



سی پلاس پلاس

به زبان ساده

:Author:

AmirHosein Bahrami

Email : info@codena.ir



((فهرست مطالب))

Basic of Developer

1- مبانی برنامه نویسی

چکیده : قبل از شروع اولین زبان زبان برنامه نویسی شخص باید یک آشنایی از الگوریتم و فلوچارت و مسئله پیدا کند تا بتواند مسائل خود را به درستی تحلیل و بررسی نماید و برای هر یک از آنها راه حل درستی پیشنهاد و انجام بدهد.

در ادامه به ترتیب هر یک را برای شما شرح میدهیم :

1-1- مسئله چیست :

Problem : an obstacle which makes it difficult to achieve a desired goal, objective or purpose

((یک مانعی که دستیابی به هدف مورد انتظار را سخت می کند))



خب حالا ما چند نوع هدف داریم =>

Problem Types

1 : Purpose (اهداف نهایی)

2 : Objective (اهدافی که برای رسیدن به هدف نهایی باید کلی برنامه ریزی کرد)

3 : Goal (اهداف کوتاه مدت)

1-2- الگوریتم چیست :

Algorithm : an effective method for solving a problem expressed as a finite sequence of instructions

((یک روش کارا برای حل یک مسئله که در قالب توالی محدودی از دستورالعمل ها تعریف شده))



در ادامه الگوریتم، چند مسئله را مثال میزنیم <=

مسأله : شمارش تعداد افراد کلاس

((الگوریتم))

1- شروع

2- همه بایستید

3- به خودتان عدد 1 را نسبت دهید

4- عددتان را با یک نفر در اطرافتان مقایسه کنید و

عددتان را به عدد او اضافه کنید

5- اگر (کسی در اطرافتان نیست)

6- }

7- عددتان را به مربی اعلام کنید

8- بروید به 12

9- {

10- در غیراینصورت

11- یکی از شما باید بنشیند و

دیگری به مرحله 4 برو

12- پایان



مسأله : یافتن و پوشیدن جوراب

((الگوریتم))

1- شروع

2- در کیسه را باز کن

3- یک جوراب از داخل کیسه بردار

4- جوراب را بپوش

5- تا زمانی که (جفت دوم جوراب پیدا شود یا جوراب‌های

کیسه تمام شود)

6- }

7- یک جوراب بردار

8- اگر (جوراب جدید=جفت دوم جوراب)

9- }

10- جوراب را بپوش

20- برو به 4

11- برو به 22

21- {

12- }

22- پایان

13- در غیر این صورت

14- جوراب را در کیسه دیگری بگذار

15- {

16- اگر (جوراب‌های کیسه تمام شد و جفت دوم پیدا نشد)

17- }

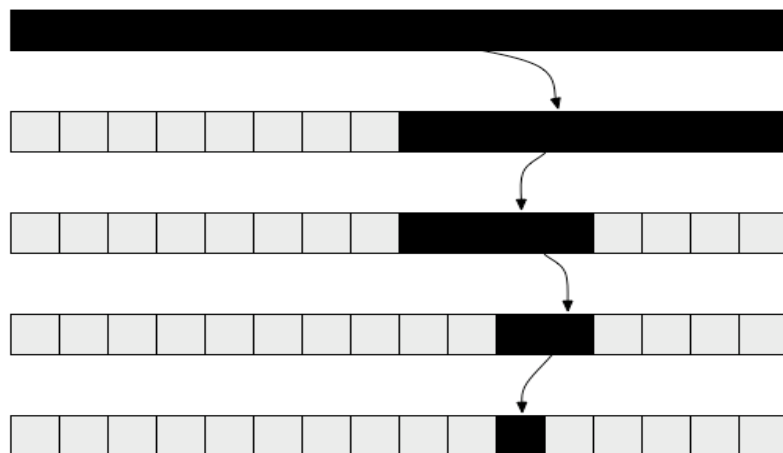
18- جوراب اول را به کیسه اول بینداز



مسأله : جستجوی یک لغت در یک فرهنگ لغت :



Binary Search



خب ما حالا چند نوع جست و جو در سیستم داریم <=

- Search Types: {
- 1 : Line Limier Search (خطی)
 - 2 : Binary Search (دودویی)

چند بازی الگوریتمیک برای تقویت هوش برنامه نویسان <=

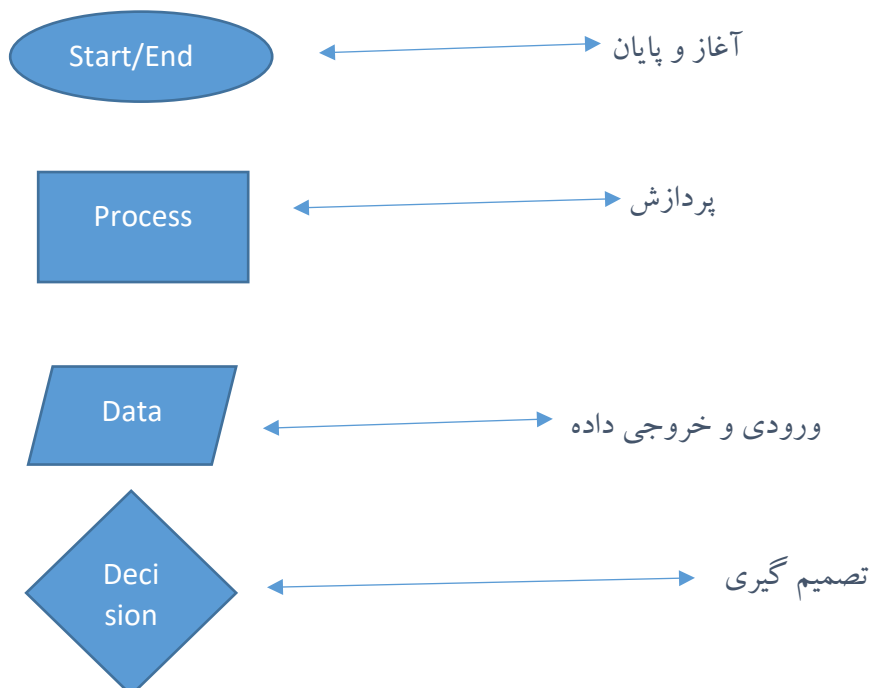
- 1 - Light Bot(computer & Mobile)
- 2 - Mine Sweeper(computer)
- 3 - Limbo(computer & Mobile)
- 4 - Scratch(computer)

3-1- فلوچارت چیست :

Flowchart : A flowchart is a common type of diagram, that represents an algorithm or process, showing the steps as boxes of various kinds, and their order by connecting these with arrows.

((فلوچارت یک دیاگرام عمومی است که نماد یک الگوریتم یا یک فرآیند است. در فلوچارت گام های الگوریتم با باکس های مختلف نمایش داده میشود و نهاد آنها نشانگر به ترتیب می باشد.))

نماد های فلوچارت <=



در ادامه فلوچارت، چند مسئله را مثال میزنیم <=

مسئله: جمع اعداد 1 تا 100

((الگوریتم))

1- شروع

2- یک مخزن بزرگ در نظر بگیر

3- یک شمارنده در نظر بگیر که از 1 شروع به شمردن می کند

4- تا زمانی که (شمارنده کمتر از 100 است)

5- }

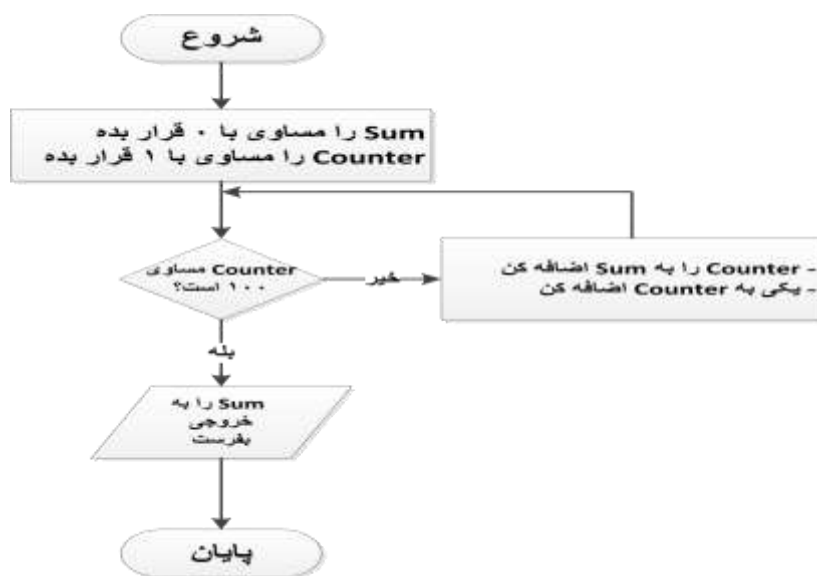
6- عدد شمارنده را به مخزن اضافه کن

7- یکی به شمارنده اضافه کن

6- }

8- پایان

7- مقدار مخزن را به خروجی بفرست.



((فلوچارت))



2-1- تاریخچه : ++C توسط بی یارنه استراس تروپ

ریاضیدان دانمارکی در سال ۱۹۷۹ در آزمایشگاه‌های بل (Bell Labs)، برای بهبود زبان سی و بر مبنای آن ساخته شد و آن را C با کلاس (C With Classes) نام گذاری نمود.

زمانی که استراس تروپ کار خود را در آزمایشگاه‌های بل (Bell Labs) آغاز نمود با مشکل تحلیل هسته یونیکس با توجه به محاسبات توزیع شده روبرو شده بود.

او زبان C را با استفاده از ویژگی‌های سیمولا گسترش داد. (C) به این دلیل انتخاب شد که یک زبان عمومی، سریع، قابل حمل، و در سطح گسترده‌ای در حال استفاده بود.

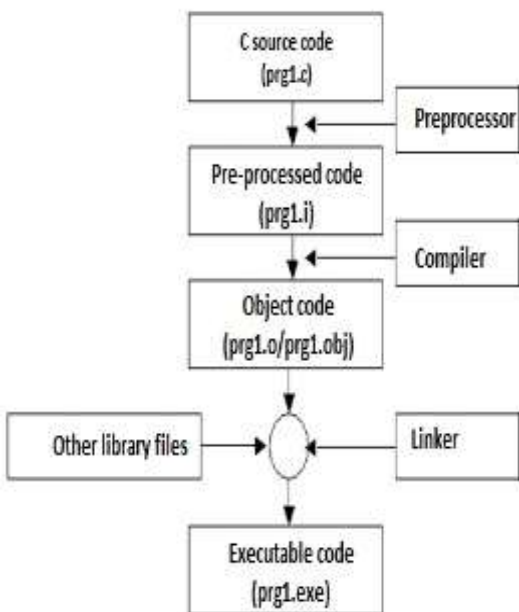
علاوه بر C و سیمولا زبان‌های دیگری مانند ALGOL ۶۸، ADA, CLU, ML نیز بر ساختار این زبان جدید اثر گذاشت. در ابتدا ویژگی‌های کلاس، کلاس‌های مشتق شده، کنترل نوع قوی، توابع درون خطی و آرگومان‌های پیش فرض از طریق Cfront به C اضافه شد و اولین نسخه تجاری ++C در سال ۱۹۸۵ ارائه شد.

2-2- ساختار یک برنامه structure of a program

یک برنامه برای اجرا 0 تا 100 طی فرایند هایی قرار میگیرد.

1: Editor\IDE = < یک برنامه برای نوشتن کدها داخل آن

2: Preprocessor = < برنامه های پیش پردازنده دستورات پیش پردازنده را فراهم می کند که کامپایلر را پیش از پردازش کد منبع قبل از کامپایل مجاز می کند.



2-3- Compiler: 3 = < برنامه ای است که غلط املائی و غلط گرامری را یافت و گوشزد میکند.

4 = Linker: 4 = < برنامه ای است که فایل های مورد نیاز برای برنامه اصلی را لینک میکند.

5 = Interpreter: 5 = < یعنی کد نوشته شده به زبان C برای پلتفرم های نهایی آماده شده و برای آن خروجی گرفته شود.

Syntacs غلط املائی

Semantic غلط گرامری

PlatformTypes :

+Hardware : تلویزیون-یخچال و....

+Software : ویندوز-لینوکس و....

چند کامپایلر C++ = <

GCC

CLANG

Turbo C++

Clang

And.....

IDE چیست = < مخفف (integrated development environment) به معنای محیط توسعه یکپارچه یک نرم افزار کاربردی را فراهم می کند که امکانات جامع برای برنامه نویسان کامپیوتر برای توسعه نرم افزار را میدهد..شامل تمام نیاز برای ایجاد یک برنامه —Linker—editor Compiler.....

چند IDE برای C++ = <

Turbo C++

Borland C++
Visual Studio
Dev C++
And.....

ساده ترین برنامه به زبان C++

```
// My First Program In C++  
#include <iostream.h>  
Using namespace std;  
int main()  
{  
    cout << "Hello world" ;  
    return 0 ;  
}
```

کد برنامه

Hello World

خروجی

// My First Program In C++

Comment <= این یک خط Comment یا توضیحات می باشد. در برنامه نویسی، توضیحات خطوطی هستند که کامپایلر نمی شوند و فقط برای خوانایی برنامه بکار برده می شوند. در برنامه های بزرگتر و با زیاد شدن خطوط برنامه، توضیحات به فهم راحتتر برنامه برای برنامه نویس و دیگر کسانی که کد را مورد بررسی قرار می دهند کمک شایانی می نماید.

انواع Comment <=

1 : // My First Program In C++

کامنت یک خطی

2 : /* My First

Program

کامنت چند خطی

In C++ */

1-Help

2-Disable

کاربردها:

#include <iostream.h>

کتابخانه <= از فایل‌های سرآیند بعنوان کتابخانه های C++ یاد می کنند که از قبل نوشته شده اند و ما برای استفاده از برخی از توابع و روالها از آنها استفاده می کنیم . کامپایلر فقط کلمات کلیدی را می شناسد و همانطور که گفته شد برای استفاده از یک سری دستورات و توابع مانند دستورات ورودی و خروجی و ... باید از این فایل‌های سرآیند یا Header استفاده کنیم. این نکته را هم خاطر نشان می کنم که پسوند این فایلها h می باشد.

int main()

{
}

تابع Main <= این خط تعریفی برای تابع اصلی برنامه می باشد. در واقع برنامه با کامپایل از این نقطه شروع و پردازش می شود. هر برنامه C++ باید دارای تابع main() باشد. در این تابع است که بلوکها و خطوط برنامه نوشته میشود. در واقع سرآیند برنامه های C و مبتنی بر C است.

برای مثال شما ساختمانی را در نظر بگیرید که میخواهد وارد آن شوید و یک کاری را انجام دهید . خب قطعاً قبل از هر کاری شما باید از درب اصلی وارد آن ساختمان شوید و بعد کارهای خود را انجام دهید....خب این تابع هم همینگونه است، اگر درب اصلی نباشد قطعاً شما هم نمیتوانید کار خود را انجام دهید.

}

آکولاد باز در واقع علامت شروع یک بلوک از دستورات را تعریف می کند که در این برنامه بدنه تابع اصلی را در بر میگیرد و با علامت

{

یا آکولاد بسته پایان بلوک را تعیین می کند.

C++ _ C# _ Java _ JavaScript _ Php

چند زبان C-based <=

Using namespace std;

فضای نام <= باید بگم که فضاهای نام هم در ساختار C++ نقش اساسی دارند. این فضاها مجموعه دیگری از کتابخانه های C++ می باشند که در استفاده از بعضی عناصر مانند رشته ها و ... کمک شایانی به کاربران می کنند.

شما فکر کنید برنامه ای برای یک دانشگاه نوشته اید و میخواهید اساتید و دانشجویها به این برنامه دسترسی داشته باشند..خب قطعاً اصولی نیست وقتی دانشجویم زد قسمت نمرات بیاد، برای استاد هم نمایش داده بشه. برای اینکه ما دسترسی قرار بدیم و کدهای مربوط به هر بخش را از یکدیگر جدا کنیم از فضای نام استفاده می کنیم.

cout << "Hello world";

تابع cout <= Cout در تابع سرآیند iostream در زبان C++ قرار دارد و موجب ارسال اطلاعات به خروجی و چاپ و نمایش آن بر روی مانیتور کاربر می شود.

return 0 ;

<= Return این عبارت مقدرا صفر را به تابع در برگیرنده خود که در این مثال تابع اصلی است برمی گرداند که این مورد در مبحث توابع و انواع بازگشتی آن توضیح داده خواهد شد .

Variables and Data type

متغیر ها و نوع داده ها -2-4

متغیر=> بخشی از حافظه که در طول اجرای برنامه ، توسط برنامه نویس قابل دسترسی و قابل ویرایش است.

تعریف اولیه یک متغیر به صورت مختلف :

نوع متغیر مقدار_اولیه_متغیر = نام_متغیر

Int x = 100;

Int x ;

قواند نام گذاری یک متغیر :

1. نام متغیر نمیتواند با عدد شروع شود.
2. نام متغیر فقط میتواند با حروف کوچک یا بزرگ انگلیسی یا (-)شروع شود.

3. در نام متغیر استفاد از حروف بزرگ و کوچک انگلیسی و عدد و (-) مجاز است.

4. نام متغیر نمیتواند از کلمات کلیدی زبان C استفاده کند.

((انواع داده های پایه ای))

Name	Description	Size	Range
Char	Character or small integer	1 bytes	signed: -128 to 127 unsigned: 0 to 255
Short int(short)	Short Integer	2 bytes	signed: -32768 to 32767 unsigned: 0 to 65535
Int	Integer	4 bytes	signed: -2147483648 to 2147483647 unsigned: 0 to 4294967295
Long int(long)	Long integer	4 bytes	signed: -2147483648 to 2147483647 unsigned: 0 to 4294967295
Bool	Boolean value. It can take one of two values: true or false	1 bytes	signed: -2147483648 to 2147483647 unsigned: 0 to 4294967295
Float	Floating point number	4 bytes	+/- 3.4e +/- 38 (~7 digits)
Double	Double precision floating point number	8 bytes	+/- 1.7e +/- 308 (~15 digits)
Long double	Long double precision floating point number	8 bytes	+/- 1.7e +/- 308 (~15 digits)
Wchar_t	Wide character	2 or 4 bytes	1 wide character

تعریف اولیه چند متغیر به صورت مختلف :

نام_متغیر_دوم , مقدار_اولیه_متغیر_اول = نام_متغیر_اول نوع متغیر
;مقدار_اولیه_متغیر_دوم

جدا کردن با (,) از یکدیگر..

Int x = 100,y=20,w=0;

Int x,y,w ;

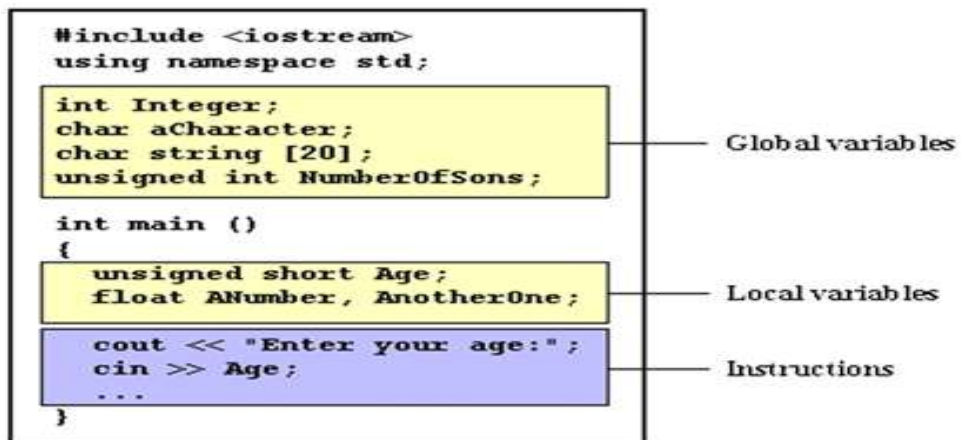
Scope Of Variables

2-4-1- محدوده متغیر ها

(1) Global : متغیر در تمام زیر مجموعه ها قابل استفاده است.

(2) Local : متغیر در محدوده یک آکولاد باز و بسته و زیر مجموعه
هایش قابل استفاده است.

انواع Scope



کد برنامه

```

#include <iostream.h>
Using namespace std;
int main()
{
    Int x=10;
    String y="SALAM";
    cout << x,y ;
    return 0 ;
}

```

خروجی

10SALAM

رشته ها در زبان C++ => اگر کتابخانه <String> در سرآیند برنامه اشاره شود ، نوع داده String در برنامه قابل استفاده خواهد بود.

انواع دریافت پارامتر از کاربر :

1) اگر در بین رشته ها فاصله وجود ندارد از Cin استفاده میکنیم.

(2) اگر در بین رشته ها فاصله وجود دارد از GetLine استفاده میکنیم.

کد برنامه

```
#include <iostream.h>
#include <string>
Using namespace std;
int main()
{
    Int Age;
    String Name;
    cout << "Please Enter Age: /n" ;
    cin >> Age;
    cout << "Please Enter Age: /n" ;
    cin >> getline(cin,Name);
    cout << Age,Name<<endl ;
    return 0 ;
}
```

```
17 Amir
Hosein
```

خروجی

انواع رفتن به خط بعدی <=

/n

Endl

```
Cout << "Test \n";
Cout << Endl ;
```

مانند متغیر ، بخشی از حافظه هستند که برنامه نویس امکان ذخیره سازی داده در آن مکان را دارد. اما بربر خلاف متغیر ، پس از آن که یک متغیر ثابت مقدار دهی اولیه شد ، در طول اجرا برنامه مقدار آن تغییر نمی کند.

انواع ثوابت :

- Literals : از پیش تعریف شده ← (با استفاده از Define)
 مقدار اسم_ثابت #Define ← #Define Pi 3.14

- UserDifined : تعریف شده از کاربر ← (با استفاده از Const)
 ; متغیر_دارای_مقدار اسم_ثابت نوع_ثابت Const ← Const int Pi = R ;

((کاراکترهای Escape Code در C++))

Describe	Abstract	Example	Description
\n	New Line	Cout<<"Test \n";	رفتن به خط جدید
\r	Carriage Return	Cout<<"Test \r";	اوردن متن به اول خط
\t	Tab	Cout<<"Test \t";	رفتن 8 کاراکتر جلو تر
\v	Vertical Tab	Cout<<"Test \v";	رفتن 8 کاراکتر پایین تر
\b	Backspace	Cout<<"Test \b";	پاک کردن یک کاراکتر
\f	Form Feed(form feed)	Cout<<"Test \f";	چاپ \

\a	Alert(beep)	Cout<<"Test \a";	ایجاد یک صدا
\'	Single Quote(')	Cout<<"Test \'";	چاپ '
\"	Double Quote(“)	Cout<<"Test \";	چاپ “
\?	Question Mark(?)	Cout<<"Test \?";	چاپ ?
\\	backslash	Cout<<"Test \\";	چاپ \

نکته : طول cmd برابر با 80 کاراکتر است.

Operators

2-6- عملگرها

عملگر $=$ عملگرها نمادهایی هستند که اعمال خاصی را انجام میدهند. بعنوان مثال، نماد '+' عملگری است که دو مقدار را با هم جمع میکند. پس از اعلان متغیرها و مقدار دادن به آنها باید بتوان عملیاتی را روی آنها انجام داد. برای انجام این عملیات باید از عملگرها استفاده کرد.

عملگرها روی یک یا دو مقدار عمل میکنند ، مقادیری را که عملگرها روی آنها عمل میکنند، عملوند گویند .

انواع عملگرها :

- 1 - محاسباتی
- 2 - افزایشی و کاهشی
- 3 - رابطه ای
- 4 - انتسابی
- 5 - منطقی
- 6 - بیتی

عملگر محاسباتی <= عملگرهای محاسباتی عملگرهایی هستند که اعمال محاسباتی مثل جمع ، تفریق ، ضرب و.. را روی روی عملوندها انجام می دهند.

((انواع عملگرهای محاسباتی))

Arithmetic Operators	Example	Description
+	Int x = 12 * 8;	جمع دو یا چند مقدار باهم
-	Int x = 48 * 23;	تفریق دو یا چند مقدار باهم
*	Int x = 7 * 3;	ضرب دو یا چند مقدار باهم
/	Int x = 18 / 2;	تقسیم دو یا چند مقدار باهم
%	Int x = 69 % 2;	باقیمانده تقسیم دو یا چند مقدار باهم

```
#include <iostream>
int main()
{
    int a = 11;
    int b = 3;
    int c = a % b;
    cout <<"c:";
    cout <<c;
    return 0;
}
```

کد برنامه

عملگر افزایشی و کاهشی <= افزایش مقدار نگهداری شده در یک متغیر، یکی از امور رایج در برنامه نویسی است. فرض کنید شما یک متغیر به اسم counter تعریف کردید و می‌خواهید هر بار که عملیاتی در برنامه شما انجام شد، به متغیر counter شما یک واحد اضافه شود.

((انواع عملگرهای افزایشی و کاهشی))

Incremental and decreasing operators	Example	Description
++	Int x = 22; x++;	اضافه کردن یک عدد به مقدار و رفتن به خروجی
++	Int x = 22; ++x;	رفتن به خروجی و اضافه کردن یک عدد به مقدار
--	Int x = 22; x--;	کم کردن یک عدد از مقدار و رفتن به خروجی
--	Int x = 22; --x;	رفتن به خروجی و کم کردن یک عدد از مقدار

```
#include <iostream>
int main()
{
    int a = 2, b = 3;    // a=2, b=3
```

کد برنامه

```

a += b++;      // a+=(b+1) ---> a=a+(b+1) ---> a=2+4=6
cout <<"a:";
cout <<a;
return 0;
}

```

عملگر رابطه ای $=$ عملگرهای رابطه ای ارتباط بین عملوندها را مشخص می کنند. اعمالی مثل تساوی دو مقدار. کوچکتر یا بزرگتر بودن. مقایسه با صفر و غیره توسط عملگرهای رابطه ای مشخص می شود. در مورد عملگرهای رابطه ای شاید با عملگر $==$ آشنایی نداشته باشید. این عملگر در دستورات شرطی برای مقایسه دو مقدار مورد استفاده قرار می گیرد.

((انواع عملگرهای رابطه ای))

Relational Operators	Example	Description
$>$	$x > y;$	بزرگتر
$<$	$x < y;$	کوچکتر
$>=$	$x >= y;$	بزرگتر مساوی
$<=$	$x <= y;$	کوچکتر مساوی
$==$	$x == y;$	متساوی
$!=$	$x != y;$	نا متساوی

کد برنامه

```

#include <iostream>
int main()
{
    int a = 2, b = 3;

    if(a > b)
        cout <<"a:";
    if(a < b)
        cout <<"b:";
}

```

```

if(a >= b)
    cout << "a:";
if(a <= b )
    cout << "a:";
if(a == b)
    cout << "a:";
if(a != b)
    cout << "a:";

return 0;
}

```

خروجی

3
2
2

عملگر انتسابی <= این عملگرها ترکیبی از عملگرهای محاسباتی و عملگر انتساب یعنی = است.

((انواع عملگرهای انتسابی))

Assigned Operators	Example	Description
+=	x+=y;	انتساب جمع
- =	x-=y;	انتساب تفریق
/=	x/=y;	انتساب تقسیم
=	X=y;	انتساب ضرب
%=	x%=y;	انتساب درصد

```

#include <iostream>
int main()
{
    int a = 2, b = 3;
    a*=b;
}

```

کد برنامه

```
cout<<a;
return 0;
}
```

6

خروجی

عملگر منطقی $\leq$ عملگرهای منطقی روی منطق برنامه عمل میکنند مثلاً آیا متغیر x با مقدار 10 بین 5 و 15 وجود دارد یا نه؟

((انواع عملگرهای منطقی))

Logical Operators	Example	Description
!	$(x > 5 \text{ ! } x == 10)$	نباشد
&&	$(x > 5 \text{ \&\& } x == 10)$	و
	$(x > 5 \text{ } x == 10)$	یا

```
#include <iostream>
int main()
{
    int a = 2, b = 3;

    if(a > 0 && b == 4)
        cout >> "a: ";
    if(a > 0 || b == 10)
        cout << "3";

    return 0;
}
```

کد برنامه

عملگر بیتی $\leq$ این عملگرها روی بیت‌ها اعمال می‌شود و مشابه عملگرهای منطقی است فرض کنید دو بیت داریم به صورت ۰ و ۱ بنابراین برای جدول درستی آنها داریم:

((انواع عملگرهای بیتی))

Bit Operators	Example	Description
And	(x>5 And x==10)	و
Or	(x>5 Or x==10)	یا
Xor	(x>5 Xor x==10)	یا نباشد
Not	(x>5 Xor x==10)	نباشد

```
#include <iostream>
int main()
{
    int a = 2, b = 3;

    if(a>0 or b==4)
        cout << "a:";
```

کد برنامه

```
return 0;
}
```

2

خروجی

Basic Input & Output

2-7- ورودی و خروجی مقدماتی داده

هدف از ساخت کامپیوتر و ایجاد برنامه نویسی دادن اطلاعات به ماشین و دریافت جواب بوده که به این روال، جریان ورودی خروجی گفته میشود.

برای استفاده از کلمات کلیدی `cin` و `cout` لازمه که از کتابخانه `iostream.h` در برنامه خود استفاده کنیم .

دستور `cout` باعث چاپ اطلاعات در مانیتور می شود به این صورت که اگر بعد از این دستور، عبارت مورد نظر رو تو " " قرار بدیم عینا همون عبارت تو مانیتور نشان داده می شود و معمولا جایی که کاربر قصد نشان دادن عین یک عبارت را در خروجی دارد مورد استفاده قرار می گیرد و اگر بخواهیم مقدار یک متغیر یا ثابتی را نمایش بدهیم باید نام را نوشته و از " " استفاده نکنیم . یک عبارت را در خروجی دارد مورد استفاده قرار می گیرد و اگر بخواهیم مقدار یک متغیر یا ثابتی را نمایش بدهیم باید نام را نوشته و از " " استفاده نکنیم .

```
#include <iostream.h>
#include <conio.h>
int main()
{
    int a=5;      //a=?
    cout << "Test"; //Prints a: on screen
    cout << a;     //Prints value of a
    getch();
    return 0;
}
```

کد برنامه

Test

5

خروجی

دستور `cin` باعث می شود که از کاربر یا برنامه دیگه ای یک مقدار یا متنی را بگیریم . در واقع کامپایلر با رسیدن به این دستور منتظر ورود اطلاعات از طرف کاربر شده تا زمانی که کاربر `Enter` را فشار بده و مقدار ورودی کاربر را در متغیری که در جلوش اعلان شده بریزه و همونطور که قبلا اشاره شد موجب مقدار دهی به متغیر مربوطه میشود.

```
#include <iostream.h>
#include <conio.h>
int main()
{
    int a, b;           //a=?,b=?
    cout << "Please inter value of a:";
    cin >> a;           //a=value of user input
    cout << "Please inter value of b:";
    cin >> b;           //b=value of user input
    return 0;
}
```

```
Please inter value of a:
    Input a
Please inter value of b:
    Input b
```

Program Structure

3- ساختار برنامه

Control Structure

3-1- ساختار کنترلی

بطور کلی، در هر برنامه نویسی اجرای دستورات از اولین سطر شروع شده و به ترتیب تا آخرین سطر ادامه میابد .

اما گاهی وقتها لازم است که یک دستور چندین بار تکرار شود و یا اینکه تحت شرایط خاصی اجرا گردد و یا از اجرای آن جلوگیری شود .

ساختارهای کنترلی به برنامه نویس این اجازه را می دهند که بر روی دستورات کنترل داشته باشد و آنها را تکرار، اجرا و متوقف سازد.

Conditional Structure :

ساختارهای شرطی

- If
- Switch

Loops Structure :

ساختارهای حلقه

- While
- Do – while
- For

ساختار اول If : در این حالت یک شرط در صورت درست بودن ، دستورات داخل شرط اجرا میشود.

If(شرط)

{

دستورات در صورت برقراری شرط

}

```
#include <iostream>
int main()
{
    int a = 2;

    if (a==2)
    {
        cout << "True";
    }

    return 0;
}
```

کد برنامه

True

خروجی

ساختار دوم if: در این حالت یک شرط در صورت درست بودن، دستورات قسمت بالایی داخل شرط اجرا میشود و در صورت نادرست بودن شرط، دستورات پایینی داخل آکولاد (قسمت else) اجرا میشود.

If (شرط)

```
{
    دستورات در صورت برقراری شرط
}
else
{
    دستورات در صورت برقرار نبودن شرط
}
```

```
#include <iostream>
int main()
{
    int a = 33;
```

کد برنامه

```

if (a==2)
{
    cout << "True";
}
Else
{
    cout << "False";
}
return 0;
}

```

False

خروجی

ساختار سوم if: در این حالت چند شرط تک به تک بررسی میشود و در صورت درست بودن هریک دستورات داخل آن شرط اجرا میشود و در آخر اگر هیچ شرط درست نبود دستورات داخل else پایانی اجرا میشود.

if (شرط_اول)

{

دستورات در صورت برقراری شرط

}

else if(شرط_دوم)

{

دستورات در صورت برقراری شرط

}

else if(شرط_سوم)

{

دستورات در صورت برقراری شرط

```
}  
else  
{
```

دستورات در صورت برقراری هیچ یک از شرط ها

```
}
```

```
#include <iostream>  
int main()  
{  
    int a = 33;  
  
    if (a==2)  
    {  
        cout << "1";  
    }  
    Else if(a<0)  
    {  
        cout << "2";  
    }  
    Else if(a>100)  
    {  
        cout << "2";  
    }  
    Else  
    {  
        cout << "else";  
    }  
  
    return 0;  
}
```

کد برنامه

else

خروجی

نکته : اگر فقط یک دستور بیشتر زیر شرط وجود ندارد ، میشود از آکولاد استفاده نکرد.

if (شرط)

دستورات در صورت برقراری شرط

ساختار switch: زمانی کاربرد دارد که تمام شرط های if و else برابر با تساوی باشد. داخل case ها شرط های در صورت برقرار بودن با متغیر اجرا میشود و در آخر اگر هر هیچیک از case ها درست نبود default اجرا میشود.

switch (متغیر شرط)

{

Case مقدار_اول :

دستورات در صورت برقراری شرط اول

Break;

Case مقدار_دوم :

دستورات در صورت برقراری شرط دوم

Break;

Case مقدار_سوم :

دستورات در صورت برقراری شرط سوم

Break;

Default:

دستورات در صورت برقرار نبودن هیچیک از case ها

Break;

کد برنامه

```
#include <iostream>
int main()
{
    int a = 1;
    switch(a)
    {
        Case 1:
            Cout<<"True1"
            Break;
        Case 2:
            Cout<<"True2"
            Break;
        Default:
            Cout<<"False";
    }
```

True1

خروجی

ساختار while: یک حلقه میشود ایجاد کرد که تا زمان درست بودن ، آن حلقه اجرا میشود.

while (شرط اجرای حلقه)

{

دستورات در صورت برقراری حلقه

}

کد برنامه

```
#include <iostream>
int main()
{
    int a = 0;
    while(a<=5)
    {
        Cout<<"Hello \n";
        a++;
    }
```

خروجی

Hello
Hello
Hello
Hello
Hello

ساختار do—while: یک حلقه میشود ایجاد کرد که اول بدنه اجرا بعد شرط اجرا میشود.

نکته: در این حلقه حتی اگر شرط اشتباه باشد، یک دور از حلقه اجرا میشود.

do

{

دستورات در صورت برقراری حلقه

}while(شرط اجرای حلقه)

کد برنامه

```
#include <iostream>
int main()
{
    int a = 0;

    do
    {
        Cout<<"Hello \n";
        a++;
    }while(a<=5)
```

خروجی

Hello
Hello
Hello
Hello
Hello

ساختار for : یک نوع حلقه که داری ظاهر بهتر ، شمارنده حلقه لوکال (محلّی) و از تداخل شمارنده جلوگیری میکند.

(افزایش شمارنده ; شرط اجرای حلقه ; تعریف شمارنده) For

{

دستورات در صورت برقراری حلقه

}

```
#include <iostream>
int main()
{
    int a = 0;

    for(int i=0;i<=5;i++)
    {
        Cout << "Hello \n";
    }
}
```

کد برنامه

Hello

Hello

Hello

Hello

Hello

خروجی

Function(I)

توابع 3-2-

وقتی خطوط برنامه ما زیاد می شود درک، پیگیری، خطایابی و دیگر اعمال بر روی برنامه دشوار خواهد شد. توابع ابزاری هستند که به ما در بهبود برنامه کمک می کنند و برنامه نویسی ساخت یافته را ارائه می دهند، بدین معنا که برنامه اصلی به قسمتهای منطقی و مستقل کوچکتری تقسیم می شود که توابع نام دارند .

برای بکارگیری توابع به سه جزء نیازمندیم: تعریف تابع، اعلان تابع، فراخوانی تابع

تعریف تابع <=

(لیست_پارامترها) نام_تابع نوع_بازگشتی

{

بدنه_تابع

}

```
#include <iostream>
int main()
{
    int a = 0;

    int Test(int x, int y)
    {
        Cout << x+y ;
    }
}
```

کد برنامه

اعلان یا الگوی تابع: در برنامه نویسی برای اینکه به کامپایلر وجود تابع یا توابعی را اطلاع دهیم باید آنرا اعلان کنیم که اینکار را قبل از تابع main و بعد از فایل‌های سرآیند برنامه انجام خواهیم داد.

;(نوع پارامترها) اسم تابع نوع_تابع

```
#include <iostream>
Int Test(int,int);
int main()
{
    int a = 0;

    int Test(int x, int y)
```

کد برنامه

```

{
    Cout << x+y ;
}
}

```

فراخوانی تابع: در نهایت باید در درون برنامه خود، تابع را صدا بزنیم که به اینکار فراخوانی تابع گفته می شود .

اسم-تابع (پارامترهای ورودی) ;

```

#include <iostream>
Int Test(int,int);
int main()
{
    int a = 0;

    int Test(int x, int y)
    {
        Cout << x+y ;
    }

    Test(2,4);
}

```

کد برنامه

6

خروجی

Function(II)

3-3- توابع

Recursive Functions

توابع بازگشتی

اگر یک تابع در هین اجرا خودش را فراخوانی کند ، به این تابع (تابع بازگشتی) گفته میشود.

مثال : یک تابع بازگشتی برای محاسبه فاکتوریل

```
#include <iostream>
Int Test(int,int);
int main()
{
    Int fact(int x)
    {
        If (x==1)
            Return 1;
        Else
            Return x * fact (x-1);
    }
}
```

کد برنامه

Compund Data Types

4- انواع ترکیب داده ها

Array

4-1- آرایه

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 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 .

نکته : - آرایه های زبان C در اصطلاح Zero - Based هستند. یعنی اندیس آنها از 0 شروع میشود.

- به هر یک از خانه های آرایه در اصطلاح یک Element یا عنصر یا سلول گفته میشود

نوع اول تعریف آرایه : مشخص کردن طول آرایه و بدون مقداردهی

نوع_آرایه اسم_آرایه [طول_آرایه];

```
#include <iostream>
int main()
{
    Int Test [20];
}
```

کد برنامه

نوع دوم تعریف آرایه : مشخص کردن طول آرایه و مقداردهی به آنها

{مقادیر_هر_اندیس} = [طول_آرایه] اسم_آرایه نوع_آرایه

```
#include <iostream>
int main()
{
    Int Test [5] = {1,2,12,6,23}
}
```

کد برنامه

نوع سوم تعریف آرایه : مشخص کردن طول آرایه و مقدار دهی به آنها در طول برنامه

```
#include <iostream>
int main()
{
    Int Test [20];

    Test[0]=20;
    Test[1]=43;
}
```

کد برنامه

Pointers

4-2- اشاره گر ها

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 8 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 2 یا 4 بایت را اشغال می کند و به این معنی است که تعداد 4 تا 8 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 بگم که چیز خوبی و سرعت، کارایی، دسترسی و ... را بالا می برد.

```
#include <iostream>
int main()
{
    Type *Name;

    int *P;
}
```

کد برنامه

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 .

قت 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 متغیرهایی از نوع `double` است نمی تواند `int` یا `char` یا هر چیز دیگری غیر از همان `double` باشد .

تعریف نوع داده های سفارشی : یک نوع داده سفارشی که مقادیری با انواع متفاوت را در خود نگه میدارد.

Data Structure

نوع داده سفارشی

نوع داده های سفارشی :

نام_ساختمان Struct

```
{  
    نام_عضو_اول      نوع_داده_اول  
    نام_عضو_دوم      نوع_داده_دوم  
    نام_عضو_سوم      نوع_داده_سوم  
}
```

نام شی سوم , نام شی دوم , نام شی اول}

```
#include <iostream>          کد برنامه  
int main()  
{  
    Struct human  
    {  
        String  Fname;  
        String  Lname;  
        Int      Age;  
        bool     Gender;  
    }h1,h2,h3,h4  
}
```

دسترسی به نوع داده سفارشی :

; نام_عضو . نام_شئ

Cin >> h1.Fname;

Cout << h1.Fname;

```
#include <iostream>
int main()
{
    Struct human
    {
        String    Fname;
        String    Lname;
        Int       Age;
        bool      Gender;
    }h1,h2,h3,h4

    Cout << "Please Enter a FirsName : ";
    Cin >> h1.Fname;
    Cout << "Hello",h1.Fname;
}
```

کد برنامه

Other Data Types

دیگر ساختار داده -4-4

نوع داده های سفارشی :

; داده_جدید_جایگزین نوع_داده Typedef

Typedef char C ;

char یعنی C

```
#include <iostream>
int main()
{
    Typedef char C;

    C Name;
}
```

کد برنامه

تعریف : یک مخزن را با نام داده های مختلف و اسم های مختلف به او اشاره می کنیم.

Union یک_نام_دلخواه

```
{  
    ; اسم_خاص_اول      نوع_داده_اول  
    ; اسم_خاص_دوم      نوع_داده_دوم  
    ; اسم_خاص_سوم      نوع_داده_سوم  
}
```

Union Number

```
{  
    char c;  
    int i;  
    float f;  
} mytypes
```

```
#include <iostream>  
int main()  
{  
    Union Number  
    {  
        Char c;  
        Int I;  
        Float t;  
    }  
}
```

کد برنامه

دسترسی :

; اسم_داده_اول نام_دلخواه

Cin >> Number.C;

Cout << Number.C;

```
#include <iostream>
int main()
{
    Union Number
    {
        Char C;
        Int I;
        Float T;
    }

    Cout << "Please Enter a Number : ";
    Cin >> Number.C;
    Cout << "Hello", Number.C;
}
```

کد برنامه

نکته : فضایی که Union اشغال میکند به اندازه بزرگترین عنصر تعریف شده در آن است .

Enumerations (enum)

یک لیست از مقادیر

تعریف : با Enum میتوانیم چک کنیم داده ای که وارد میشود داخل لیست هست یا نه .

Enum {مقدار_سوم , مقدار_دوم , مقدار_اول} نام_دلخواه

enum colors{black, blue, green, cyan, red, purple, yellow, white};

دسترسی :

کد برنامه

```
#include <iostream>
int main()
{
    enum colors{black, blue, green, cyan, red, purple, yellow, white};
    if(colors == red)
    {
        Cout << "Hello", Number.C;
    }
}
```

Class

5- کلاس

کلاس ها همان نوع داده تجریدی یا انتزاعی (Abstract) هستند. به عبارتی یک نوع داده هستند که توسط برنامه نویس برای کار با داده ها و توابع و رویدادهای مختلفی تعریف می شوند .

- کلاس ها قالبی هستند برای اشیاء .
- کلاس یک نوع داده تعریف شده توسط کاربر است .
- کلاس یک نوع داده پیشرفته است .

تعریف :

نام_کلاس Class

{

بدنه_کلاس

}