**به نام خدا**

# آموزش Pointer در چند دقیقه

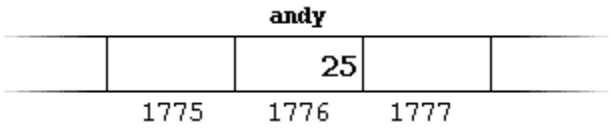
**تهیه و تنظیم : امیرحسین آزمون – عرفان علی محمدی**

## در زبان C/C++ می توان متغیر ها را به دو دسته تقسیم کرد

۱) متغییر های معمولی (که باید بلد باشید اگه نیستید امتحان کامپیوتر را فراموش کنید و یه امتحان دیگه بخونید)

۲) متغییر های Pointerیی

هر متغییر معمولی دارای یک شماره خانه در RAM می باشد به شکل زیر توجه کنید :




در این شکل متغییری به اسم andy وجود دارد که مقدار آن ۲۵ است و در شماره ی خانه ی ۱۷۷۶ RAM قرار گرفته است. در مثال های کد andy همان a است . :دی

```
int main()
{
	int a = 25;
	cout << &a << endl;
}
```

این برنامه شماره ۱۷۷۶ را چاپ می کند پس اگر متغییر معمولی وجود داشت که می خواستیم شماره خانه آنرا بفهمیم از دستور&a استفاده می کنیم.

نکته : &a مانند تابع شماره خانه ای که در رم است را بر میگرداند و نمی توان آن را برابر با مقداری قرار داد و به اصطلاح read-only می باشد. یعنی :

```
&a = 1777;
```

غلط است.

البته عددی که چاپ می کند در مبنای ۱۶ می باشد یعنی اول عدد با 0x شروع می شود که زیاد چیز وحشتناکی نیست برای مثال

```
int a = 10; (برابر با عدد ۱۰ در مبنای ۱۰)
int a = 010; (برابر با عدد ۱۰ در مبنای ۸ و عدد ۸ در مبنای ۱۰)
```

int a = 0x10; (برابر با عدد ۱۰ در مبنای ۱۶ و عدد ۱۶ در مبنای ۱۰)

کلا هر چه با 0x شروع شود مبنای ۱۶ و هرچه با 0 مبنای ۸ است.

و حالا متغییر های نوع دوم که همان متغییر های Pointerیی می باشند

طریقه تعریف:

```
int * p;
```

وقتی یک متغییر Pointerیی تعریف می شود خانه ای از رم را میگیرد مانند یک متغییر معمولی اما تفاوت آنها در این است که می توان خانه ی متغییر Pointerیی را تغییر داد و حتی شماره خانه آن را برابر با شماره ی خانه یک متغییر معمولی کرد اما نوع استفاده آن کمی متفاوت است.

شماره ی خانه ی یک متغییر Pointerیی را اینگونه چاپ می کنیم.

```
int main()
{
	int * p;
	cout << p << endl;
}
```

اگر دقت کرده باشید متوجه شدید که وقتی متغییر را صدا میزنیم شماره خانه ی آن را برمیگرداند نه مقدار آن.

به کد زیر توجه کنید:

```
p = &a;
```

این کد درست است زیرا شماره ی Pointer مورد نظر با شماره ی یک متغییر معمولی برابر شده و در نتیجه مقدار آنها هم برابر می شود پس شرط if پایین true است (درست است):

```
int a = 10;
int * p = &a;
if(*p = a)
```

اگر به کد بالا توجه کرده باشید متوجه شدید که برای فهمیدن مقدار یک Pointer از * و بعد اسم Pointer ( *Pointer_Name )

**به جدول زیر توجه کنید:**

| متغییر های Pointerی | متغییر های معمولی | |
|---|---|---|
| *p | a | **مقدار متغییر** |
| p | &a | **شماره خانه متغییر در** RAM |

**نکته : اگر متغییر P از نوع متغییر های نوع دوم باشد (متغییر های Pointerی) P = 0xbb12a223 غلط است اما وقتی P را تعریف می کنید P به صورت خودکار مانند متغییر های معمولی برابر با شماره خانه ای تصادفی می شود**

**نکته: تمامی موارد زیر درست است و مواردی که بر زیر آن خط کشیده شده غلط است :**

int a = 10, b = 21;

int * p = **مثال های درست**

p = &a;  p = &a ± 2;  p = &a ± b;  p = &a ± b/2;

p = &a±&b;  p = &a/2;  p = &a*3;  p = 0x193bad121;  و مانند اینها

**به برنامه زیر توجه کنید: (در کد پایین هرچه بین "" بعد از // باشد به معنای چاپ شدن در console است)**

```
#include <iostream>
using namespace std;
int main()
{
	int a = 10;
	int * p = &a;
	cout << *p << endl; //*p = a => “10”
	*p = 11; (*p)++;
	cout << a << endl;//a = *p = 12 => “12”
}
```

**در کد بالا چون شماره خانه a با شماره خانه p برابر است پس تغییرات اعمال شده روی مقدار p روی a هم اعمال می شود و برعکس یعنی درکل متغییر p همان a است زیرا هر دو یک خانه اند.**

**به مثال زیر هم توجه کنید:**

```
int main()
{
	int a = 12, b = 20;
	int * p = &a;
	cout << *p << endl; //*p = a => "12"
	p = &b;
	(*p)++;
	cout << b << endl;//b = *p = 21
}
```

**اگر به مثالهای قبل توجه کرده باشید باید متوجه این شده باشید که اگر بخواهیم مقدار یک Pointerرا زیاد کنیم باید به شکل زیر نوشت**

**(*p)++;**

**و نوشتن به صورت زیر غلط است:**

***p++;**

## استفاده از Pointerدر آرایه ها

**استفاده از Pointer در آرایه تفاوت خاصی ندارد اما چند تفاوت کوچک دارد مثال زیر را ببینید:**

```
int main()
{
	int a[10] = {0};
	int * p = a - 1;//نکته ۱
	for(int i = 0; i < 10 ; i++)
```

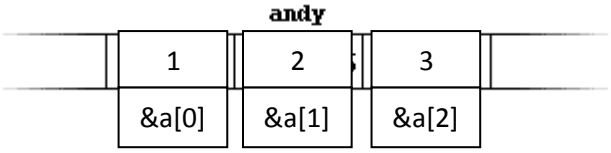
```
	{
		p++;//نکته ۲
		(*p) = i+1;
		cout << a[i] << endl;
	}
}
```

**این مثال اعداد ۱ تا ۱۰ را در آرایه a می ریزد و همزمان با آن چاپ می کند.**

**نکته ۱ : معنی خط بالا این است :**

```
int * p = a - 1;    =>  int * p = &a[0] - 1;
```

**وقتی که با آرایه کار می کنیم می توانیم خط بالا را نوشت یعنی &a[0] را نوشت a . aهمان شماره خانه ای است در رم که اولین خانه آرایه در آن است و خانه های بعدی به ترتیب پشت سرهم است. به شکل زیر توجه کنید :**




**نکته ۲ : در حالت خط نکته ۱ ما از شروع آرایه یک دانه کم کردیم زیرا وقتی به این خط برسد p همان a می شود یک سوال در اینجا پیش می آید که چرا خط نکته ۲ را در آخر حلقه ننوشتیم دلیل آن این است که p بعد از حلقه خانه ای نا معلوم می شود**

**دستور delete را ته برنامه ها بگذارید تا حافظه استفاده شده Pointer را برگرداند.**

```
delete p;
delete[] p;///age arraye bud
```

**درس به پایان رسیده مثال هایی در ادامه آمده که آنها را بخوانید اما هنوز مفهموم هایی از Pointer هست که نگفتیم زیرا در امتحان نمی آید. مفهوم هایی از قبیل Pointerدر Pointer و .....**

```
#include <iostream>

using namespace std;
```

```
int main()
}
	int b = 10;
	int* a;
	a = &b;
	cout << *a << endl; //10
	(*a)++;
	cout << *a << endl; //11
	cout << b << endl;  //11
}
```

```
#include <iostream>

using namespace std;

int main()
{
	int* a = new int[5];//Tarif kardane arraye e 5tayi ba khane haye 0
	a[0] = 10;
	cout << a[0] << " " << a[1] << " " << a[2] << " " << a[3] << " " << a[4] << endl;
	delete[] a;//Hazfe arraye pointer az hafeze
}
```

**اگر سوالی هست ما هستیم:**

erfaniaa@yahoo.com

hajazita@yahoo.com