان آدرس جدیدی در آن نوشت. اما میتوان از طریق این پوینتر متغیر را که به آن اشاره میکند عوض کرد. در خط بعد از طریق پوینتر به متغیر `b` یک واحد اضافه می شود و در خط بعد مستقیماً به خود `a` دوباره یک واحد اضافه می شود بنا براین با دستور چاپ در خط بعد مقدار متغیر `a` یعنی 4 چاپ میشود. همچنین پوینتر `b` که آدرس یک خانه در حافظه در آن ذخیره شده است (آدرس `a`) نیز چاپ میشود.

(کد 2)

کد کامپایل نمی شود. ابتدا متغیری از نوع `const int` به نام `a` تعریف شده است. سپس دو متغیر `z` از نوع `int` به نام های `c` و `d` تعریف شده است. سپس یک پوینتر از نوع `int*` تعریف شده به نام `b` که به `a` اشاره میکند از نحوه تعریف این پوینتر می توان فهمید که خود این پوینتر `const` نیست ولی نمیتوان از طریق آن مقدار متغیری که به آن اشاره میکند را تغییر داد. در بخش بعدی آدرس `c` در پوینتر `b` نوشته می شود (این کار مشکلی ندارد زیرا خود پوینتر `const` نیست) از این به بعد این پوینتر به `c` اشاره می کند. در خط بعد اگر کد کامپایل می شد متغیر `a` و پوینتر `b` و محتوای پوینتر `b` چاپ میشد. (مقدار `a` 10 است و مقدار `b` آدرس خانه ای از حافظه است که `c` در آن ذخیره شده است و محتوای پوینتر همان `c` است که 20 است). در خط بعد یک پوینتر از نوع `int*` به نام `e` ایجاد می شود که آدرس `c` در آن ذخیره می شود. از نوع تعریف این پوینتر مشخص است که خود این پوینتر `const` است اما میتوان از طریق آن متغیری که به آن اشاره می کند را تغییر داد. در خط بعد برنامه نویس می خواهد آدرس `d` را در `e` ذخیره کند که `error` برنامه در همین خط است زیرا همانطور که گفتیم خود پوینتر `const` است و نمیتوان آدرس جدیدی در آن نوشت. در خط بعد متغیر `d` و پوینتر `e` و محتوای پوینتر `e` قرار است چاپ شود که اگر کد کامپایل می شد چاپ میشد (مقدار `d` 30 است و پوینتر `e` آدرس خانه `c` در آن ذخیره شده و محتوای پوینتر `e` هم همان `c` است که مقدارش 20 است)

(کد 3)

نوع متغیر خط 3: متغیر از نوع `const char*` است (به نام `p1`) که آدرس عنصر اول رشته ی کاراکتر `name` یعنی "A" در آن ذخیره شده (خانه های بعد از `p1` در حافظه به ترتیب آدرس کاراکترهای بعدی رشته اند). از نوع تعریف پوینتر می توان فهمید خود پوینتر `const` نیست اما نمی توان از طریق آن رشته ای که به آن اشاره میکند را تغییر داد.

حکم خط 5: مجاز است زیرا همانطور که گفتیم خود پوینتر `const` نیست و میتوان آدرس جدیدی در آن ذخیره کرد.

حکم خط 8: مجاز نیست و `error` دارد زیرا همانطور که گفتیم نمیتوان از طریق `p1` رشته را تغییر داد.

حکم خط 9: مجاز نیست و `error` دارد زیرا پوینتر از نوع `char*` برای اشاره به یک متغیر از نوع `char` است ولی رشته که میان `double quotes` است از نوع `const char*` است و یک پوینتر `char*` نمیتواند به آن اشاره کند

کد 4)

خط 1 : آرایه تک بعدی 10 تایی از نوع int که با new کردن پویتر p1 به دست می آید

خط 2 : آرایه 10 تایی از نوع int\* (آرایه ای از پویتر ها)

خط 3 : function pointer به نام p3 که آرایه تک بعدی از نوع int را ورودی می گیرد.

خط 4 : آرایه 10 تایی از function pointer ها است که هر کدام از این function pointer ها آرایه دو بعدی که تعداد سطر هایشان مشخص نیست ولی 10 ستون دارند را به عنوان ورودی میگیرند.

خط 5 : آرایه دو بعدی 10\*10 از نوع int . تفاوتش با آرایه دو بعدی معمولی این است که بر خلاف آرایه دو بعدی دینامیک معمولی هر 100 خانه پشت سر هم allocate میشوند .

کار اضافه -- درکدی که در پوشه Q1 است من آدرس تمام 100 خانه آرایه که به این ترتیب تعریف شده را به ترتیب چاپ کردم که مشاهده می شود تمام خانه ها به ترتیب اند. همچنین خانه های آرایه دو بعدی معمولی 10\*10 را نیز چاپ کردم که مشاهده می شود خانه های به فرم 10\*n ام و 10\*n+1 پشت سر هم نیستند!

(Q2

توابع مورد استفاده:

الگوریتم به این شکل است که ابتدا کل فایل را در قال یک string میخوانیم. سپس با معرفی علائم نگارشی و به طور کلی کاراکتر های که جزو حروف الفبا نیستند فایل را pars میکنیم و در یک vector تمام کلمات فایل را ذخیره میکنیم. میدانیم کلمات دو دسته اند (1: کلماتی که فقط از حروف الفبا تشکیل شده اند 2) کلماتی که ممکن است میان آن ها character های غیرالفبایی باشد مانند : space-X و SS\_spring و غیره. ادامه الگوریتم به این شکل است که یک تابع به نام split\_1 نوشتیم که کلمات دسته دوم را نیز از به کلماتی میشکند که خروجی های این تابع کلماتی از دسته اول اند. یک تابع split\_2 نیز نوشتیم که کلمات دسته اول را از محل کاراکتر های a,e,i,o,u,A,E,I,O,U برش میدهد. خروجی این تابع کلماتی هستند که فقط از حروف بی صدا تشکیل شده اند. حال باید این کلمات چک شوند که 5 حرف صدادار یا بیشتر پشت سر هم در آن ها نیامده باشد. این کار با تابع check\_normal\_str انجام میپذیرد و خروجی یک Boolean است.

بخش اضافه بر سوال که انجام دادم: در سوال کلمات نوع دوم فقط به طوری در نظر گرفته میشود که کاراکتر "-" در میان آن باشد. اما کد من تمام کاراکتر های غیر الفبایی دیگر که در میان کلمات آمده اند را میفهمد. مثلا اگر کلمه همراه apostrophe باشد آن را تشخیص میدهد مانند I'am و it's و can't و ...

نحوه کار check\_normal\_str:

ورودی: string

خروجی: Boolean (خروجی true برای رشته های واجد شرایط صورت سوال و false برای بقیه ورودی ها)

چون ورودی این تابع کلمات دسته اول هستند پس ابتدا در این تابع آن ها را از صافی تابع `split_2` میگذرانیم و خروجی آن زیر رشته هایی است که همگی از حروف بی صدا تشکیل شده اند. حال با حلقه ساده همه این زیر رشته ها را چک میکنیم که ببینیم آیا 5 حرف بی صدا در آن ها آماده است یا نه. اگر حداقل در یکی از زیر رشته ها 5 حرف بی صدا یا بیشتر پشت سر هم آمده بود `true` بازگردانده میشود. در غیر این صورت `false` بازگردانده میشود.

نحوه کار `check_str` :

ورودی: `string`

خروجی: `Boolean` (خروجی `true` برای رشته های واجد شرایط صورت سوال و `false` برای بقیه ورودی ها)

ورودی این تابع کلماتی هستند که در آن ها کاراکترهای غیر الفبایی وجود دارند. پس باید ابتدا این ورودی ها را از صافی تابع `split_1` بگذرانیم حال خروجی های این تابع همه رشته های از نوع اول هستند. پس کافیت آن ها را ب تابع `check_normal_str` بدهیم.

به عنوان جمع بندی نحوه کار برنامه به این صورت است :

(1) ابتدا فایل به کلمات سازنده اش تجزیه میشود و در `vector` ذخیره میشود.

(2) سپس کلمات دسته اول تشخیص داده میشوند و به تابع `check_normal_str` تحویل داده میشوند.

(3) سپس کلمات دسته دوم نیز به تابع `check_str` تحویل داده میشوند

(4) کلماتی که به ازای آن ها خروجی دو تابع بالا برایشان `true` است چاپ میشوند

---

(Q3)

کد جزییات زیادی دارد و ذکر همه ی آن ها در گزارش به اطناب بی موردی می انجامد بنابراین من الگوریتم و بخش های مختلف کلاس را به صورت کلی مطرح میکنم.

`variable` های مورد نیاز کلاس :

ارایه دینامیک که عناصر در آن ذخیره می شود (`nums`) و ارایه دینامیک از جنس `bool*` که هم اندازه با ارایه `nums` است (`flags`) (اگر خانه `nums` پر باشد خانه `flags` 1 است در غیر این صورت 0 است.) متغیر `size` از نوع `int` که اندازه ارایه دینامیک است. دو متغیر از نوع `int` به نام های `reader_p` و `writer_p` که مکان `pointer` ها را روی ارایه نشان میدهد. و در نهایت متغیر به نام `capacity` از نوع `int` که مقدار جای خالی در صف را نشان می دهد.

`Method` های مهم کلاس:

(1) `constructor`: فایل را به عنوان ورودی می گیرد. ارایه های دینامیک کلاس همینجا `new` میشوند. همچنین `writer_p` و `pointer_p` و `capacity` هم همینجا تعیین میشود.

بخش اضافه بر صورت سوال که انجام دادم: با `catch` و `throw` عمل `error handling` را انجام دادم که اگر فایل به هر دلیلی درست باز نشد `error` بدهد.

2) `enqueue`: در محل فعلی `writer_p` مینویسد سپس `writer_p` را به محل جدیدش منتقل می کند اگر سمت راستش خالی بود و از ارایه بیرون نمی زد یکی به سمت راست می رود در غیر این صورت به ابتدای ارایه یعنی خانه صفر می رود. (البته در ابتدا چک میشود که ارایه پر نباشد)

3) `dequeue`: خانه که `reader_p` روی آن است خالی میکند و سپس `reader_p` را به محل جدیدش منتقل می کند. اگر سمت راستش خالی بود و از ارایه بیرون نمیزد یکی به سمت راست می رود در غیر این صورت به ابتدای ارایه یعنی خانه صفر می رود. (البته در ابتدا چک میشود که ارایه خالی نباشد)

4) `displayQueue`: به این صورت است که از خانه `reader_p` به سمت راست چاپ میکند تا انتهای ارایه. سپس از ابتدای آرایه تا `reader_p` چاپ میکند به این ترتیب ما صف را به درستی مشاهده میکنیم.

5) `destructor`: تمام متغیر های دینامیک کلاس را پاک میکند.

---

(Q4)

توابع مورد استفاده (سوال بسیار ساده است بنابراین فقط توابع مورد استفاده را معرفی میکنم)

`Find_min`: یک ارایه از اعداد را به عنوان ورودی میگیرد و کوچکترین عنصر را به عنوان خروجی میگیرد.

`Replace`: دو `int` به نام های `a` و `b` میگیرد و یک ارایه نیز میگیرد. و جای دو عنصر `a` و `b` ام ارایه را عوض میکند.

`Sort`: با استفاده از دو تابع بالا الگوریتم `selection sort` را اجرا می کند

---

نحوه کار با git

ابتدا با `git init` در کامپیوتر خود `local repository` ایجاد میکنیم سپس یک `repository` جدید در `github` یا سایت های دیگر درست می کنیم. حال `remote` را با استفاده از آدرس `http` که داریم با دستور `git remote add origin [URL]` میسازیم. حال فایل های مربوطه را به همراه فایل `.gitignore`. داخل `local repository` قرار می دهیم و با اجرای دستور های زیر به ترتیب آن ها را به سرور منتقل میکنیم:

`Git add [FILES]`

`Git commit -m [MESSAGE]`

`Git push origin master`

[لینک پروژه](#)

