به نام یزدان پاک

موضوعات : توابع ، متغیر ها ، آرایه و اشاره گر

مدرس : استاد علیاری

توابع (Functions) در C++

هربرنامه از بخشهای کوچکتری به نام تابع تشکیل میشود.هر تابع ، کار واحدی انجام میدهد .
هر تابع باید یک نام اختصاصی داشته باشد که بتوان آن را فراخوانی کرد.
چگونگی تعریف یک تابع را باهم مرور میکنیم :

```
type name(parameter1, parameter2, ...)
{
     statements
}
```

type	نوع داده ای هر تابع عبارت است از نوعی که بازگردانده خواهد شد.
name	نام تابع را تعیین میکند .
parameter	پارامتر های ورودی تابع را مشخص میکند
statement	دستورات اجرایی تابع است.

۱. توابع هم میتوانند پارامتر دریافت کنند و هم پارامتر ورودی نداشته باشند.
۲. توابع هم میتوانند نوع خاصی از داده هارا بازگردانند و هم میتوانند فقط عملی خاص را انجام داده و عبارت بازگشتی نداشته باشند.

ما تنها دو حالت را باهم مرور میکنیم که اولی داده بازگشتی دارد و دومی بدون عبارت بازگشتی ست.

مثال یکم :

```
1   // function example
2   #include <iostream>
3   using namespace std;
4
5   int addition (int a, int b)
6   {
7     int r;
8     r=a+b;
9     return r;
10  }
11
12  int main ()
13  {
14    int z;
15    z = addition (5,3);
16    cout << "The result is " << z;
17  }
```

```
The result is 8
```

توضیح برنامه :

- این برنامه به دو بخش main و addition تقسیم شده است.البته هنوز هم طبق قوانین برنامه با اجرای main اجرا میگردد و تابع addition به طور غیر مستقیم و از طریق فراخوانی در تابع main به اجرا در می آید.
- در زمان فراخوانی هر تابع و در صورتی که پارامتر ورودی داشته باشد باید به همان تعداد ، پارامتر به آن ارسال کنیم.
- نوع داده ای پارامتر ارسالی باید با نوع داده ای پارامتر های تابع یکسان باشد .
- ترتیب در ارسال پارامتر ها مهم است.
- متغیر های a و b به عنوان متغیر محلی در تابع شناخته میشود.
- در خطوط بعدی که عمل جمع کردن و انتساب به متغیرr انجام میگردد.
- همانطور که میبینید نوع داده ای باز گردانده شده با نوع خود تابع یکی ست.
- در خطوط بعدی مقدار بازگردانده شده به متغیر z انتساب میابد.و در آخر نیز در خروجی چاپ میشود.

```
int addition (int a, int b)
                ↑      ↑
z = addition (  5   ,   3  );
```

```
int addition (int a, int b)
  ↓
z = addition (  5   ,   3  );
```

مثال دوم :

نوع داده ای که در توابع بدون مقدار بازگشتی ، به جای type ذکر میگردد ، عبارت void است.
void به معنای پوچ و خالی ست.

دراینجا این عبارت نشانگر این است که این تابع مقداری به عنوان خروجی یا بازگرداننده ندارد..

```
1  // void function example
2  #include <iostream>
3  using namespace std;
4
5  void printmessage ()
6  {
7    cout << "I'm a function!";
8  }
9
10 int main ()
11 {
12   printmessage ();
13 }
```

```
I'm a function!
```

لازم به ذکر است void را میتوان به عنوان پارامتر نیز معرفی کرد.
در زبان C++ نوشتن این عبارت اختیاری ست ولی در زبان C الزامی ست.

```
void printmessage(void)
{
	cout << "I'm a function!";
}
```

❖ اعلام تابع

در C++ ب طور کلی هر متغیر ابتدا باید تعریف گردد و بعد مورد استفاده قرار گیرد.به عنوان مثال ابتدا متغیر x تعریف میشود و بعد مورد استفاده قرار میگیرد.

```
int x;
```

همین مطلب در مورد توابع نیز صدق میکند.توابع را قبل از معرفی نمیتوان فراخوانی کرد.

- اگر تابعی بعد از main تعریف شود کامپایل نمیشود مگر در حالتی خاص.

همیشه کامپایلر C++ در ابتدا توابع را در حد جزئیات کوتاه (نوع بازگشتی و نوع و تعداد پارامتر) بررسی میکند.و در زمان اجرا به بررسی کامل میپردازد.
حال ما برای اینکه توابعمان را بعد از main تعریف کنیم ، جزئیات تابع مورد نظرمان را قبل از main ذکر میکنیم و به نوعی فقط اعلام میکنیم که چینین تابعی با نام مشخص و ویژگی های مشخص وجود دارد.

```
type name(parameters…);
```

- در این اعلام حتی نام پارامتر ها نیز اهمیتی ندارد و ذکر آن اختیاری ست.
- نکته مهم و حائز اهمیت این است که بعد از دستور اعلام اولیه ی تابع ، باید علامت سمی کولن(;) گذاشته شود.

مثالی را بررسی میکنیم :

```
1   // declaring functions prototypes
2   #include <iostream>
3   using namespace std;
4
5   void odd (int x);
6   void even (int);
7
8   int main()
9   {
10    int i;
11    do {
12      cout << "Please, enter number (0 to
13  exit): ";
14      cin >> i;
15      odd (i);
16    } while (i!=0);
17    return 0;
18  }
19
20  void odd (int x)
21  {
22    if ((x%2)!=0) cout << "It is odd.\n";
23    else even (x);
24  }
25
26  void even (int x)
27  {
28    if ((x%2)==0) cout << "It is even.\n";
29    else odd (x);
    }
```

```
Please, enter number (0 to
exit): 9
It is odd.
Please, enter number (0 to
exit): 6
It is even.
Please, enter number (0 to
exit): 1030
It is even.
Please, enter number (0 to
exit): 0
It is even.
```

در محل مشخص شده میبینید که در اعلام یکی از توابع نام پارامترها وارد نشده.اما هر دو تابع به درستی فراخوانی میشوند.پس نتیجه میشود که کامپایلر در ابتدا توجهی به نام پارامتر ها ندارد و وارد کردن آنها اختیاری ست.

توضیحات:

- همانطور که میبینید تابع odd و even بعد از main تعریف شده اند.اما به علت اعلام اولیه ، و شناسایی کامپایلر ، در main استفاده شده اند.
- روند کار بدین صورت است که حلقه do/while تازمانیکه کلید صفر فشرده نشده ، دستورات را اجرا میکند.تابع <<cin که متغیر i را مقدار دهی میکند و در مرحله بعداین مقدار به تابع odd که فرد بودن عدد را بررسی میکند ارسال میشود که درصورت فرد بودن پیغام مربوطه چاپ میگردد و در غیر اینصورت نیز تابع even را فراخوانی میکنید.

❖ مقادیر پیشفرض در پارامترها

در C++ میتوان ورود برخی پارامتر هارا اختیاری اعلام نمود.به عنوان مثال تابعی که دارای سه پارامتر است را با وارد کردن دوپارامتر فراخوانی کرد.

مثالی با هم ببینیم:

```
1   // default values in functions
2   #include <iostream>
3   using namespace std;
4
5   int divide (int a, int b=2)
6   {
7     int r;
8     r=a/b;
9     return (r);
10  }
11
12  int main ()
13  {
14    cout << divide (12) << '\n';
15    cout << divide (20,4) << '\n';
16    return 0;
17  }
```

```
6
5
```

همانطور که میبینید تابع divide دوبار در main فراخوانی شده.در اولی :

```
divide (12)
```

در این تابع تنها یک مقدار ارسال میگردد.در صورتی که divide دو پارامتر دارد.اما پارامتر دومی دارای مقدار پیشفرض (b=2) است.پس نتیجه برابر ۶ میشود.

اما در فراخوانی دوم هردو مقدار وارد شده که در این صورت 20/4 برابر ۵ میشود.

❖ توابع بازگشت (Recursivity) :

یک خاصیت برای توابع است که توابع را قادر میسازد که خود را فراخوانی کنند.

این ویژگی در بعضی عملیات مانند مرتب کردن و یا محاسبه فاکتوریل کاربرد دارد.

```
1   // factorial calculator
2   #include <iostream>
3   using namespace std;
4
5   long factorial (long a)
6   {
7     if (a > 1)
8      return (a * factorial (a-1));
9     else
10     return 1;
11  }
12
13  int main ()
14  {
15    long number = 9;
16    cout << number << "! = " << factorial (number);
17    return 0;
18  }
```

```
9! = 362880
```

میبینید که تابع factorial مانند یک حلقه تکرار شونده while عمل کرده و تا زمانیکه شرط a>1 برقرار است ، تابع factorial اجرا میگردد.

انواع متغیر در C++ از نظر سطح دسترسی

همانطور که قبلا در تعریف متغیر داشتیم به خانه هایی از حافظه کامپیوتری که میتونند اطلاعاتی را درخود نگهداری کنند متغیر گفته میشود.

متغیر دارای ویژگی هایی ست :

- نام : نامگذاری متغیر در هر زبان برنامه نویسی قوانین خاص خود را دارد.

بر فرض مثال در C++ نام متغیر نباید با اعداد شروع شود و در نامگذاری محدودیتی در طول نام وجود ندارد.همچنین C++ به بزرگی و کوچکی حروف لاتین حساس است(CaseSensitve) و نباید از کلمات کلیدی باشد.

- اندازه : هر متغیر بسته به نوعی که دارد اندازه معینی از حافظه را اشغال میکند.به جدول توجه نمایید :

Name	Description	Size*	Range*
char	کاراکتر یا اعداد صحیح کوتاه	1byte	signed: -128 to 127 unsigned: 0 to 255
short int (short)	اعداد صحیح کوتاه	2bytes	signed: -32768 to 32767 unsigned: 0 to 65535
int	اعداد صحیح	4bytes	signed: -2147483648 to 2147483647 unsigned: 0 to 4294967295
bool	درست یا غلط	1byte	true or false
float	اعداد اعشاری کوتاه.	4bytes	+/- 3.4e +/- 38 (~7 digits)
double	اعداد اعشاری بلند	8bytes	+/- 1.7e +/- 308 (~15 digits)

توابع از نظر سطح دسترسی به دو دسته تقسیم میشوند:

۱. متغیر های محلی Local
۲. متغیر های سراسری Global

این دو نوع از نظر ماهیتی هیچ تفاوتی با هم ندارند به جز در سطح دسترسی :

متغیر های محلی (Local) :

۱. این داده ها در داخل یک تابع پنهان هستند.
۲. تنها در همان تابع قابل دسترسی و تغییرند.

متغیر های سرسری (Global):

۱. در همه توابع قابل دسترسی هستند.
۲. در خارج از تابع خاص تعریف میشنود.

```
1   #include<iostream>
2   #include<conio.h>
3
4   using namespace std;
5   // Global Variable
6   int iGlobal = 5;
7   int Rusult();
8   int main()
9   {
10          //Loval Variable
11          int iMain = 10;
12          cout << "Global :" << iGlobal;
13          cout << endl << "Local :" << iMain;;
14          _getch();
15  }
16  int Result()
17  {
18          //      Local Variables]
19          int iResult = 2;
20          return iResult;
21  }
22
23
```

```
Global :5
Local :10
```

آموزش آرایه (Array) در C++

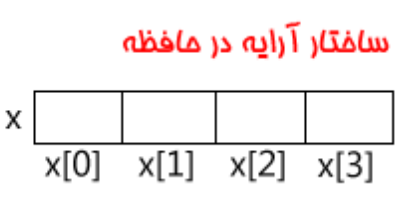
پیش از این ما داده ها را در درون متغیرها و یا ثوابت ذخیره می کردیم. اما گاهی لازم است تعدادی از داده ها را بگونه ای ذخیره نماییم که متعلق به یک مجموعه بوده و دسترسی به آنها آسان باشد.

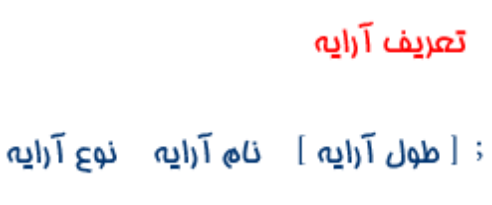
گاهی وقتها لازم است که یک سری از داده هم نوع را دریافت کرده و بروی آنها عملیات خاصی را مثل مقایسه یا مرتب کردن و ... را انجام دهیم، لذا باید آنها را بگونه ای خاص در کنار یکدیگر قرار دهیم که این مسئله در برنامه نویسی را با استفاده از آرایه ها حل خواهیم کرد .

❖ آرایه چیست؟

آرایه مجموعه ای از عناصر هم نوع است. در برنامه نویسی C++ برای تعریف آرایه باید نوع عناصر آنرا مشخص کنیم و آرایه باید حتما دارای نام باشد که از قانون نامگذاری برای متغیرها تبعیت می کند و بعد از نام از [] استفاده می کنیم که درون آن می توان از یک عدد صحیح برای تعیین طول آرایه استفاده نمود .

با هم ساختار یک آرایه را مرور کنیم:





(برای دسترسی به عناصر آرایه از اندیس استفاده می کنیم که اولین اندیس هر آرایه ای از عدد صفر شروع می شود.)

```
type name[Array size]={value1,value2,...};
```

type	نوع آرایه مشخص شده
name	نام آرایه مورد نظر
Array size	اندازه آرایه
value	مقدار های آرایه (عضو های ارایه)

```
int array[4]={2,485,15,58}
```

کد برنامه نویسی C++ بالا آرایه ای را با نام array تعریف می کند که عناصر آن از نوع عدد صحیح هستند و تعداد عناصر آرایه (طول) برابر با ۴ است.

به این نکته توجه کنید که تعداد عناصر آرایه بالا ۴ است ولی اندیس عناصر آن از صفر شروع شده و به سه ختم می شود و نباید تعداد را با اندیس اشتباه گرفت.

گفتنی است که عناصر آرایه پشت سر هم در خانه های حافظه ذخیره می شوند و هر عنصر (خانه) از آرایه به اندازه طول نوع آرایه فضا اشغال می کند. در آرایه بالا چون نوع آرایه تعریف شده int است پس هر عنصر مقدار ۴ بایت و چون طول آن ۴ است در نهایت ۱۶ بایت پشت سر هم از حافظه را اشغال می کند.

معمولا آرایه ها با توجه به ابعادشان تقسیم بندی می شوند:

آرایه های یک بعدی:

آرایه های یک بعدی دارای یک سطر و چند ستون و یا دارای یک ستون و چند سطر هستند و فقط دارای یک اندیس برای دسترسی به عناصرشان می باشند.

مقدار دادن به عناصر آرایه ها:

برای مقدار دادن به آرایه ها هم می توان به تمامی عناصر بصورت یکجا مقدار داد که حالت مجموعه در ریاضی را بخود می گیرد و هم بصورت تک به تک. وقتی طول آرایه را مشخص نمی کنیم با مقدار دادن به عناصر، طول آرایه نیز مشخص خواهد شد.

(نکته) دوستان دقت داشته باشند هنگام تعریف آرایه باید طول را قید کرد مگر اینکه همانجا بدون ذکر طول، به یکایک عناصر مقدار دهیم. پس یا باید طول آرایه را مشخص کنیم یا با مقدار دادن به عنصرهای آرایه،طول برای کامپایلر مشخص شود و اگر غیر از این باشد کامپایلر از برنامه خطا خواهد گرفت .

```
int  x[5] = { 10, 37, -3, 8023, 0 };

////////////// یا \\\\\\\\\\\\

int  x[5];

x[0] = 10;
x[1] = 37;
x[2] = -3;
x[3] = 8023;
x[4] = 0;

////////////// یا \\\\\\\\\\\\

int  x[] = { 10, 37, -3, 8023, 0 };
```

- نکته) همیشه آخرین اندیس آرایه، یک واحد از طول آن کمتر است و تنها دلیل آن شروع شدن اولین اندیس از صفر برای آرایه می باشد.
- برای دسترسی به عناصر آرایه کافی است اندیس آن عنصر از آرایه را در درون [] قید نماییم:

مثال) برنامه ای به زبان C++ بنویسید که تعداد ۵ عدد را از کاربر دریافت کرده و حاصلجمع آنها را در خروجی نمایش دهد:

```
#include <iostream.h>
#include <conio.h>

int main()
{
    int num[5];
    int sum = 0, count;

    for(count = 0 ; count < 5 ; count++)
    {
        cout << "Enter number " << count+1 << " :";
        cin >> num[count];

        sum += num[count];
    }
    cout << "\nSum of numbers is " << sum;
    getch();
    return 0;
}
```

```
Enter number 1: 1
Enter number 2: 5
Enter number 3: 9
Enter number 4: 0
Enter number 5: 18

Sum of numbers is 33
```

در برنامه بالا آرایه ای بنام num با طول ۵ تعریف شده که از یک حلقه تکرار for برای مقدار دادن به عناصر آن استفاده می کنیم و در درون حلقه، مقدار هر عنصر از آرایه را با متغیر sum که دارای مقدار اولیه صفر است جمع می کنیم. دقت کنید که حلقه for باید حتما از صفر شروع شود چون اندیس اولین عنصر آرایه صفر است. در نهایت و با خروج از حلقه حاصلجمع عناصر آرایه که در متغیر sum ریخته شده است را چاپ می کنیم .

ارسال آرایه به عنوان پارامتر برای تابع در برنامه نویسی C++

در این بخش قصد داریم به ارسال آرایه ها به عنوان پارامتر در تابع بپردازیم. در فصل توابع دیدم که متغیرها را به عنوان پارامتر به یک تابع ارسال کردیم، از آرایه ها هم می توان در پارامترهای توابع استفاده نمود. برای اینکار می توان آرایه را با طول ارسال نمود و یا آرایه را بدون طول ارسال کرد و با یک پارامتر دیگری طول را تعیین و ارسال نمود. اما بهتر آن است که آرایه را بدون طول ارسال نماییم و طول را با یک پارامتر دیگر ارسال کنیم، در هر صورت توجه داشته باشید که مانند تعریف آرایه ها هم نام، هم طول و هم نوع آرایه برای تابع و بطور کل برنامه C++ مشخص باشد. به اتفاق هم مثالی را در این زمینه بررسی می کنیم :

```
#include <iostream.h>
#include <conio.h>

int minFunction(int[], int);  //------------------> اعلان تابع با پارامترهای نام آرایه و طول آن

void main()
{
    const int k=4;
    int array[k];

    cout << "\nMinimum of array elements is " << minFunction(array, k);  //------> فراخوانی تابع
    getch();
}

int mainFunction(int arr[], int length)  //------> تعریف تابع با تمامی متعلقات
{
    for(count = 0 ; count < length ; count++)
    {
        cout << "Enter number [" << count+1 << "] :";
        cin >> arr[cout]
    }

    int minNum = arr[0];

    for(count = 1 ; count < length ; count++)
    {
        if (arr[count] < minNum)
            minNum = arr[cout]
    }

    return minNum;
}
```

```
Enter number 1 : 12
Enter number 2 : 3
Enter number 3 : 28
Enter number 4 : 109

Minimum of array elements is 3
```

همانطور که در کد C++بالا می بینیم، در اعلان توابع باید نوع و نام آرایه به همراه عللامت [] آورده شود تا کامپایلر تشخیص دهد پارامتر ورودی یک تابع است نه عدد صحیح. دومین پارامتر، طول آرایه را به تابع ارسال می کند که اینکار را می توانستیم مستقیما در درون پارامتر اول (آرایه) نیز انجام دهیم.

نکته) می بینیم که در فراخوانی تابع که در دستور cout قرار گرفته است ما فقط نام های پارامترها را قید می کنیم و از بیان نوع و [] اجتناب می کنیم.

در نهایت امر، تابع خود را با تمامی پارامترها و انواع و نامهایشان و ملزومات دیگر تعریف می نماییم.

نکته مهم) هر متغیر جز در درون تابع (حوزه) تعریفی خود قابل دسترسی نیست. بعنوان مثال ما در تابع main در کد بالا آرایه arrayرا تعریف کردیم و در و دیگر نمی توانیم از آن در تابع minFunction استفاده کنیم چون کامپایلر محدوده آنرا فقط در تابع خود می داند. اما در تابع minFunction آرایه دیگری را بنام arr تعریف کرده که از آن استفاده می کنیم. این نکته را بخاطر داشته باشید که ما آرایه array را بعنوان پارامتر به تابع ارسال کردیم، و آرایه arr همان آرایه array است اما با نامی دیگر.

برنامه C++بالا با استفاده از توابع و ارسال آرایه بعنوان پارامتر تابع، کوچکترین عدد آرایه را به ما نشان می دهد. برای این کار لازم است ابتدا عناصر آرایه را با یک دستور حلقه تکرار for مقدار دهی کنیم و سپس اولین عنصر را برابر با minNum بگیریم. سپس با یک حلقه for دیگر آنرا با دیگر عناصر آرایه مقایسه می کنیم، اگر مقداری کمتر از آن باشد در متغیر minNum ریخته می شود و اگر نباشد پس خود اولین عنصر از همه کوچکتر است. در دستور حلقه تکرار for دوم به این دلیل count را از ۱ شروع کردیم چون در بالا عنصر با اندیس صفرم آرایه را به عنوان کوچکترین عنصر در نظر گرفتیم و دیگر احتیاجی نیست آنرا دوباره با خودش مقایسه نماییم. عبارت const که در جلوی متغیر k آمده است به کامپایلر می گوید که مقدار این متغیر ثابت است و در طول برنامه تغییری نمی کند و نوشتن هر دستوری مبنی بر تغییر مقدار آن باعث بروز خطا در برنامه خواهد شد.

- آرایه چند بعدی Multi-Dimensional Arrays

A : array [0...3, 0...3] of number

A[3][3]

	0	1	2	3
0	0	1	2	3
1	4	5	6	7
2	8	9	10	11
3	12	13	14	15

آرایه دو بعدی با ۴ سطر و ۴ ستون

15	A[3,3]
14	A[3,2]
13	A[3,1]
12	A[3,0]
11	A[2,3]
10	A[2,2]
9	A[2,1]
8	A[2,0]
7	A[1,3]
6	A[1,2]
5	A[1,1]
4	A[1,0]
3	A[3,0]
2	A[2,0]
1	A[1,0]
0	A[0,0]

آرایه دوبعدی دارای چند سطر و چند ستون می باشد. اولین بعد آرایه بیانگر سطرها و دومین بعد آن بیانگر ستونهاست. همانطور که از شکل بالا پیداست به ترتیب از اولین سطر شروع کرده و تا انتهای ستونهای آن سطر را اندیس گذاری کرده و سپس به سطر بعد رفته و به اندیس گذاری تا آخرین ستون سطر مربوطه می پردازیم و این عمل را تا انتها برای اندیس گذاری تکرار می کنیم. سمت

راست شکل هم کاملا گویاست که عناصر آرایه به چه ترتیبی در حافظه کامپیوتر قرار می گیرند. در ادامه به نحوه تعریف این نوع از آرایه و بحث پیرامون آن می پردازیم.

- مقدار دهی به آرایه های چند بعدی:

```
char x[3][4] = {{'t', 'b', 'p', 'z'},   // Row 1
                {'m', 'c', 'c', 'q'},   // Row 2
                {'a', 'z', 'd', 'g'}};  // Row 3

x[0][0] = t;
x[1][3] = q;
x[2][1] = z;
x[2][0] = a;
x[2][3] = g;
```

همانطور که در دستور برنامه نویسی C++بالا مشخص است به ازای هر سطر (بعد اول) یک مجموعه و در درون آن مجموعه، عناصر ستونها (بعد دوم) را مقدار دهی می کنیم. در دستور تعریف آرایه بالا ما ۳ سطر و ۴ ستون داریم که به ازای بعد اول یک مجموعه و به ازای بعد دوم ۴ عنصر در هر مجموعه را تعریف می کنیم.

نکته) عزیزان توجه کنند که چون آرایه ما از نوع char است بنابراین برای هر عنصر باید از '' استفاده نماییم، و می دانیم که اگر stringباشد باید هر عنصر و بطور کل هر متغیر از این نوع را در " " قرار دهیم. (این یک قانون است).

مثال) برنامه ای با آرایه دو بعدی بنویسید که شماره هر سطر را نوشته و در مقابل آن جمع عناصر آن سطر را هم محاسبه و چاپ نماید:

```
#include <iostream.h>
#include <conio.h>

void sum(int[][2], int);  //------------------> اعلان تابع با پارامترهای نام آرایه و طول آن

void main()
{
    const int m=3, n=2;
    int matrix[m][n] = {{2,5}, {15,9}, {0,32}};

    sum(matrix, m);
    getch();
}

void sum(int x[][2], int a)
{
    int i, j;
    cout << "Row\t\t" << "Sum\t";
    cout << "---------------------\n";

    for (i=0 ; i<a ; i++)
    {
        int sum = 0;
        for (j=0 ; j<2 ; j++)
            sum += x[i][j];
        cout << i+i << "\t\t" << sum << "\n";
    }
}
```

```
Row         Sum
---------------
1            7
2            24
3            32
```

Pointers in C++

آموزش اشاره گر ها در برنامه نویسی C++

دوستان عزیز دقت داشته باشند که داریم کم کم از درب ورودی برنامه نویسی رد می شیم .
شاید پیش خودتون بگویید که پس مطالب قبلی چی بودند؟ باید عرض کنم که می شود گفت که اونا الفبای ابتدایی برنامه نویسی بودند. از این به بعد کم کم با مفاهیم اساسی برنامه نویسی روبرو خواهیم شد که درک هر چه بهتر این مفاهیم می تواند ما را در زمینه برنامه نویسی C++یا هر برنامه نویسی استاندارد دیگری به برتری برساند. اصول و مفاهیم کلی برنامه نویسی یکیه اما در زبانهای مختلف، راهها و قوانین نگارشی متفاوتی برای پیاده سازی کدها وجود دارد .
از این به بعد باید مفاهیمی را درک کنیم که در برنامه نویسی نقش بسزایی را ایفا می کنند، پس سعی کنید که مطالب را با دقت پیگیری کرده و به مثال ها و نوشتن برنامه بپردازید تا که به این مفاهیم تسلط پیدا نمایید .

تعریف مفاهیم

اشاره گر : متغیری است که آدرس خانه های سیستم را در خود نگه می دارد (آدرس هر متغیر در حافظه اشاره گر است).

متغیر هایی را که تا کنون با هم تعریف کردیم جدای از نوعشان همگی در خانه های حافظه ذخیره می شوند و هر کدام با توجه به طول نوع تعدادی از خانه ها را در حافظه اشغال می کنند.

آدرس خانه های حافظه : حافظه کامپیوتر از مجموعه ای از بایتهایی که هر کدام شامل ۸ بیت هستند تشکیل شده است. برای دسترسی به هر یک از این بایتها که به آنها خانه های حافظه گفته می شود شماره آدرسی وجود دارد که به ترتیب به هر یک داده می شود. به این شماره ردیف ها آدرس خانه حافظه گفته می شود.

آدرس متغیر : آدرس اولین بایت از حافظه که به یک متغیر اختصاص می یابد آدرس آن متغیر نامیده می شود.

می دانیم که متغیر ها در حافظه ذخیره می شوند اما مقدار حافظه مورد نیازشان با یکدیگر متفاوت است:

- char : 1 Byte
- int : 2-4 Byte
- float : 4 Byte
- double : 8 Byte

مثلا اگر ما متغیری از نوع int را تعریف کنیم ۲ یا ۴ بایت را اشغال می کند و به این معنی است که تعداد ۴ تا ۸ خانه پشت سر هم از حافظه را اشغال می کند که آدرس آن اولین آدرس خانه حافظه در نظر گرفته خواهد شد.

در مورد مزایای استفاده از اشاره گرها در C++یا زبانهای برنامه نویسی دیگر همین قدر باید گفت که در مدیریت حافظه و سرعت اجرای برنامه کمک میکند وسرعت، کارایی، دسترسی و ... را بالا می برد.

آدرس حافظه را فقط می توان در یک متغیر از نوع اشاره گر تعریف نمود :

```
Type *name;

int *Ptr;
```

چنانچه در بالا می بینیم برای تعریف متغیری از نوع اشاره گر، ابتدا نوع آن و سپس نام که از قانون نامگذاری برای متغیرها تبعیت می کند استفاده خواهیم کرد با این تفاوت که قبل از نام از علامت * برای نشان دادن تعریف اشاره گر بهره می گیریم .

تعریف اشاره گر بالا را می توان به طریق زیر تفسیر نمود :

- pمتغیر اشاره گری از نوع int است. (میتوان سطح تایپ هایی از نوع ....,float , double را انتخاب کرد)
- pآدرس خانه هایی از حافظه را نگهداری می کند که محتویات آن خانه ها، مقادیری از نوع int هستند.
- pمتغیری است که به محل هایی از حافظه که محتویاتی از نوع int دارد اشاره می کند.

دقت کنید که نوع یک اشاره گر به نوع متغیرهایی که به آدرس آنها اشاره می کند بستگی دارد و باید با آن یکسان باشد. یعنی نوع اشاره گری که به آدرسهایی که شامل متغیر هایی از نوع duoble است نمی تواند int یا char یا هر چیز دیگری غیر از همان duoble باشد.

تعریف مفاهیم

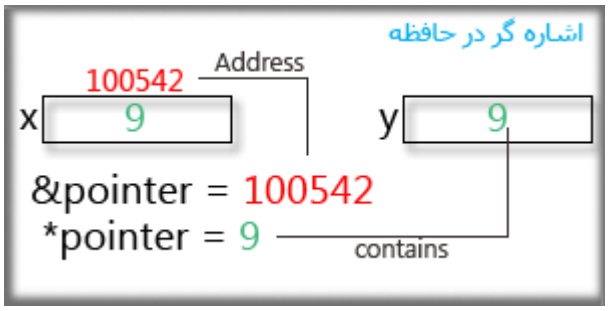
عملگرهای اشاره گر در C++

عملگر & : این عملگر دارای یک عملوند است و آدرس عملوند خود که نام یک متغیر است را مشخص می نماید.

عملگر * : این عملگر هم دارای یک عملوند است و محتویات (مقادیر) جایی که عملوندش (نام اشاره گر) به آنجا اشاره می کند را مشخص می کند.

```
int *pointer, x, y;    // define three variables whit int type
x = 9;                 // equal x to 9
pointer = &x;          // pointer contains Address of x
y = *pointer           // y equal the contains of content address of pointer
```

در کد C++بالا در خط اول ۳ متغیر از نوع int که یکی از آنها یک اشاره گر است تعریف شده اند. خط دوم مقدار ۹ را در متغیر x قرار می دهد. متغیر x با مقدار ۹ در جایی از حافظه ذخیره شده است که خط سوم آدرس آن محل را در متغیر اشاره گر pointer قرار می دهد یعنی pointer به جایی اشاره می کند که متغیری بنام x و از نوع int که مقداری برابر ۹ را دارد اشاره می کند. در آخرین خط محتویات جایی که آدرس آن در اشاره گر pointer ذخیره شده است در متغیر y قرار می گیرد یعنی مقدار ۹.




```
01  #include <iostream>
02  #include <conio.h>
03
04  using namespace std;
05  int main()
06  {
07        int firstValue, secondValue;
08        int *mypointer;
09
10        mypointer = &firstValue;
11        *mypointer = 10;       // firstValue = *myponter = 10
12        mypointer = &secondValue;
13        *mypointer = 20;       // secondValue = *myponter = 20
14        cout << "The firstValue is " << firstValue;
15        cout << "\nThe secondValue is " << secondValue;
16
17
18        getch();
19  }
```

```
The firstValue is 10
The secondValue is 20
```

در خطوط ۸ و ۷ برنامه C++بالا سه متغیر که ۲ متغیر معمولی از نوع int تعریف شده است و یک متغیر .pointer دستور خط ۱۰ آدرس متغیر firstValue را در اشاره گر قرار می دهد. در خط بعد محتویات جایی که اشاره گر به آنجا اشاره می کند را برابر با مقدار ۱۰ در نظر می گیریم. چون جایی که متغیر myPointer به آنجا اشاره می کند آدرس متغیر firstvalueاست پس مقدار ۱۰ در آنجا و در واقع در متغیر firstValue ریخته خواهد شد و به همین ترتیب برای متغیر secondValue

❖ عملیات روی اشاره گرها

کلا ۳ نوع عمل را می شود بر روی pointer انجام داد:

- 1.انتساب اشاره گر ها به همدیگر
- 2.اعمال محاسباتی جمع و تفریق
- 3.مقایسه

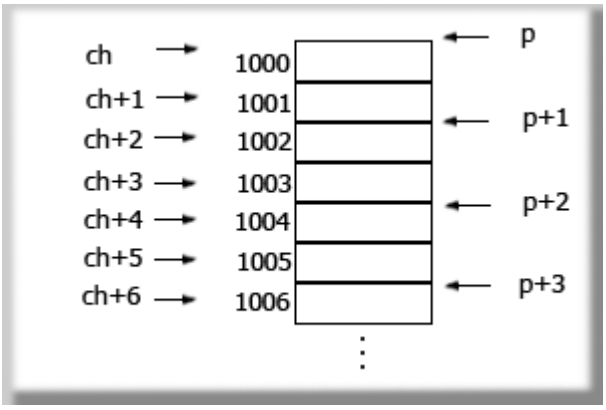
- انتساب اشاره گرها به همدیگر:

وقتی دو اشاره گر (pointer)را برابر با یکدیگر قرار دهیم با ۲ حالت مواجه خواهیم شد، یکی آدرس و دیگری مقدار است. به کد زیر دقت نمایید:

```
int *p1, *p2, *p3, a, b, c;
a = 50;
b = 100;
c = 200;
p1 = &a;  // p1 = Address a and *p1 = 50 and assume that &p1 = 10000
p2 = &b;  // p2 = Address b and *p2 = 100 and assume that &p2 = 10020
p3 = &c;  // p3 = Address c and *p3 = 200 and assume that &p3 = 10040
*p1 = *p2;   // *p1 = *p2 = 100 and &p1 = 10000 and &p2 = 10020
p1 = p3;   // &p1 = &p3 = 10040 and *p1 = *p3 = 200
```

در کد بالا و در خط ۵ اشاره گر p1 به جایی که متغیر a در حافظه ذخیره شده اشاره می کند بنابراین p1*برابر با مقدار آن آدرس یعنی ۵۰ خواهد شد. در دستور خط بعدی آدرس جایی که متغیر b در آنجا ذخیره شده درون اشاره گر p2 قرار می گیرد یعنی به آن آدرس اشاره می کند و چون محتویات آن آدرس برابر با ۱۰۰ است پس مقدار p2*برابر با ۱۰۰ می شود. در سطر ۸ با نوشتن این دستور فقط محتویات جایی که p1 به آنجا اشاره می کند درون درون جایی که p2 به آنجا اشاره می کند ریخته می شود و آدرس دو اشاره گر تغییر نخواهد کرد. اما در سطر ۹ آدرس جایی که p3 به آنجا اشاره می کند در اشاره گر p1 قرار می گیرد، بنابراین جایی که p1 به آنجا اشاره می کرد برابر با جایی می شود که p3 اشاره می کند که به ازاء این تغییر بنابراین محتویات جایی که p1 به آنجا اشاره می کرد به مقدار ۲۰۰ تغییر می کند .

- اعمال جمع و تفریق:



وقتی اشاره گری را افزایش یا کاهش می دهیم در واقع اینکار را با آدرسها انجام خواهیم داد و متناسب با طول آن متغیر کم یا زیاد می کنیم. در شکل بالا p متغیری از نوع int است پس با هر واحد افزایش، pointer 2خانه یعنی ۲ بایت در حافظه پیش خواهد رفت و متغیر char از نوع char در نظر گرفته شده که با هر واحد افزایش یا کاهش فقط یک خانه کم یا زیاد می شود.

- مقایسه اشاره گرها:

وقتی در مورد اشاره گرها صحبت می کنیم ذهن خود را به سمت آدرس حافظه ببرید. وقتی دو اشاره گر با یکدیگر برابرند که هر دو به یکجا اشاره نمایند، در این صورت هر تغییر در آن آدرس منجر به تغییر هر دو اشاره گر خواهد شد اما اگر مقدار آدرس دو اشاره کر برابر نباشد هر چند که محتویات آن دو آدرس مختلف با هم برابر باشد نمی توان گفت که آن دو با هم برابر هستند زیرا با تغییر در محتوا یا آدرس یکی از آن دو دیگری تغییری نمی کند.

```
int *p1, *p2, char1, char2;
char1 = char2 = 'Q';
p1 = &char1;
p2 = &char2;
if( p1 == p2 )
    ...
else
    ...
```

د C++ بالا چون اشاره گرها هر یک به آدرسهای مختلفی اشاره می کنند هر چند مقادیر آن آدرسها یکی است اما با هم برابر نیستند.

❖ اشاره گر ها به عنوان آرگومان های توابع :

همچنان که پیشتر در فصل توابع در مورد آرگومان های توابع بحث شد می توان از اشاره گر ها بعنوان آرگومان تابع استفاده نمود. در کد زیر از تابع foo بهره می گیریم:

```
#include <iostream.h>
```

```cpp
#include <conio.h>

using namespace std;

void foo(int &y)
{
    cout << "y=" << y << "\n";
    y = 6;
    cout << "y=" << y << "\n";
}

void main()
{
    int x = 5;
    cout << "x=" << x << "\n";
    foo(x);
    cout << "x=" << x << "\n";
    getch();
}
```

```
x=5
y=5
y=6
x=6
```

در کد برنامه نویسی C++تابعی با نام foo که دارای یک آرگومان از نوع اشاره گر است را تعریف نمودیم. این تابع حاوی آدرس پارامتر ورودی خود است پس یک اشاره گر است. در درون تابع foo ابتدا یک بار مقدار متغیر ورودی که در این تابع نامش y و در تابع main نامش x است راچاپ می کند اما با استفاده از مراجعه به آدرس آن متغیر.

از قبل یادگرفتیم که در استفاده از توابع با آرگومان های معمولی، با آدرس آن متغیر کامپایلر کاری ندارد بلکه کپی از آنرا تهیه نموده و از آن استفاده می کند، پس هر گونه تغییر در مقدار متغیر ها در این حالت با برگشت به تابع main از بین خواهد رفت چون کامپایلر آن کپی را دور می اندازد و اینبار و در تابع main یا هر حوزه جدید دیگری دوباره از آن متغیر کپی گرفته و استفاده می کند.

اما در مورد اشاره گر ها دیگر چون مستقیما در محل حافظه تغییرات صورت می گیرد پس با کپی گرفتن مجدد در هر حوزه یا تابع دیگری مقادیر جدید را بدست می آورد، بنابراین در کار با اشاره گر ها تغییرات آنی اعمال می شوند ومتغیر ها حوزه ای ندارند.

خوب، در سطر ۸ مقدار متغیر به ۶ تغییر می کند و در سطر بعدی عدد ۶ بعنوان مقدار متغیر جاری درج می گردد. در بازگشت به تابع main چون آرگومان اشاره گر بوده پس تغییرات در آدرس متغیر اعمال شده و با مراجعه کامپایلر به آدرس مورد نظر و چاپ مقدار آن دوباره عدد ۶ چاپ می شود در صورتی که اگر آرگومان معمولی بود ۵ چاپ می شد یعنی همان مقدار اولیه متغیر در این تابع. فراموش نکنید که برنامه از تابع اصلی آغاز می شود بنابراین در ابتدا مقدار x که ۵ است چاپ خواهد شد.

برای درک تفاوت میان روش ارسال پارامتر ها با استفاده از مقادیر یا ارجاع (آدرس) مثالی دیگری را بررسی خواهیم نمود:

```
#include <iostream.h>
#include <conio.h>

void pass(int, int *);

void main()
{
    int x = 1, y=1;
    pass(x, &y);
    cout << "x=" << x << "\n";
    cout << "y=" << y << "\n";
    getch();
}
void pass(int a, int *b)
{
    cout << "x=" << ++a << "\n";
    cout << "y=" << ++*b << "\n";
}
```

```
x=2
y=2
x=1
y=2
```

با اجرای کد برنامه C++بالا، کامپایلر در سطر ۶ ابتدا وارد تابع main خواهد شد. دو متغیر از نوع int با نامهای x,y تعریف شده که هر کدام دارای مقدار برابر یک هستند. در سطر ۸ کامپایلر با فراخوانی تابع pass به سطر ۱۴ می رود. در این تابع دو پارامتر که یکی int و دیگری اشاره گر است را می بیند. ما با اینکار مقدار x را با ارسال پارامتر بروش مقدار و y را با استفاده از ارسال پارامتر بروش ارجاع انجام داده ایم، یعنی مقدار x را و آدرس y را به تابع می فرستیم .

در سطر ۱۶ ابتدا یک واحد به x اضافه شده وسپس چاپ می شود که مقدار ۲ را خواهیم داشت.

در سطر ۱۷ ابتدا یک واحد به مقدار جایی که آدرس آن محل به عنوان پارامتر (اشاره گر) به تابع ارسال شده اضافه خواهد شد و بعد ۲ چاپ خواهد شد.

با پایان بلوک تابع pass کامپایلر باز به تابع main بازگشته و ادامه دستورات را از سطر ۱۰ از سر می گیرد.

در سطر ۱۰ چون x با مقدار به تابع ارسال شده بود پس مقدار آن در ابن تابع همان ۱ است پس اینبار ۱ چاپ خواهد شد.

اما در سطر ۱۱ چون با آدرس متغیر y کار کردیم پس مقدار کنونی در این تابع هم دستخوش تغییرات در تابع pass خواهد بود و عدد ۲ برای آخرین دستور چاپ می شود و برنامه پایان میابد.

امیدوارم که توانسته باشم مفهوم ارسال با مقدار و ارجاع یا آدرس را رسانده باشم. اما اگر متوجه نشدید اصلا نگران نباشید و با کمی تمرین براحتی مطلب را درک خواهید نمود.

❖ اشاره گر ها و آرایه ها :

دیدیم که عناصر آرایه پشت سر هم در حافظه قرار می گیرند و بدانید که نام آرایه یک اشاره گر است. در واقع نام آرایه در برگیرنده آدرس اولین عنصر آرایه در حافظه است و چونکه عناصر آرایه بدون فاصله و به ترتیب در حافظه قرار می گیرند بنابراین می توان با داشتن نام آرایه که در واقع یک نوع اشاره گر است به کلیه عناصر آن در برنامه نویسی زبان C++دست پیدا کرد.

در قطعه برنامه C++زیر به بررسی ارتباط اشاره گر ها و آرایه ها می پردازیم:

```
char array[] = {'c','s','h','a','r','p'};
char *pointer;
pointer = array;            // آرایه و اشاه گر هر دو به یکجا یعنی ابتدای آرایه اشاره می کنند
*pointer = array[0];        // = 'c'
*(pointer+1) = array[1];    // = 's'
pointer[3] = *(array+3);    // = 'a'
*pointer = *array;          // = 'c'
```

همانطور که در بالا ملاحظه می کنید به راحتی می توان از اشاره گر بجای آرایه ها استفاده کرد.

```
#include <iostream.h>
#include <conio.h>

void main()
{
    int numbers[5];
    int *p;

    p = numbers; //9
    *p = 10; //10
    p++;
    *p = 20;
    p = &numbers[2];
     *p = 30;
    p = numbers+3;
    *p = 40;
    p = numbers;
    *(p+4) = 50;
    for(int n=0 ; n<5 ; n++) //19
        cout << numbers[n] << ", ";  //20
    getch();
}
10, 20, 30, 40, 50
```

در قطعه برنامه C++بالا آرایه ای بنام numbers و یک اشاره گر بنام p تعریف کرده ایم. در سطر ۹ اشاره گر را برابر با عنصر اول آرایه قرار می دهیم پس هر دو به یکجا اشاره می کنند. سپس در سطر ۱۰ بجایی که اشاره گر اشاره می کند و برابر با عنصر اول آرایه است مقدار ۱۰ را می دهیم. در سطر بعد اشاره گر را یکواحد افزایش می دهیم که با اینکار به خانه بعدی یا همان عنصر دوم آرایه اشاره می کند و باز مقدار محتوای آن خانه را برابر با ۲۰ قرار می دهیم. در کد بالا سعی شده تا انواع برابری های آرایه و اشاره گر را نشان دهم تا برای شما عزیزان بحث جذابتر و روشنتر گردد. به همین

روال ما به اشاره گر مقدار می دهیم و چون اشاره گر و آرایه با هم برابرند پس آرایه نیز با اینکار مقداردهی می شود.
سپس با استفاده از یک حلقه تکرار for در سطرهای ۱۹و۲۰ مقادیر آرایه را چاپ می کنیم.

خدمت عزیزان باید عرض کنم که از اشاره گر ها نیز می توان بجای رشته ها استفاده نمود و مطالب مانند مطالب آرایه ها دنبال خواهد شد پس از بیان این مورد صرف نظر می کنم.

```
//Life motto
while (noSuccess)
{
    tryAgain();

    if (Dead)
        break;
}
```