بسم الله الرحمن الرحیم

پاسخ امتحان پایان ترم درس برنامه نویسی مقدماتی

قسمت تشریحی

امیر جهانشاهی

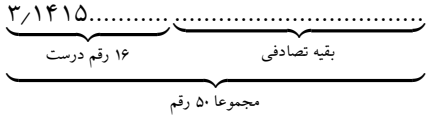
۲۸ دی ۱۳۹۴

۱. escape sequence ها تمامی کاراکتر هایی هستند که با \ شروع می شوند، مثل t\ ، n\ و .... علت وجود این هست که اغلب قابلیت نشان دادن این ها با صفحه کلید را نداریم. مثلا backspace را نمی توان به طور مستقیم نشان داد و برای همین از b\ استفاده می شود. همینطور t\ یا n\ که برای TAB یا new line هستند.

۲.

```
char str[2];
str[0] = n / 10 + '0';
str[1] = n % 10 + '0';
```

۳. در کد نوشته شده عدد π با ۵۰ رقم نمایش داده خواهد شد. اما باید دقت داشت که از عدد حدود ۱۶ به بعد که دقت long double می باشد، جواب تصادفی خواهد بود. این دانشجو درک درستی از تعداد رقم های نمایش داده شده در کامپیوتر و ربط آن با دقت long double نداشته است که خروجی را با ۵۰ رقم اعشار خواسته است. منطقی تر بود دقت در cout را حداکثر تا ۱۶ قرار می داد.



۴.

```
int Color;
char r{ Color & 0xFF };
char g{ (Color >> 8) & 0xFF };
char b{ (Color >> 16) & 0xFF };
```

توجه کنید که and کردن با 0xFF اختیاری می باشد و هنگامی که int داخل char ریخته می شود، ۸ بیت اول استفاده می شود.

۵.

```
style &= ~close_mask  //remove close button
style |= close_mask   //insert close button
```

دو تای دیگر به طور مشابه به همین صورت انجام می شود.

جواب راحت تر:

```
style ^= ~close_mask  //remove close button
style ^= close_mask   //insert close button
```

اگر ۱ باشد صفر می کند و بالعکس.

۶. (آ) داریم:

$a = 3, b = 4, a > b$ returns true and $a = 4 \Rightarrow ++a + b$

$\rightarrow a = 5, a + b = 9 \rightarrow c = 9$

به صورت باز:

```
if (a++ > b)
        c = ++a + b;
else
        c = a++ + b;
```

(ب) از آنجا که a از جنس اشاره گر می باشد و سیستم عامل ۳۲ بیتی است، آدرس ها ۴ بایت حافظه می گیرند ← sizeof(a) = 4. همچنین sizeof(char) = 1 ← output = 4

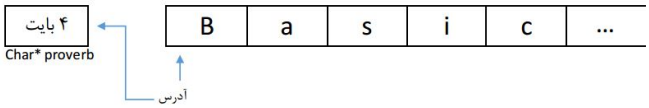
(ج) از آن جایی که داخل شرط i++ شده است، هنگامی که به دستور sum می رسیم، i برابر N می شود که دستور sum += arr[i] اجرا می شود. پس المان اول را نادیده می گیرد و از آخر یکی بیشتر از طول آرایه می رود که خطا است.

۷.

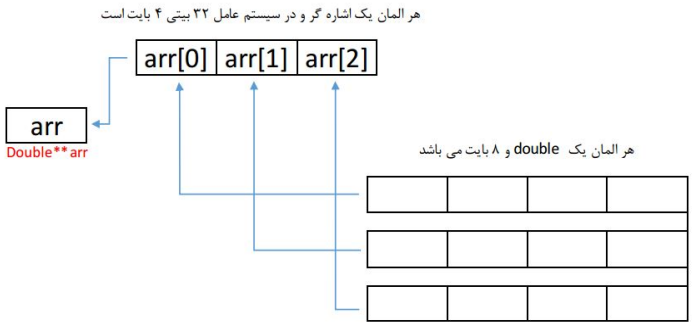
```
for(int i{2}; i <= n; i++)
        a*=i;

// -> in one line
for(int i{2}; i <=n; a *= i++)
```

۸. اولا اینکه بایستی آخرین المان را چاپ کند نه اولین المان. می شود کد را به صورت x[x.size() - 1] تغییر داد که حداقل درست اجرا می شود. در نهایت اینکه سطح سواد این دانشجو خیلی بالا نیست. چون اگراین آرایه ۱۰۰۰ المان داشته باشد، ۱۰۰۰ بار سایز آن عوض می شود تا محتویات را چاپ کند که کار بیهوده و غلطی می باشد. چون ۱۰۰۰ بار عوض کردن سایر وقت حافظه را خیلی می گیرد. بهتر بود یک for ساده روی المان های بردار و یا یک range based for می داشتیم.

۹. (آ) هر دو متغیر از جنس استاتیک می باشند، هم اشاره گر و هم آرایه.



(ب) هنگام delete کردن داخل for ابتدا آرایه های double پاک می شوند. سپس آرایه ۳ تایی از نوع اشاره گر در delete[ ] arr پاک می شود. این دو آرایه از جنس دینامیک هستند و متغیری که با قرمز نشان داده شده استاتیک است.



۱۰. اول از همه باعث تمیزی بیشتر می شود. ثانیا خیلی وقت ها نیازی نیست که cpp. را به کاربر بدهیم. فقط فایل h. و فایل o. که object می باشد را می دهیم و کد اصلی را از دید کاربر پنهان می کنیم.

۱۱. signature یک تابع عبارت است از تابع و تعداد و نوع آرگومان های ورودی. با داشتن اسم مشترک و آرگومان های ورودی متفاوت توابعی با یک نام ولی متفاوت خواهیم داشت که به این عمل یعنی تولید توابعی با نام یکسان ولی با آرگومان های ورودی متفاوت، function overloading گفته می شود. مثلا:

```
//two different functions
int f(int x);
int f(double x);
```

.۱۲

```
f(double x);  //call by value
f(double &x); //call by reference
```

چند تا فرق عمده دارند. یکی اینکه در حالت اول x از داخل main به داخل تابع کپی می شود. در صورتی که در دومی خودش می آید. اگر داخل تابع f اولی x عوض شود، x خارج از تابع عوض نمی شود. ولی داخل دومی عوض می شود. حالت اولی از لحاظ برنامه نویسی کد مطمئن تری را می دهد، چون مطمئن هستیم که x خارج از تابع عوض نمی شود، ولی دومی معمولا بسیار سریعتر عمل می کند. خصوصا هنگامی که آرگومان های ورودی از جنس کلاس های حجیم باشد.

۱۳. مانند سوال قبل جلوگیری از کپی نشدن و اجرا شدن سریعتر. عوض کردن متغیر ها داخل توابع که می تواند خیلی مفید باشد. همچنین با استفاده از اشاره گر ها می توان آدرس یک متغیر بزرگ را داخل کلاس های یا توابع فرستاد و همه این توابع و کلاس ها به آدرس و در نتیجه متغیر، دسترسی دارند. نمونه این را در SFML دیده بودید. در مورد فونت ها و background.

۱۴. باعث می شود یک برنامه نوشته شده را بتوانیم با آرگومان های متفاوت اجرا کنیم. مثال این را در انتگرال گیری دیده بودیم که یک برنامه برای انتگرال نوشته شده بود و بدون کامپایل کردن برنامه می توانستیم حدود انتگرال گیری را عوض کنیم.

.۱۵

```
int (*p_f1)(int x, int y); //pointer to function
```

کد بالا یک اشاره گر به نام p_f1 به تابع درست می کند که تابع باید خروجی int و دو ورودی از جنس int داشته باشد. این اشاره گر می تواند به تمام توابعی که از این جنس هستند اشاره کند. کاربرد این برا باز هم مثلا در انتگرال گیری دیده بودیم که لازم نبود کد مربوط به انتگرال گیری را عوض کنیم تا تابع انتگرال گیری نوشته شده عوض شود. تنها اشاره گر می توانست به تابع دلخواه کاربر اشاره کند و انتگرال گیری را انجام دهد.

.۱۶

```
int& larger(int& a, int& b)
{
        return a > b ? a : b;
}
```

```
int* smaller(int& a, int& b)
{
        return a > b ? &a : &b;
}
```

توجه: نوشتن تابع average معنی ندارد. چرا که $\frac{a+b}{2}$ همواره rvalue است و نمی تواند به عنوان یک lvalue به کار رود.

۱۷. یعنی اینکه بازه را مدام تقسیم بر دو کنیم تا به جواب برسیم. نمونه کار این الگوریتم در جستجو کردن یک عدد در یک آرایه منظم شده آمده بود. برای پیدا کردن min بین مثلا ۱۶ عدد ابتدا اعداد را دوتایی مقایسه می کنیم. یعنی ۸ تا مقایسه. سپس ۸ عدد حاصله را دو تا دو تا مقایسه می کنیم، می شود ۴ مقایسه. سپس ۲ و ۱ مقایسه.

تعداد مقایسه = (n - 1) $\rightarrow$ 1 + 2 + 4 + 8 = 15

حل ریاضی:

$$۱ + ۲ + ۴ + ... + ۲^n = \frac{۲^{n+۱} - ۱}{۲ - ۱} = ۲^{n+۱} - ۱$$

در اینجا داریم:

$$۱ + ۲ + ۴ + ... + ۲^{log_۲^n - ۱}$$

در نتیجه:

$$= ۲^{log_۲^n} - ۱ = n - ۱$$

توجه شود که n بزرگترین مربع کامل می باشد.

نکته ای که در اینجا وجود دارد این است که از روش تقسیم و حل سودی نبرده ایم. چون از ابتدا با داشتن n عدد می توانستیم با یک حلقه با n - 1 مقایسه کوچکترین را پیدا کنیم.

۱۸. کلاس برای تعریف داده از جنس دلخواه به کار برده می شود. مثلا یک داده از جنس دانشجو، جعبه، انتگرال و ...

properties : اجزاء داخلی کلاس مثل طول، عرض، ارتفاع، شماره دانشجویی، ...

methods : توابع برای کاربرد های متناسب با کلاس مثل محاسبه مساحت، ...

constructor : مقادیر اولیه که کلاس به آنها نیاز دارد. مثل است دانشجو یا اندازه طول جعبه

destructor : محلی برای از بین بردن داده هایی که به صورت دینامیک ایجاد شده اند.

copy constructor : هنگامی که یک شیء از کلاس توسط یک شیء دیگر از همان کلاس بوجود می آید.

۱۹. برنامه نویسی پویا به این معنا است که در طی روند برنامه نویسی هنگامی که به یک عدد یا محاسبه تعداد دفعات زیادی نیاز داریم هر دفعه آن را حساب نکنیم و فقط یک بار محاسبات را انجام دهیم، نتیجه را ذخیره کرده و در دفعات بعدی فقط از آن استفاده کنیم. مثلا این را در fib دیدیم که یک بردار درست کردیم و نتایج fib را درون آن ذخیره کردیم تا مجددا مجبور به حساب کردن نشویم. چون به صورت بازگشتی، همانطور که در کلاس بحث شد، مثلا برای محاسبه fib(5) نیاز داریم fib(4) و fib(3) را حساب کنیم. حال برای محاسبه fib(4) دوباره نیاز داریم fib(3) را حساب کنیم که اگر این را از قبل ذخیره کرده باشیم، صرفه جویی بزرگی در وقت صورت خواهد گرفت.