جلسه هشتم(توابع واشاره گرها)

در این مبحث به روشهای ارسال نتایج توابع به برنامه اصلی با استفاده از اشاره گرها و رفرنس ها می پردازیم.

اشاره

تابه حال برای گرفتن خروجی توابع به کمک دستور returnاز مقدار برگشتی استفاده می نمودیم اما محدودیتش این بودکه با این روش فقط می توانستیم یک مقدار را به برنامه اصلی بر گردانیم

اما را ه حل این مشکل اشاره گر و رفرنس است اجازه بدهید نگاه دقیقی به پارامتر های توابع داشته باشیم

تا به حال برای ارسال مقادیر به توابع مقادیریا نام متغیر محتوی مقادیر را در لیست آرگمان ها هنگام فرا خوانی تابع می نوشتیم با این کار درون حافظه تخصیص داده شده به تابع مورد نظر متغییر هایی ایجاد می شدند که مقادیر ارسالی به توابع درونشان کپی میشد بعد از پایان کار تابع کل حافظه تخصیصی به همراه متغیر های مذکور از بین می رفتند و نتایج تابع که مثلا درون آن متغیر ها بود بدون دستیابی از بین می رفت

خب چاره چیست می خواهیم بوسیله اشاره گر ها کاری کنیم که متغییر های مذکور خارج از حافظه تخصیصی به تابع باشند تا بعد از پایان کار تابع از بین نروند اما خارج از حافظه تخصیصی تابع یعنی کجا دقیقا!!؟ خب منظو همان حافظه ی تخصیصی تابعی که تابع مد نظر رو فرا خوانی می کنه.

شرح

تفاوت آرگمان ها و پارامترها:

آرگمان ها متغیر های برنامه اصلی هستند که هنگام فرا خوانی تابع درون لیست آرگمان ها قرار میگیرندوبه تابع فرستاده می شوند پارامترها متغیر های محلی تابع هستند که در لیست پارامتر های تابع هستند.در مثال بعد a,bآرگمان تابع و x,yپارامترهای تابع.

قبلاً گفتیم که توابع با استفاده از نوع بازگشتی یک و فقط یک مقدار را بر میگردانند و هر گونه تغییر در متغیرهای پارامتر ها ی توابع هیچ انعکاسی در برنامه اصلی(آرگمان ها) ندارد.

زیرا c++تنها کاری که انجام میدهد کپی کردن آرگمان ها در پارامتر هاست.

به مثال توجه کنید.

در این مثال تابعی به نام swapوجود دارد که باید دو متغییر را بگیرد و جای محتویات آنهارا عوض کند.

```
#include<iostream.h>
#include<conio.h>
void swap(int ,int );
int main(){
int a=١٠,b=٣٢٤;
swap(a,b);
cout<<a<<" "<<b;
getch();
return ٠;
}
void swap(int x,int y){
int temp=x;
x=y;
y=temp;
}
```

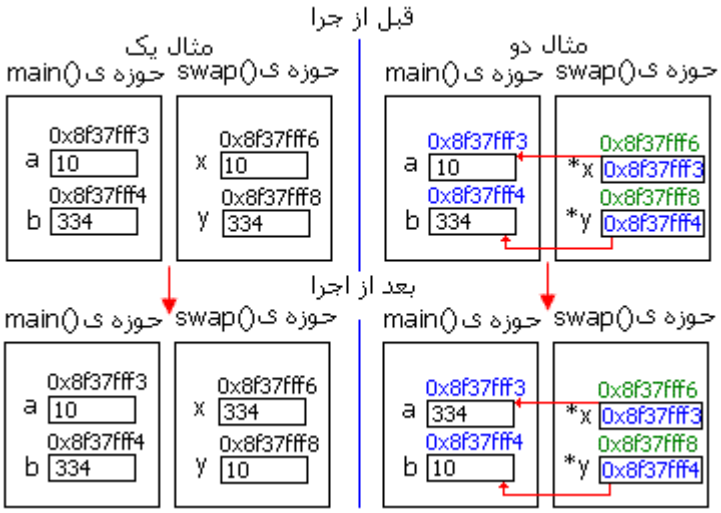
cpplearning.blogfa.com

اول از همه این که تابع swap باید دو مقدار بر گداند اما چون با نوع باز گشتی نمیشه این کار رو انجام داد بیخیال نوع بازگشتی میشیم و خیال همه رو راحت می کنیم و voidتعریفش میکنیم .
حالا برنامه رو اجرا میکنیم در این برنامه دو متغییر a,bتعریف می شن و مقادیری رو ذخیره میکنن بعد تابع رو فرا خوانی میکنیم و از متغیرهای مذکور بعنوان ارگمان تابع استفاده می کنیم حالا اتفاق مهمی رخ میده اون این که متغیر های x,yکه متغیرهای محلی برای تابع swapمحسوب می شوند ایجاد میشوند وبعد با مقادیر a,bمقدار اولیه میگیرن خب پس مشخص شد که متغیرهای x,yمجزا از a,bهستند و هیچ ربطی به هم ندارن.بعد از فراخوانی تابع مقادیر درون x,yباهم عوض میشن اما چون هیچ ربطی به a,bندارن a,bهیچ تغییری نخواهند داشت.بعد از خاتمه اجرا تابع a,bچاپ میشن و می بینیم که هیچ تغییری نداشتن .

خب مشکل اینه که ما بجای جابه جا کردن محتویات a,bاین کار رو به x,yانجام دادیم اما اگر بتونیم با استفاده از اشاره گرها به محتویات a,bدسترسی داشته باشیم می تونیم به راحتی به جای جابجا کردن محتویات x,yمحتویات a,bرا جابجاکنیم.
حالا مثال قبل را با اشاره گر ها باز نویسی میکنیم.

```
#include<iostream.h>
#include<conio.h>
void swap(int* ,int* );
int main(){
int a=١٠,b=٣٣٤;
swap(&a,&b);
cout<<a<<" "<<b;
getch();
return ٠;
}
void swap(int* x,int* y){
int temp=*x;
*x=*y;
*y=temp;
}
```

در این مثال آرگمان های تابع از نوع اشاره گر عدد صحیح تعریف شده با فرا خوانی تابع آدرسa,bرو بوسیله عملگر &بدست آوردیم بعد از به وجود آمدن اشاره گر های x,yبا این آدرس ها مقدار دهی میشوند حالا تابع با استفاده از این اشاره گر ها به a,bدسترسی داره وبا عملگر*محتویاتشون رو جابجا میکنه.
برای درک بهتر به شکل توجه کنید.



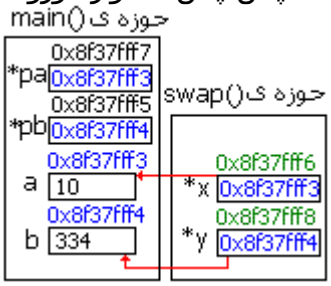
در تصویر مثال یک دیده می شود که متغیر ها کاملاً مستقل عمل می کنن در تصویر مثال دو x,yاشاره گر هایی به a,bهستند که به وسیله ی این اشاره گر ها محتویات a,bدستیابی شده و عوض میشوند.
به آدرس های سبز رنگ توجه کنید اشاره گر ها هم مثل همه متغیر ها آدرس

دارند که البته این ربطی به موضوع نداره فقط گفتم بدونید.

توجه کنید که در فراخوانی تابع swapنیاز به دو پارامتر داریم که آدرس حافظه باشن اگه بخواهیم متغیر معمولی برای این منظور استفاده کنیم باید با عملگر& آدرسشونو بدست بیاریم اما اگه مثلا یک اشاره گر به متغیر مد نظرمون داشته باشیم مستقیماً ازش استفاده میکنیم چون آدرس متغیر مد نظرمون در اشاره گر هست و خب دو بار دو بار که آدرس بدست نمی یارن.

```
int pa=&a,pb=&b;
swap(pa,pb);
```

چون pa,pb اشاره گر هستن وخودشون ماهیت آدرس حافظه دارن نیازی به & نیست. به تصویر توجه کنید.(توجه کنید که آرگمان های این تابع از نوع اشاره گر تعریف شده اند پس پس همواره ورودی این تابع اشاره گر یا آدرس حافظه است)

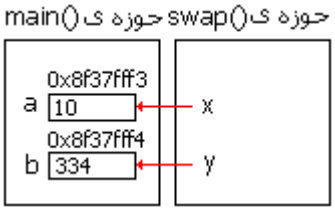


حالا می خواهیم همین مثال رو با رفرنس ها باز نویسی کنیم.

```
#include<iostream.h>
#include<conio.h>
void swap(int& ,int& );
int main(){
int a=١٠,b=٣٣٤;
swap(a,b);
cout<<a<<" "<<b;
getch();
return ٠;
}
void swap(int& x,int& y){
int temp=x;
x=y;
y=temp;
}
```

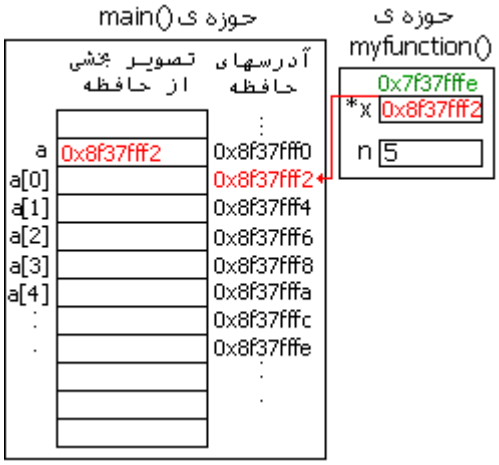
قبلاً گفتیم که رفرنس ها نام دوم برای دیگر متغیر ها محسوب میشوند و از خودشون حافظه ندارن واز حا فظه ی دیگر متغییر ها استفاده میکنن همچنین رفرنس ها در لحظه به وجود آمدن مقدار اولیه ای می گیرند که در واقع این مقدار اولیه آدرس متغیر دیگری است (در لحظه اول و فقط لحظه اول به وجود آمدن مثل اشاره گر ها رفتار میکنند و فقط آدرس حافظه می گیرند و نه چیز دیگری البته این تصور من هست برای فهم بهتر موضوع)و از آن پس رفرنس نام دوم همان متغیر میشود.
با این توضیحات اگه بتونیم پارامترهای تابع رو رفرنسی به متغیر های برنامه تعریف کنیم نام دوم متغیر های برنامه محسوب می شوند و تغیر پارامتر ها همان تغییر متغییر های برنامه اصلی است.
با فرا خوانی تابع رفرنس های x,yایجاد شده وبا متغیر های a,bمقدار دهی اولیه می شوند و به عبارتی نام دوم a,bمی شوند.

در شکل می بینید که x,yخودشان حافظه ندارند و از حافظه a,bاستفاده می کنند.



آرایه بعنوان آرگمان تابع:

هنگام ارسال آرایه به تابع,تابع باید چند مورد رابداند.
۱.آدرس اولین خانه آرایه.
۲.ابعاد و اندازه آرایه.
۳.نوع آرایه.
به این نکته توجه کنید که تابع فقط موارد فوق را میداند.ونمیداند که در کدام خانه آرایه چه مقداری نوشته شده.
در واقع هنگام فراخوانی تابع فقط یک کپی از اشاره گر آرایه(همان نام آرایه) به تابع فرستاده می شود و لذا ارسال آرایه به توابع به صورت اشاره گر است و تغییر در آرایه به برنامه اصلی هم منتقل می شود.در واقع شرایط دستیابی تابع به عناصر آرایه مو جود فراهم میشود و در حقیقت تک تک عنا صر آرایه به صورت یک آرایه ی محلی برای تابع کپی نمی شوند.این مو ضوع را در شکل می توان دید.(با دقت خیلی زیاد به آدرس های حا فظه و اعداد ارقام نگاه کنید)



نوشتار آرگمان های توابع برای آرایه ی ها.

```
type myfunction(int a[],int n);
type myfunction(int a[][const] ,int n);
type myfunction(int a[][const][const],int n);
type myfunction(int a[][const][const]...[const],int n);
```

در این روش بعد اول آرایه نامعلوم وقابل تغییر است که به وسیله ی آرگمان دوم به تابع فرستاده میشود.روش دیگری هم وجود دارد که تمام ابعاد آرایه ثابت باشند. می توان گفت بیشتر برنامه نویسان حالت اول را بیشتر دوست دارند.

خب این پست دیگه زیادی زیاد شده.
جلسه ی بعد به چپ مقدار ها و راست مقدار ها و نوع بازگشتی تابع به صورت رفرنس می پردازیم.