جلسه دهم(رشته ها در C++)

اصولا رشته ها در زبان های برنامه نویسی به دو دسته تقسیم می شوند.

۱.رشته های ایستا:

این رشته ها مقدار مشخصی از حافظه را اشغال می نمایند و از نظر طول دارای یک مقدار حد اکثری میباشند.عملیات روی این رشته ها با سرعت بیشتری انجام می شود.

۲.رشته های پویا:

این رشته ها بسته به نیاز, طولشان تغییر میکند و مقدار حافظه ای که اشغال میکنند بسته به نیاز تغییر میکند و تمام این عملیات تخصیص حافظه بدون دخالت برنامه نویس انجام میشود.عملیات روی این نوع از رشته ها سرعت کمتری نسبت به نوع قبل دارد.اما کار با آنها بسیار ساده تر است.

رشته های استاندارد زبان Cاز نوع اول هستند در این مبحث به بررسی این نوع رشته می پردازیم نوع دوم رشته های پویا در زبان C++تحت عنوان کلاسstringدر هدر فایل string.hارائه شده که برای درک بیشتر در مباحث شی گرایی مطرح می شود.

رشته های استاندارد زبان C:

این رشته ها دقیقا به صورت آرایه ای از نوع کاراکتر(char) هستند اصولا برای انجام بیشتر عملیات برروی رشته ها باید روشی برای تعیین طول رشته ها داشته باشیم برای این کار دو روش وجود دارد

۱.قرار دادن کاراکترNULLبا کد اسکی صفر در اولین خانه بعد از آخرین کاراکتر رشته که این دقیقاً روش مورد استفاده Cاست

۲.استفاده از خانه شماره صفر آریه مورد استفاده رشته برای ذخیره طول رشته به صورت عدد که در زبان های BasicوPascalاستفاده میشود.

که به نظر من روش اول انعطاف بیشتری دارد هر چند کار کردن با رشته ها را مشکل تر میکند.

روش تعریف رشته ها :

char str۱[۱+حداکثر طول رشته];

این همان تعریف آرایه از نوع کاراکتر است

نکته:

بدلیل استفاده از آخرین خانه برای کاراکتر NULL طول آرایه همیشه باید یک واحد بیشتر از طول رشته باشد.

برای مقدار دهی اولیه به آرایه کارکتر غیر از روش معمولی مشابه سایر آرایه ها روشی مخصوص رشته ها نیز وجود دارد.

روش معمولی char str۱[]={'b','o','o','k','\'};

روش خاص char str۲[]="book";

کار باروش دوم ساده تر است توجه داشته باشید که کاراکتر NULLتعیین کننده انتهای رشته به صورت "برای کاراکتر پنجم در روش اول مقدار دهی شده در حالی که این عمل در روش دوم بوسیله کامپایلر انجام می شود.

نکته:

cpplearning.blogfa.com

۱.در روش اول بجای عبارت " میتوان از عبارت NULLنیز استفاده نمود چرا که ("==NULL)

```
char str۱[]={'b','o','o','k', NULL };
```

۲.مثل تمام آرایه های دیگر بیان طول آرایه در تعریف آرایه به همراه مقدار دهی اولیه اختیاریست اما در صورت بیان نباید تعداد مقادیر اعلام شده بیشتر از طول اعلام شده باشد.

۳.مثل تمام آرایه ها بعد از تعریف آرایه دیگر نمی توان مقدار دهی اولیه استفاده نمود بعنوان مثال دستورات زیر صحیح نمی باشد.

```
char str[۵];
str[]="book";

char str۱[۵];
str۱[]={'b','o','o','k',''};
```

یک نکته در تعریف آرایه:

در تعریف آرایه باید طول آرایه مشخص باشد.لذا تعریف به صورت

```
char str[];
```

خطای کامپایلری دارد.

روش دیگر استفاده از آرایه ها برای رشته ها تعریف آرایه پویا به صورت اشاره گر است که انعطاف زیادی به رشته ها میدهد.

```
char *str۱="book";
```

در این دستور به اشاره گر حافظه تخصیص داده می شود و رشته در حافظه تخصیصی کپی می شود.

رشته هایی که به صورت آرایه کاراکتر تعریف می شوند (رشته هایی که در این مبحث مطرح شد) دو مشکل بزرگ دارند

۱.در صورتی که رشته مورد نظر بزرگتر از فضای کل آرایه باشد مجاز به نوشتن رشته در آرایه نیستیم چراکه دسترسی غیر مجاز به حافظه تخصیص داده نشده صورت میگیرد.

۲.عملگر های پایه مثل + و> و< و==و = روی رشته ها عمل نمی کنند برای این منظور باید از توابع مخصوصی استفاده نماییم که این توابع در هدر فایل string.hتعریف شده اند در ادامه به بررسی تعدادی از توابع پر کاربرد این هدر فایل می پردازیم.

تابع char* strcpy(char* a,char* b)

این تابع کار عملگر انتساب(=)را انجام می دهد به این صورت که تمام کاراکتر های موجود در حافظه تخصیصی bتا رسیدن به کاراکتر NULL در حافظه تخصیصی aکپی می نماید.منظور از حافظه تخصیصی همان حافظه مورد اشاره ی اشاره گر است.
همان طور که می بینید آرگمان های این تابع اشاره گر هستند و ماهیت آدرس حافظه دارند پس باید همیشه یک اشاره گر یا یک عبارت با ما هیت آدرس حافظه به آنها داد (نه فقط این تابع بلکه توابعی که در ادامه معرفی می شوند نیز چنین اند)در مبحث آرایه ها دیدیم که نام آرایه به تنهایی ماهیت اشاره گری دارد و به اولین خانه آرایه اشاره می کند پس در این جانیز نام یک رشته به تنهایی و بدون کروشه ([]) یک اشاره گر است و ماهیت آدرس دارد اما همین که کروشه جلویش قرار گیرد مثل تمام آریه ها ماهیت (نوع) خانه های آرایه را دارد یعنی همان نوع داده ای charپس مثلاً برای بدست آوردن آدرس کاراکتر xام باید بنویسیم &str[x] نتیجه جالب توجه این است که دو عبارت &x[۰]وxدقیقاً هر دو آدرس خانه صفر آرایه هستند

نوع باز گشتی این تابع هم که معمولاً استفاده نمیشود اشاره گریست به آرگمان اول یعنی همان a
در ادامه چند مثال از کاربرد این تابع را می بینیم.

```
char str۱[۵];
strcpy(str۱,"book");
```

عبارت bookدر str۱کپی می شود.

```
char str۲[۵];
strcpy(str۲,str۱);
```

محتویات str۱درstr۲کپی می شود.

```
char str۳[٤];
strcpy(str۳,&str۲[۲]);
```

str۲از کاراکتر ایندکس ۲ تا انتها در str۳کپی می شود.

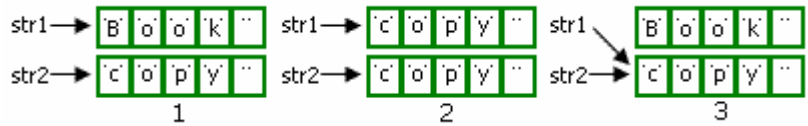
```
strcpy(str۲,&str[۲]);
```

str۲از کاراکتر ایندکس ۲ تا انتها بر روی خودش کپی می شود نتیجه این عمل حذف عبارت bo از ابتدای رشته book است.

```
strcpy(&str۲[۱],&str[۲]);
```

str۲از کاراکتر ایندکس ۲ تا انتها بر روی کاراکتر اندکس ۱ تا انتهای خودش کپی می شود نتیجه این عمل حذف عبارت o ایندکس۱ از عبارت book است.

نکته:
عبارت انتساب به صورت str۱=str۲; تنها زمانی صحیح است که عبارت سمت چپ به صورت اشاره گر تعریف شده باشد.که باوجود صحیح بودن, این عمل با عمل تابع strcpyمتفاوت است. این عمل در واقع محل مورد اشاره ی اشاره گر را عوض میکند و بعد از اجرای این دستور دو اشاره گر مورد نظر به یک رشته اشاره میکنند اما تابع strcpyواقعاً مقادیر درون آرایه ها را در یکدیگر کپی میکند و رشت ها یکی نمی شوند.



در شکل ۱ حالت اولیه رشته های فرضی str۱وstr۲را می بینید شکل دو نتیجه را بعد از اجرا دستورstrcpyاست اما شکل ۳ نتیجه اجرا دستور str۱=str۲; است.

تابع char* strcat(char* a,char* b)

این تابع برای اتصال دو رشته استفاده می شود به این صورت که یک کپی از bبه انتهای متصل می کند.

```
char *str۱="book";
char str۲[۹]="note";
strcat(str۲,str۱);
```

بعد از اجرای این مجموعه دستورات رشته ی str۲حاوی عبارت "notebook"می باشد.
این توابع هیچ کنترلی بر تخصیص حافظه ندارند اگر در خط دوم طول آرایه را ۹ معرفی نمی کردیم حافظه کافی تخصیص داده شده برای اضافه شدن عبات bookبه انتهای رشته نداشتیم و دسترسی غیر مجاز به حافظه تخصیص داده نشد پیش می آمد.

هشدار:
دسترسی غیر مجاز به حافظه تخصیص داده نشده عواقب بد و غیر قابل پیش بینی خواهد داشت مثلا ممکن است محتویات یکی از متغیر ها بوسیله متغیر دیگر رو نویسی شود و منشاء مشکلات متعددی شود.

تابع int strcmp(char* a,char* b)

این تابع دو رشته a,bرا باهم مقایسه می کند که برای مرتب سازی لیست اسامی و... بسیار مفید است
در جدول زیر مقادیر بازگشتی تابع را مشاهده می کنید.

strcmp(a,b)>۰	a>b
strcmp(a,b)==۰	a==b
strcmp(a,b)<۰	a<b

این تابع به حروف کوچک و بزرگ حساس است مثلا فرق Aligaliرا متوجه می شود و یکی را بزرگتر از دیگری تشخیص می دهد. تابع دیگر به نام strcmpiوجود دارد که دقیقاً قالب همین تابع را دارد و به حروف کوچک و بزرگ حساس نیست و مثلاً ali,Aliرا مساوی تشخیص می دهد.
منظور از کوچکی و بزرگی یک رشته طول آن نیست بلکه جایگاه قرار گیری یک رشته در لیست مرتب شده رشته ها می باشد در واقع رشته ها کاراکتر به کاراکتر از سمت چپ ترین کاراکتر به راست ترین کاراکتر از نظر بزرگی کد اسکی باهم مقایسه می شوند و چون جدول کد اسکی براساس حروف مرتب شده اند ارزش رشته بدست می آید در واقع مثل مقایسه اعداد ی در مبنای ۲۵٦ می باشد!!!

تابع int strlen(char* a)

این تابع طول رشته که برابر تعداد کارکتر های آن می باشد را برمیگرداند. طول رشته یعنی تعداد تمام کاراکتر ها از ابتدای رشته تا اولین کاراکتر NULL که خود کاراکتر NULLحساب نمیشود.

تعداد زیادی توابع مفید گوناگون برای کار بارشته ها وجود دارد که می توانید در Helpکامپایلر آنها را بیابید بهتر است به خواندن متون ساده انگلیسی این نرم ازار ها و EBookهای رفرنس عادت کنید چون کاملترین مر جع چیزی جز helpکامپایلر نیست.

توجه:
دستور های چاپ رشته ها رشته هارا از ابتدا تا آخرین کاراکتر قبل از کاراکتر NULLچاپ می کنند.

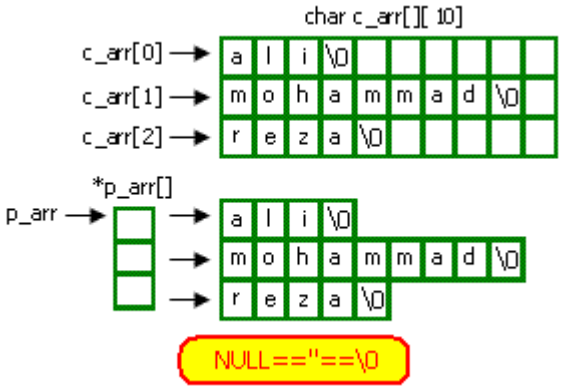
آرایه ای از رشته ها:
فرض کنید می خواهیم لیستی از اسامی داشته باشیم برای ذخیره چنین چیزی ساده ترین ساختار استفاده از آرایه ای از رشته هاست برای پیاده سازی چنین چیزی دو روش در پیش داریم یکی استفاده از آرایه دو بعدی کاراکتری هر سطر برای یک رشته. و روش دوم استفاده از آرایه ای از اشاره گر ها و تخصیص حا فظه به اشاره گر ها روش اول روشی ساده اما خشک است در مقابل روش دوم مشکل است اما انعطاف بیشتری دارد.
تعریف به صورت آرایه دو بعدی

```
char c_arr[][۱۰]={"ali","mohammad","reza"};
```

تعریف به صورت آرایه ای از اشاره گرها

```
char *p_arr[]={"ali","mohammad","reza"};
```

برای در ک تفاوت این دو به شکل دقت نمایید



در روش اول می بینید که آرایه ای دو بعدی از کاراکتر هارا داریم اما در روش دوم آرایه ای از اشاره گر ها را داریم قبلاً گفتیم که نام آرایه اشاره گریست به اولین خانه آرایه در شکل می بینیم که مانند تمام آرایه ها نام آرایه ای از اشاره گر ها نیز خود یک اشاره گر است که به آدرس اولین خانه آرایه ی مذکور اشاره میکند سپس هر یک از خانه ها به محلی از حافظه که رشته مورد نظر قرار دارد اشاره میکنند. اما چرا انقدر لقمه را بپیچانیم!!!

اگر به شکل توجه کنیم متوجه می شویم که در روش دوم مقدار کمتری حافظه از سیستم گرفته ایم دوم اینکه رشته های ما می توانند در محل های گوناگونی از حافظه قرار گیرند از همه مهمتر برای عملیات پیچیده و سنگینی مثل مرتب سازی برای جابجا کردن رشته ها میتوانیم بجای استفاده از دستور strcpyوجابجا کردن واقعی رشته ها با عمل انتساب اشاره گر ها به یکدیگر محل مورد اشاره ی اشاره گر ها را باهم عوض کنیم و این به طور چشمگیری سرعت عملیات را افزایش می دهد.

ورودی خروجی رشته ها:

برای ورودی خروجی رشته ها در زبان C++میتوان از اشیا cinوcoutاستفاده نمود به مثال توجه کنید.

```
char str[۲۱];
cin>>str;
cout<<str;
```

به همین سادگی یک رشته را از صفحه کلید می خوانیم و بعد آن را چاپ می کنیم ام این رو ش چند مشکل دارد اگر کاربر رشته ای وارد کند که یک فاصله در آن وجود داشته باشد فقط عبارت قبل از فاصله خوانده خواهد شد!! مثلاً اگر بنویسیم my bookدر خروجی بجای چاپ همین عبارت فقط عبارت my چاپ می شود!! چرا که این تنها چیزیست که خوانده شده. مشکل دوم این است که اگر کاربر رشته ای بزرگتر از ۲۰ کاراکتر وارد کند رشته خارج از دامنه آرایه نوشته می شود و دسترسی غیر مجاز به حافظه تخصیص داده نشده صورت می گیرد

اما راه حل استفاده از تابع getlineعضو شیء cinاست(خیلی خودتان را در گیر اصطلاحات تابع عضو و... نکنید بعداً در مباحث شی گرایی توضیح می دهم) این تابع با دو نسخه متفاوت وجود دارد

به مثال اول توجه کنید.

```
char str[۲۱];
cin.getline(str,۲۰);
cout<<str;
```

در این مثال تابع مذکور دو آرگمان دارد که اولی اشاره گر یا نام آرایه ی رشته است و دومی حداکثر طول رشته ای که باید خوانده شود و به این صورت مشکل اندازه رشته و آرایه حل شد. این تابع بجای شناسایی کاراکتر فضای خالی بعنوان پایان رشته از کازاکتر انتهای خط استفاده میکند. پس مشکل خوانده نشدن کامل رشته هایی که فضای خالی دارند نیز حل می شود

به مثال دوم توجه کنید

```
char str[۲۱];
cin.getline(str,۲۰,'.');
cout<<str;
```

دراین مثال از نسخه دوم تابع استفاده شده این نسخه یک آرگمان سوم دارد که مشخص کننده کاراکتر انتهای رشته ی وارد شده است که به آن کاراکتر مرز بندی هم میگویند در واقع نسخه اول حالت خاصی از نسخه دوم است که کاراکتر مرزبندی آن کاراکتر انتهای خط است.یک کاربرد این تابع زمانی است که فرض کنیم نقطه انتهای جمله ها باشد در این صورت می توانیم جمله هایی را که کاربر وارد میکند در رشته های متفاوتی بریزیم یا این که مثلاً فقط جمله ی اول را بخوانیم!!

من در این مباحث سعی میکنم فقط از C++ صحبت کنم اما در اینجا اشاره کوچکی هم به توابع ورودی خروجی زبان Cخواهیم داشت.

خواندن رشته ,کاراکتر مرز بندی فضای خالی است

```
scanf("%s",str);
```

خواندن رشته ,کاراکتر مرزبندی انتهای خط است

```
gets(str);
```

چاپ رشته

```
printf("%s",str);
```

چاپ رشته به صورت رنگی با استفاده از رنگ تعیین شده توسط دستورات رنگ بندی

```
cprintf("%s",str);
```

چاپ رشته

```
puts(str);
```

وبسیاری توابع دیگر که همگی در هدر فایل stdio.h هستند.

ارسال رشته ها به توابع بعنوان پارامتر:

رشته ها آرایه هستند پس ارسالشان به توابع نیز مانند آرایه ها است پس به مبحث آرایه ها مراجعه نمایید.

چند مثال کوچک وکلی برای آسنایی با چگونگی کار بارشته ها:

۱.خواندن آرایه ای از رشته ها به صورت آرایه کاراکتری دوبعدی

```
char arr[٥٠][٣١];
for(int i=٠;i<٥٠;i++){
if(!cin.getline(arr[i],٣٠,'/n'))
    break;
}
```

۲. خواندن آرایه ای از رشته ها به صورت آرایه ای ازاشاره گرها

```
char *arr[٥٠];
for(int i=٠;i<٥٠;i++){
arr[i]=new char[٣١];
cin.getline(arr[i],٣٠);
if(arr[i][٠]==NULL)
    break;
}
```

۳.چاپ آرایه ای از رشته ها به صورت آرایه ای از اشاره گرها

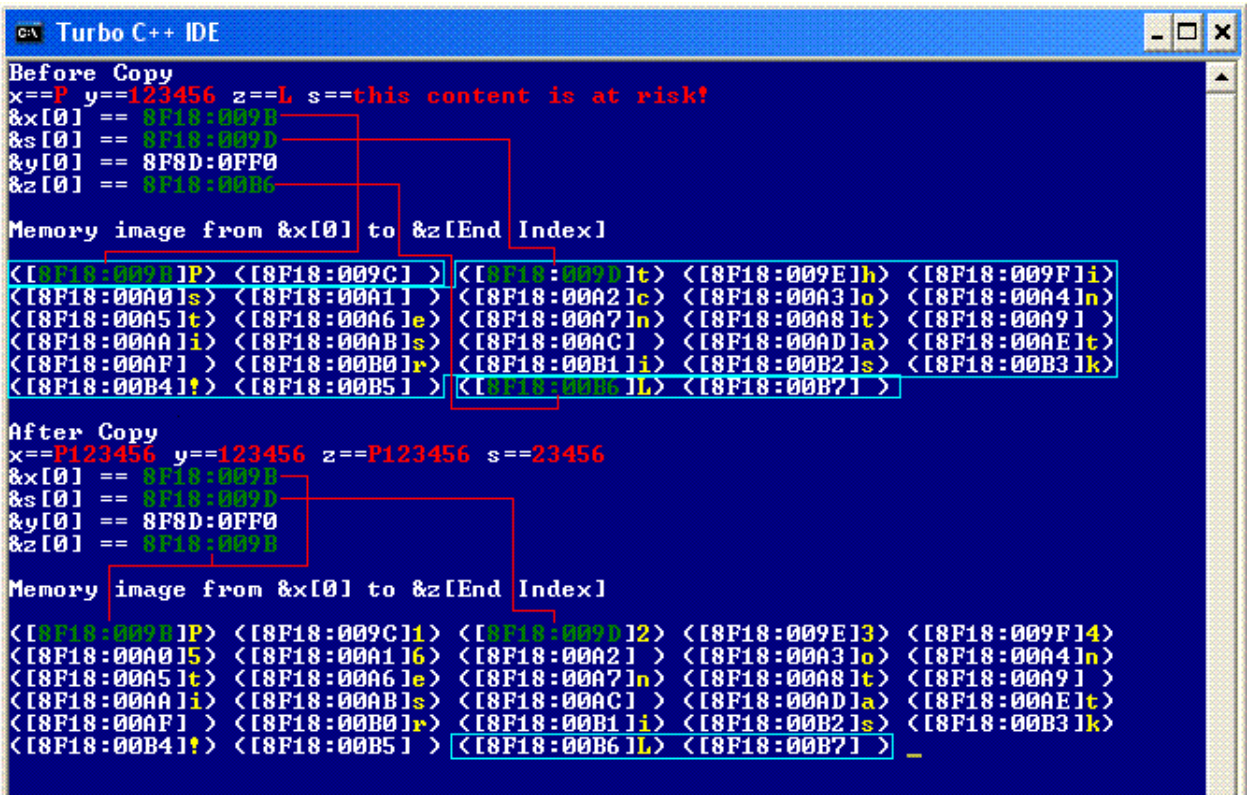
```
char *arr[٥٠];
for(int i=٠;i<٥٠;i++){
cout<<arr[i]<<endl;
}
```

٤.تعویض جای دو رشته در آرایه ای از رشته ها به صورت آرایه ای از اشاره گر ها

```
void swap_str(char*a,char*b){
char *tmp;
tmp=a;
a=b;
b=tmp;
}
int main(){
char *arr[]={"ali","mohammad","reza"};
swap(arr[٠],arr[٢]);
```

}

در ادامه ی این جلسه ی بزرگ یک برنامه نوشتم که می توانید با آن به بررسی چگونگی رفتار توابع رشته ها و.... با حافظه بپر دازید که توجه شمارا به یک مثال از بررسی تابع strcatمی پردازیم.
در شکل رشته ها و دامنه هریک در کادر های آبی رنگ مشخص شده بعد از اجرا دستور رشته ی xبدلیل نداشتن فضای کافی برای نگه داری کل عبارت جدید به دامنه ی رشته sتجاوز نموده و محتویات آنرا تا حدودی رو نویسی نموده در نکته بعد رشته ی z که به رشته ای در حافظه اشاره داشته حالا به محتویات رشته ی x اشاره دارد حالا حافظه ای که در تصویر دوم از حافظه با کادر آبی مشخص شده حا فظه ایست که سلبقاً به zتخصیص داده شده بود اما حالا هیچ اشاره گری به آن اشاره ندارد و البته این حافظه آزاد هم نشده !! و چون دیگر آدرسش در هیچ اشاره گری نیست قابل آزار نمودن هم نیست به چنین حافظه ای حافظه گمشده می گویند که پدیده ای نامطلوب است و باعث مصرف غیر اصولی حافظه می شود.نکته دیگر این که چون متغیر yبه صورت آرایه تعریف شده در حافظه استاتیک قرار دارد ودر محلی متفاوت از حافظه قرار داشته و در تصویر حافظه دیده نمیشود شاید اگر آن هم به صورت اشاره



گر بود محتویات آنهم بوسیله ی رشته x رو نویسی می شد.
سورس برنامه

```
#include <stdio.h>
#include <stdlib.h>
#include <iostream.h>
#include <conio.h>
#include <string.h>
int main(void){
  textmode(٦٤);
  textattr(١٥|١٦);
  clrscr();
```

```
 char *x="P",*s="this content is at risk!",y[]="١٢٣٤٥٦",*z="L";
 printf("Before Copy \n");
 cout<<"x=="<<x<<" y=="<<y<<" z=="<<z<<" s=="<<s;
 printf("\n");
 printf("&x[٠] == %p \n",x);
 printf("&s[٠] == %p \n",s);
 printf("&y[٠] == %p \n",y);
 printf("&z[٠] == %p \n",z);
 printf("\n");
 printf("Memory image from &x[٠] to &z[End Index] \n");
 for(i=٠;i<٢٩;i++){
  if(i٪٥==٠)  printf("\n");
  //textattr(١٥|١٦);
  cprintf("([%p]",&x[i]);
  textattr(١٤|١٦);
  cprintf("%c",x[i]);
  textattr(١٥|١٦);
  cprintf(") ");
 }

 z= strcat(x,y);
 printf("\n\n");
 printf("After Copy \n");
 cout<<"x=="<<x<<" y=="<<y<<" z=="<<z<<" s=="<<s;
 printf("\n");
 printf("&x[٠] == %p \n",x);
 printf("&s[٠] == %p \n",s);
 printf("&y[٠] == %p \n",y);
 printf("&z[٠] == %p \n",z);
 printf("\n");
 printf("Memory image from &x[٠] to &z[End Index] \n");
 for(i=٠;i<٢٩;i++){
  if(i٪٥==٠)  printf("\n");
  cprintf("([%p]",&x[i]);
  textattr(١٤|١٦);
  cprintf("%c",x[i]);
  textattr(١٥|١٦);
  cprintf(") ");
 }
 getch();
 return ٠;
```

جلسه ی آینده جریان های ورودی خروجی را بررسی میکنیم
بعد از آن در جلسات بعد ساختارها و بعد ازآن شیئ گرایی و فایلها و در ادامه ی آن ساختمان های داده را با هم یاد میگیریم
سعی کنید برای یادگیری بهتر به صورت عملی روی کامپیوتر تمرین حل کنید فقط بااین کار و البته مطالعه است که می توان برنامه نویسی شد!! من یه پیش نهاد دارم یه برنامه بنویسید که لیستس ازنام افراد از ورودی بگیره قابلیت جستجو داشته باشه و اطلاعات رو چاپ کند وهمه اینها از طریق یه منوی اصلی برای اجرا قابل انتخاب باشه

به این صورت که کاربر شماره گزینه منو را وارد میکند وبرنامه عمل میکند این به یادگیری خیلی کمک میکنه منم می تونم اگه بخواین جلسه ی بعد حلش رو براتون بنویسم