فصل چهارم – آشنایی با C/C++

C History

- BCPL ,1967, Martin Richards
  - writing operating-systems software and compiler
- B, 1969, Ken Thomson
  - based on BCPL
- C, 1972, Dennis Ritchie
  - based on BCPL and B
  - at Bell Laboratories
  - originally implemented on a DEC PDP-11

C History

- In 1983, the American National Standards Institute (ANSI) established a committee to provide a modern, comprehensive definition of C. The resulting definition, the ANSI standard, or ANSI C, was completed late 1988
  - updated in 1999

- Because C is a hardware-independent, widely available language, applications written in C can run with little or no modifications on a wide range of different computer systems
  - Portable programs

C++ Programming Language

- early 1980s, Bjarne Stroustrup
  - at Bell Labroratory
  - C++ a superset of C
  - object-oriented programming
    - Objects are essentially reusable software components that model items in the real world

- *filename.c*
- *filename.cpp*

C++ چیست

C++ یک زبان برنامه نویسی است که از paradigm یا سبک‌های مختلف برنامه نویسی پشتیبانی می‌کند.C++ نسخه توسعه یافته زبان C می‌باشد و بیشتر کدهای زبان C به راحتی می‌تواند در C++ کامپایل شود. در C++ از ویژگی‌های مهمی که به C اضافه شده است می‌توان به برنامه نویسی شی گرا ، سربارگذاری عملگرها، وراثت چندگانه و مدیریت خطاها اشاره نمود. توسعه C++ در سال ۱۹۷۹ آغاز شد و ۷ سال پس از زبان C به نمایش گذاشته شد.با وجود قدیمی بودن زبان‌های C و c++ هنوز هم به صورت گسترده ای در نرم افزارهای صنعتی مورد استفاده قرار می‌گیرد. این زبان‌ها برای ساخت هر چیزی از سیستم عامل گرفته تا نرم افزارهای توکار ، برنامه‌های دسکتاپ و بازی‌ها مورد استفاده قرار می‌گیرد .

در مقایسه با زبان‌های جدید تر ، برنامه‌های نوشته شده با C++ اغلب پیچیده تر می‌باشند و زمان بیشتری برای توسعه نیاز دارد.در عوض، C++زبانی است که به شما اجازه می‌دهد که هم به صورت High-level (نزدیک به زبان انسان) و هم به صورت low-level (نزدیک به زبان ماشین) سخت افزار را تحت کنترل خود قرار دهید.همچنین با پشتیبانی از سبک‌ها یا پارادایم‌های مختلف برنامه نویسی از جمله رویه ای ، شی گرا یا عمومی ، دست برنامه نویس را در انتخاب سبک مورد نظرش آزاد می‌گذارد.

معرفی ساختاری زبان C++

C++ عموماً از سه بخش تشکیل شده است:

- محیطی برای نوشتن برنامه و ویرایش آن.
- کامپایلر C++.
- کتابخانه استاندارد C++.

یک برنامه زبان C++ برای رسیدن به مرحله اجرا از شش مرحله عبور می کند.

مرحله اول : برنامه نویس، برنامه را در محیط ویرایشگر می نویسد و آن را بر روی دیسک ذخیره می کند.

مرحله دوم : برنامه پیش پردازنده، خطوط برنامه را از لحاظ ایرادات نگارشی بررسی می کند، و در صورت وجود اشکال در برنامه پیغام خطائی داده می شود، تا برنامه نویس نسبت به رفع آن اقدام نماید.

مرحله سوم : کامپایلر، برنامه را به زبان ماشین ترجمه می کند و آن را بر روی دیسک ذخیره می نماید.

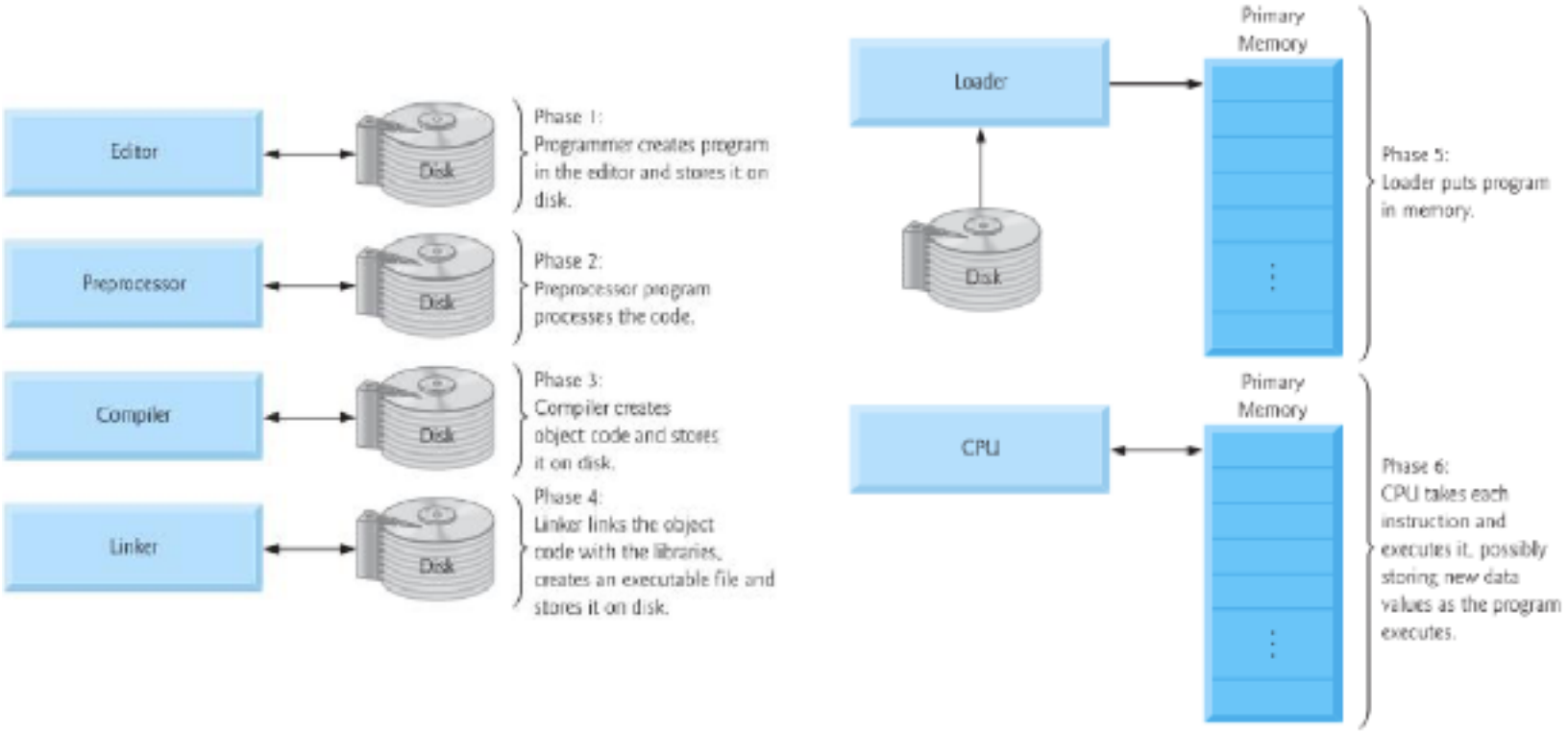
مرحله چهارم : پیوند دهنده، کدهای زبان ماشین را، به فایلهای کتابخانه هایی که مورد استفاده قرار گرفته اند پیوند می دهد و یک فایل قابل اجرا بر روی دیسک می سازد.

مرحله پنجم : بار کننده برنامه را در حافظه قرار می دهد.

مرحله ششم : واحد پردازش مرکزی کامپیوتر دستورات برنامه را اجرا می کند.

C programs typically go through six phases to be executed:

– edit, preprocess, compile, link, load and execute

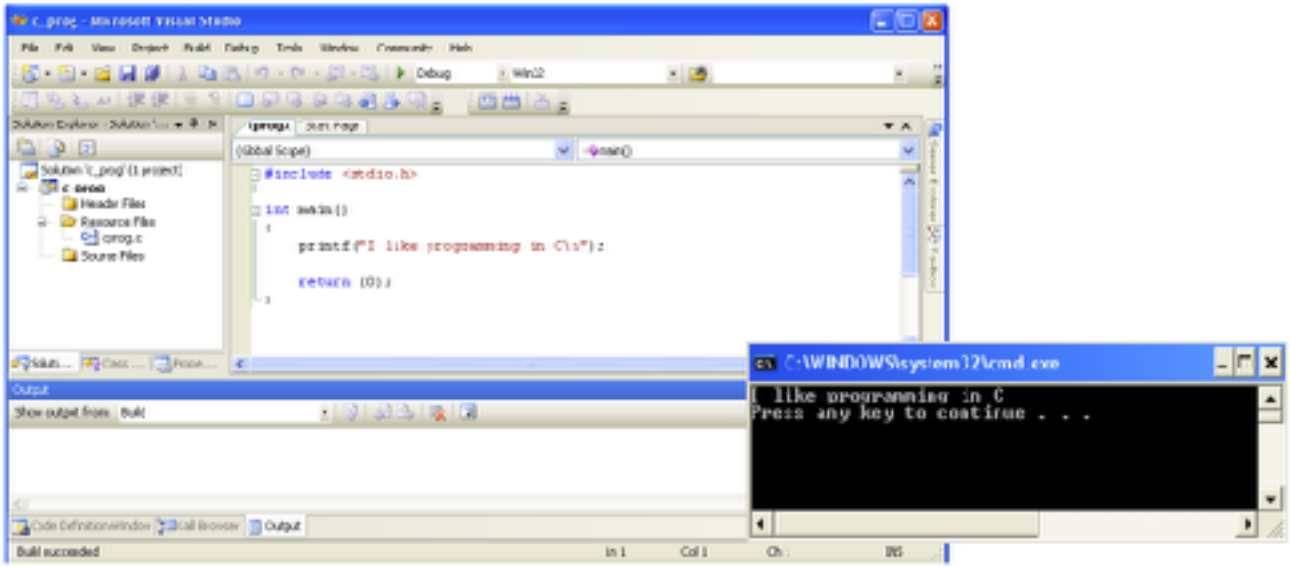


برای اجرای کدهای C++ نیاز به یک کامپایلر داریم. کامپایلرها و محیط‌های برنامه نویسی (IDE) گوناگونی برای زبان C++ وجود دارند از بین معروف‌ترین آن‌ها می‌توان موارد زیر اشاره نمود:

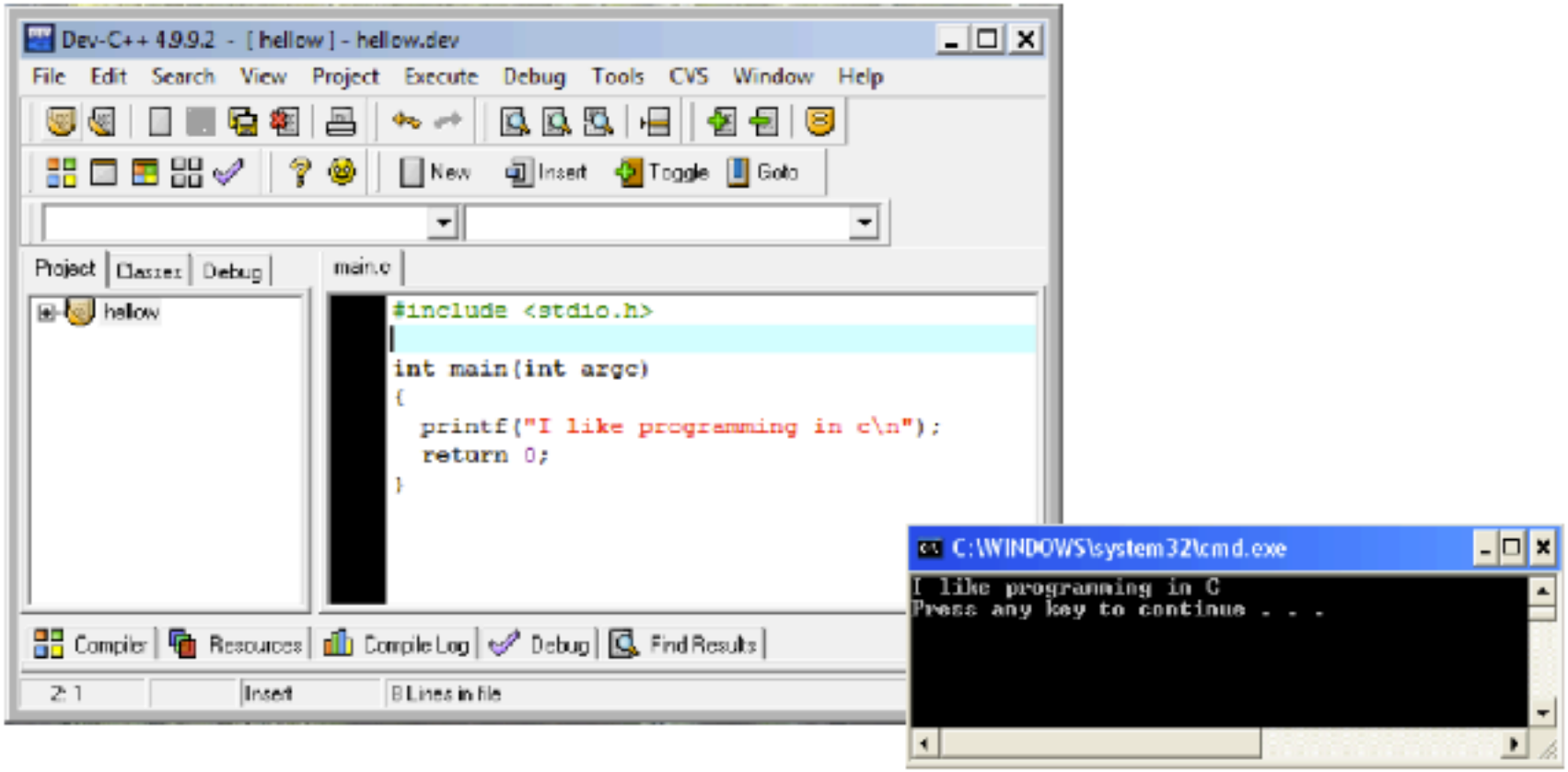
- Turbo C
- Turbo C++
- Borland C++
- Microsoft C++/C

Microsoft Visual Studio

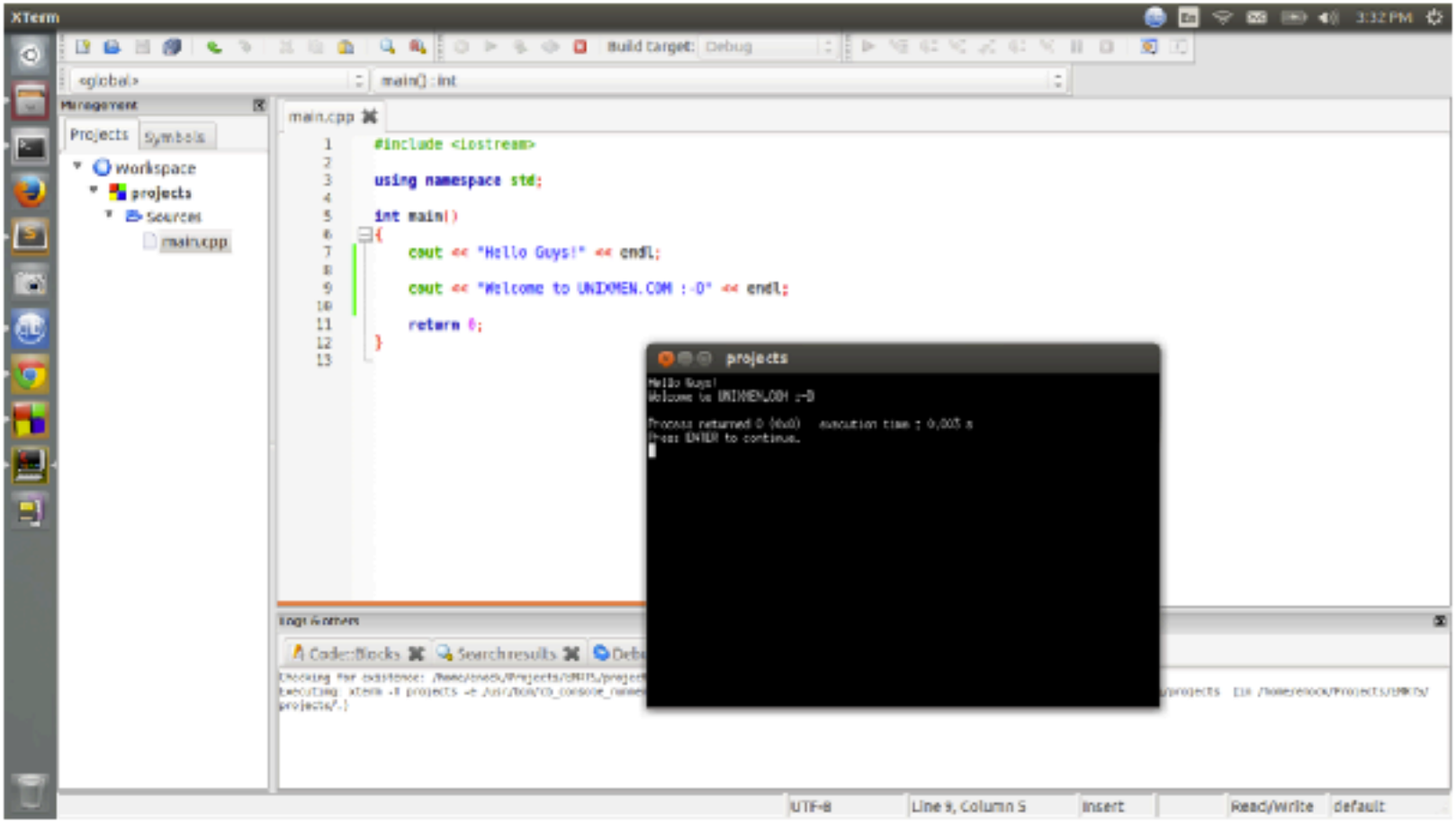
- Editing a file with an editor program
- Integrated Development Environment (IDE)



Dev-C++


Dev-C++ 4.9.9.2 - [hellow] - hellow.dev
File Edit Search View Project Execute Debug Tools CVS Window Help
New Insert Toggle Goto
Project Classes Debug
hellow
main.c
#include <stdio.h>
int main(int argc)
{
printf("I like programming in c\n");
return 0;
}
Compiler Resources Compile Log Debug Find Results
2: 1 Insert 8 Lines in file
C:\WINDOWS\system32\cmd.exe
I like programming in C
Press any key to continue . . .

Code::Blocks


XTerm
Build target: Debug
main() : int
Management
Projects
Symbols
workspace
projects
Sources
main.cpp
#include <iostream>
using namespace std;
int main()
{
cout << "Hello Guys!" << endl;
cout << "Welcome to UNIXMEN.COM :-D" << endl;
return 0;
}
projects
Hello Guys!
Welcome to UNIXMEN.COM :-D
Process returned 0 (0x0) execution time : 0.005 s
Press ENTER to continue.
Code::Blocks
Search results
UTF-8
Line 9, Column 5
Insert
Read/Write
default

ویژگی های زبان C

۱- زبان میانه است :
سطح میانه بودن این زبان به این معنا است که این زبان امکانات و قدرت بان های سطح پایین را دارد وهمچنین عناصر زبان های سطح بالا را نیز پشتیبانی می کند.
۲- زبان C دارای قابلیت حمل یا Portability می باشد یعنی برنامه نوشته شده با این زبان بر روی کامپیوترهای مختلف بدون تغییر کد منبع قابل کامپایل می باشد.
۳- Cبرای نوشتن برنامه های سیستمی به کار می رود مانند سیستم عامل ،کامپایلر، اسمبلر و ...
۴- در زبان C بین حروف کوچک و بزرگ اختلاف وجود دارد.
۵- انتهای هر خط ; می گذاریم.
۶- تعداد کلمات کلیدی آن انگشت شمار است.

C++ که از نسل C است، تمام ویژگی‌های جذاب بالا را به ارث برده است. این فرزند اما برتری فنی دیگری هم دارد: C++ اکنون «شی‌گرا» است. می‌توان با استفاده از این خاصیت، برنامه‌های شی‌گرا تولید نمود. برنامه‌های شی‌گرا منظم و ساخت‌یافته‌اند، قابل روزآمد کردن‌اند، به سهولت تغییر و بهبود می‌یابند و قابلیت اطمینان و پایداری بیشتری دارند.

Simple Program

- Examples:

```
// int is return data type
// main is entrance function
int main()
{
    statement 1;
    statement 1;
    // ....
    return 0;
}
```

```
// Simplest c program
int main()
{
    return 0;
}
```

```
/*
   Objective:  print on screen
*/
#include <stdio.h> // preprocessor  statements have not ;
int main()
{
    printf("welcome to c!!!");
    return 0; // indicate that program ended successfully
}
```

```
welcome to c!!!
```

- Examples:
  - Header file
  - Main function
  - Variables
  - Input and output
  - Process

```
#include <stdio.h>  // header file (preprocessor )
//  calculating sum of two  user input  variables
int main()
{
   /* variable definition */
   int a;
   int b;
   int result = 0;
   // get first variables form user
   printf("Enter first number:\n");
   scanf("%d", &a);
   // get scoend variables form user
   printf("Enter scoend number:\n");
   scanf("%d", &b);
   //  sum of input variables
   result = a + b;
   printf("%d + %d = %d\n", a, b, result);
   system("Pause");
   return 0;
}
```

- Examples:
  - Header file
  - Constant
  - Main function
  - Variables
  - Input and output
  - Process

```
#include <stdio.h>  // (preprocessor )

#define  PI   3.14  // PI  constant (preprocessor )

//  calculating  area of circle
int main()
{
   /* variable definition */
   float  Radius;
   float  Area = 0;
   // get radius of circle form user
   printf("Enter Radius :\n");
   scanf("%f",  &Radius);
   // calculating  area of circle
   Area  =  PI  *  Radius  *  Radius;
   printf("Area  =  %f",  Area );

   system("Pause");
   return 0;
}
```

اولین برنامه ای که می نویسیم پیغام Hello World! را نمایش می دهد

```
1  #include <iostream>
2  using namespace std;
3
4  int main()
5  {
6      // This line will print the message hello world
7      cout << "Hello World!";
8  }
```

خط ۱ : اولین خط از کد بالا یک «راهنمای پیش‌پردازنده[2]» است. راهنمای پیش‌پردازنده شامل اجزای زیر است:

1 – کاراکتر # که نشان می‌دهد این خط، یک راهنمای پیش‌پردازنده است. این کاراکتر باید در ابتدای هم‌ٔ خطوط راهنمای پیش‌پردازنده باشد.

2 – عبارت include

3 – نام یک «فایل کتابخانه‌ای» که میان دو علامت <> محصور شده است. به فایل کتابخانه‌ای «سرفایل[3]» نیز می‌گویند. فایل کتابخانه‌ای که در این‌جا استفاده شده نام دارد. iostream

خط ۲ : دومین خط از برنامهٔ بالا یعنی

```
using namespace std;
```

به کامپایلر می‌گوید که عبارت std:: را در سراسر برنامه در نظر داشته باشد تا مجبور نباشیم برای دستوراتی مثل cout این پیشوند را به کار ببریم. به این طریق می‌توانیم برای دستور خروجی به جای std::cout از عبارت cout تنها استفاده نماییم

خط ۴ : این خط به

```
4  int main()
```

کامپایلر می‌گوید که «بدنهٔ اصلی برنامه» از کجا شروع می‌شود. این خط دارای اجزای زیر است:

1 – عبارت int که یک نوع عددی در C++ است.

2 – عبارت main که به آن «تابع اصلی» در C++ می‌گویند.

3 – دو پرانتز () که نشان می‌دهد عبارت main یک «تابع[1]» است.

هر برنامه فقط باید یک تابع main() داشته باشد. وقتی برنامه اجرا شد، یک عدد صحیح به سیستم‌عامل بازگردانده می‌شود تا سیستم‌عامل بفهمد که برنامه با موفقیت به پایان رسیده یا خیر. عبارت int که قبل از main استفاده شده نشان می‌دهد که این برنامه یک عدد صحیح را به سیستم عامل برمی‌گرداند.

همیشه کدهای خود را در داخل آکولاد بنویسید.

```
{
   statement1;
}
```

خط ۶ : خط ٦ یک توضیح درباره خط ۷ است که به کاربر اعلام می‌کند که وظیفه خط ۷ چیست؟ با اجرای کد بالا فقط جمله Hello Worldچاپ شده و خط ۷ در خروجی نمایش داده نمی‌شود چون کامپایلر توضیحات را نادیده می‌گیرد . توضیحات بر دو نوع‌اند:

توضیحات تک خطی

```
// single line comment
```

توضیحات چند خطی

```
/* multi
     line
comment */
```

توضیحات تک خطی همانگونه که از نامش پیداست، برای توضیحاتی در حد یک خط به کار می‌روند. این توضیحات با علامت // شروع می‌شوند و هر نوشته ای که در سمت راست آن قرار بگیرد جز توضیحات به حساب می‌آید. این نوع توضیحات معمولاً در بالا یا کنار کد قرار می‌گیرند. اگر توضیح درباره یک کد به بیش از یک خط نیاز باشد از توضیحات چند خطی استفاده می‌شود. توضیحات چند خطی با /*شروع و با */ پایان می‌یابند. هر نوشته ای که بین این دو علامت قرار بگیرد جز توضیحات محسوب می‌شود.

خط ۷ : بدنه اصلی برنامه مجموعه ای از دستورات متوالی است که میان دو علامت{} محصور شده است

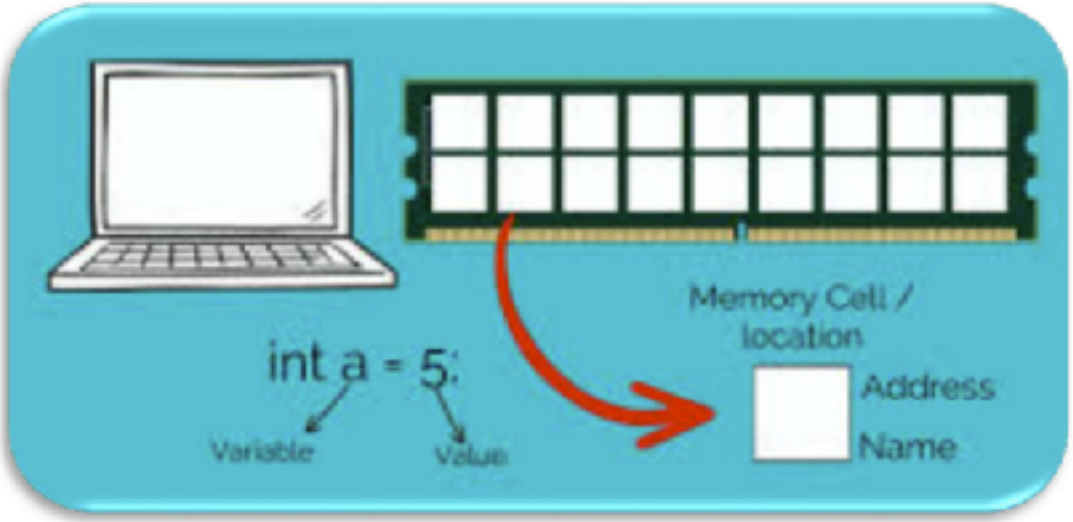
این برنامه فقط یک دستور دارد

```
7       cout << "Hello World!";
```

این دستور عبارت Hello World! را در صفحه نمایش چاپ می کند . حتما در پایان هر دستور علامت ; را قرار دهید در غیر این صورت کامپایلر از شما خطا می گیرد.

متغیر

متغیر مکانی از حافظه است که شما می‌توانید مقادیری را در آن ذخیره کنید.می‌توان آن را به عنوان یک ظرف تصور کرد که داده‌های خود را در آن قرار داده‌اید.محتویات این ظرف می‌تواند پاک شود یا تغییر کند.هر متغیر دارای یک نام نیز هست.که از طریق آن می‌توان متغیر را از دیگر متغیرها تشخیص داد و به مقدار آن دسترسی پیدا کرد.همچنین دارای یک مقدار می‌باشد که می‌تواند توسط کاربر انتخاب شده باشد یا نتیجه یک محاسبه باشد.مقدار متغیر می‌تواند تهی نیز باشد.متغیر دارای نوع نیز هست بدین معنی که نوع آن با نوع داده ای که در آن ذخیره می‌شود یکی است.



برای نامگذاری متغیرها باید قوانین زیر را رعایت کرد:

- نام متغیر باید با یک از حروف الفبا (a-z or A-Z) شروع شود.
- نمی‌تواند شامل کاراکترهای غیرمجاز مانند $. ,^ ,? ,#باشد.

- نمی‌توان از کلمات رزرو شده در C++ برای نام متغیر استفاده کرد.
- نام متغیر نباید دارای فضای خالی (spaces) باشد.
- اسامی متغیرها نسبت به بزرگی و کوچکی حروف حساس هستند. در C++ دو حرف مانند a و Aدو کاراکتر مختلف به حساب می‌آیند.

متغیر یا variableدارای نوع است در جدول زیر انواع داده هایی که C++پشتیبانی می کند مشخص شده است:

کلمه کلیدی	نوع
bool	Boolean
char	Character
int	Integer
float	Floating point
double	Double floating point
void	Valueless

انواع بالا (به جز void) می‌توانند با عباراتی مثل signed ، long، unsigned و short ترکیب شده و نوع‌های دیگری را به وجود آورند:

نوع	مقدار فضایی که از حافظه اشغال می‌کند	محدوده
char	1byte	-128 to 127 or 0 to 255
unsigned char	1byte	0 to 255
signed char	1byte	-128 to 127
int	4bytes	-2147483648 to 2147483647
unsigned int	4bytes	0 to 4294967295
signed int	4bytes	-2147483648 to 2147483647
short int	2bytes	-32768 to 32767
unsigned short int	2bytes	0 to 65,535
signed short int	2bytes	-32768 to 32767
long int	8bytes	-9,223,372,036,854,775,808 to 9,223,372,036,854,775,807

signed long int	8bytes	-9,223,372,036,854,775,808 to 9,223,372,036,854,775,807
unsigned long int	8bytes	0 to 18,446,744,073,709,551,615
float	4bytes	+/- 3.4e +/- 38 (~7 digits)
double	8bytes	+/- 1.7e +/- 308 (~15 digits)
long double	8bytes	+/- 1.7e +/- 308 (~15 digits)
wchar_t	2 or 4 bytes	1 wide character

نوع char برای ذخیره کاراکترهای یونیکد استفاده می‌شود.کاراکترها باید داخل یک کوتیشن ساده قرار بگیرند مانند.('a')

نوع bool فقط می‌تواند مقادیر درست (true) یا نادرست (false) را در خود ذخیره کند و بیشتر در برنامه‌هایی که دارای ساختار تصمیم گیری هستند مورد استفاده قرار می‌گیرد.

نوع stringبرای ذخیره گروهی از کاراکترها مانند یک پیغام استفاده می‌شود. مقادیر ذخیره شده در یک رشته باید داخل دابل کوتیشن قرار گیرند تا توسط کامپایلر به عنوان یک رشته در نظر گرفته شوند. مانند("massage")

در مثال زیر نحوه تعریف و استفاده از متغیرها نشان داده شده است:

```cpp
#include <iostream>
using namespace std;

int main()
{
    //Declare variables
    int num1;
    int num2;
    double num3;
    double num4;
    bool boolVal;
    char myChar;
    string message;

    //Assign values to variables
    num1 = 1;
    num2 = 2;
    num3 = 3.54;
    num4 = 4.12;
    boolVal = true;
    myChar = 'R';
    message = "Hello World!";

    //Show the values of the variables
    cout << "num1 = " << num1 << endl;
    cout << "num2 = " << num2 << endl;
    cout << "num3 = " << num3 << endl;
    cout << "num4 = " << num4 << endl;
    cout << "boolVal = " << boolVal << endl;
    cout << "myChar = " << myChar << endl;
    cout << "message = " << message << endl;

}
```

تعریف متغیر

نحوه تعریف متغیر به صورت زیر است :

```
data_type identifier;
```

date_type همان نوع داده است مانند int ، double و .... dentifier نیز نام متغیر است که به ما امکان استفاده و دسترسی به مقدار متغیر را می‌دهد . برای تعریف چند متغیر از یک نوع می‌توان به صورت زیر عمل کرد:

```
data_type identifier1, identifier2, ... indentifierN;
```

مثال

```
int num1, num2, num3, num4, num5;
string message1, message2, message3;
```

در مثال بالا ۵ متغیر از نوع صحیح و ۳ متغیر از نوع رشته تعریف شده است.توجه داشته باشید که بین متغیرها باید علامت کاما (,) باشد.

نامگذاری متغیرها

- نام متغیر باید با یک حرف یا زیرخط و به دنبال آن حرف یا عدد شروع شود.
- نمی‌توان از کاراکترهای خاص مانند & ,% ,#یا عدد برای شروع نام متغیر استفاده کرد مانند .2numbers
- نام متغیر نباید دارای فاصله باشد.برای نام‌های چند حرفی می‌توان به جای فاصله از علامت زیرخط یا _استفاده کرد.

نامهای مجاز:

```
num1  myNumber  studentCount  total       first_name    _minimum
num2  myChar    average       amountDue   last_name     _maximum
name  counter   sum           isLeapYear  color_of_car  _age
```

نامهای غیر مجاز:

```
123         #numbers#  #ofstudents  1abc2
123abc      $money     first name   ty.np
my number   this&that  last name    1:00
```

محدوده متغیر

متغیرهای ابتدای درس در داخل متد main() تعریف شده‌اند. در نتیجه این متغیرها فقط در داخل متد main() قابل دسترسی هستند. محدوده یک متغیر مشخص می‌کند که متغیر در کجای کد قابل دسترسی است. هنگامیکه برنامه به پایان متد main() می‌رسد متغیرها از محدوده خارج و بدون استفاده می‌شوند تا زمانی که برنامه در حال اجراست.محدوده متغیرها انواعی دارد که در درس‌های بعدی با آن‌ها آشنا می‌شوید. تشخیص محدوده متغیر بسیار مهم است چون به وسیله آن می‌فهمید که در کجای کد می‌توان از متغیر استفاده کرد. باید یاد آور شد که دو متغیر در یک محدوده نمی‌توانند دارای نام یکسان باشند. مثلاً کد زیر در برنامه ایجاد خطا می‌کند:

```
int num1;
int num1;
```

از آنجاییکه C++ به بزرگی و کوچک بودن حروف حساس است می‌توان از این خاصیت برای تعریف چند متغیر هم نام ولی با حروف متفاوت (از لحاظ بزرگی و کوچکی) برای تعریف چند متغیر از یک نوع استفاده کرد مانند:

```
int num1;
int Num1;
int NUM1;
```

مقداردهی متغیرها

می‌توان فوراً بعد از تعریف متغیرها مقادیری را به آن‌ها اختصاص داد.این عمل را مقداردهی می‌نامند. در زیر نحوه مقدار دهی متغیرها نشان داده شده است:

```
data_type identifier = value;
```

به عنوان مثال:

```
int myNumber = 7;
```

همچنین می‌توان چندین متغیر را فقط با گذاشتن کاما بین آن‌ها به سادگی مقدار دهی کرد:

```
data_type variable1 = value1, varaible2 = value2, ... variableN, valueN;
int num1 = 1, num2 = 2, num3 = 3;
```

تعریف متغیر با مقدار دهی متغیرها متفاوت است. تعریف متغیر یعنی انخاب نوع و نام برای متغیر ولی مقدار دهی یعنی اختصاص یک مقدار به متغیر.

اختصاص مقدار به متغیر

در زیر نحوه اختصاص مقادیر به متغیرها نشان داده شده است:

```
num1 = 1;
num2 = 2;
num3 = 3.54;
num4 = 4.12;
boolVal = true;
myChar = 'R';
message = "Hello World!";
```

به این نکته توجه کنید که شما به مغیری که هنوز تعریف نشده نمی‌توانید مقدار بدهید. شما فقط می‌توانید از متغیرهایی استفاده کنید که هم تعریف و هم مقدار دهی شده باشند. مثلاً متغیرهای بالا همه قابل استفاده هستند. در این مثال num1 و num2 هر دو تعریف شده‌اند و مقادیری از نوع صحیح به آن‌ها اختصاص داده شده است.

ثابت

ثابت‌ها انواعی هستند که مقدار آن‌ها در طول برنامه تغییر نمی‌کند. ثابت‌ها حتماً باید مقدار دهی اولیه شوند و اگر مقدار دهی آن‌ها فراموش شود در برنامه خطا به وجود می‌آید. بعد از این که به ثابت‌ها مقدار اولیه اختصاص داده شد هرگز در زمان اجرای برنامه نمی‌توان آن را تغییر داد . برای تعریف ثابت‌ها باید از کلمه کلیدی const و #define استفاده کرد . معمولاً نام ثابت‌ها را طبق قرارداد با حروف بزرگ می‌نویسند تا تشخیص آن‌ها در برنامه راحت باشد. نحوه تعریف ثابت در زیر آمده است:

```
const data_type identifier = initial_value;
```

در کد بالا ابتدا کلمه کلیدی const و سپس نوع ثابت و بعد نام ثابت را با حروف بزرگ می‌نویسیم. و در نهایت یک مقدار را به آن اختصاص می‌دهیم و علامت سمیکالن می‌گذاریم.

```
#define data_type identifier initial_value
```

در روش بالا فقط #define را نوشته و سپس نام ثابت و بعد مقداری که قرار است دریافت کند. به این نکته توجه کنید که در روش بالا نه علامت سمیکالن وجود دارد و نه علامت مساوی. مثال:

```cpp
#include <iostream>
using namespace std;

int main()
{
    const int NUMBER = 1;

    NUMBER = 20; //ERROR, Cant modify a constant

    cout << NUMBER;
}
```

```cpp
#include <iostream>
using namespace std;
#define NUMBER 1
int main()
{


    NUMBER = 20; //ERROR, Cant modify a constant

    cout << NUMBER;
```

- Examples:
  - Header file
  - Constant
  - Main function
  - Variables
  - Input and output
  - Process

```
#include <stdio.h>  // (preprocessor )

#define  PI   3.14  // PI  constant (preprocessor )

//  calculating  area of circle
int main()
{
   /* variable definition */
   float  Radius;
   float  Area = 0;
   // get radius of circle form user
   printf("Enter Radius :\n");
   scanf("%f",  &float );
   // calculating  area of circle
   Area  =  PI  *  Radius  *  Radius;
   printf("Area  =  %f",  Area );

   system("Pause");
   return 0;
}
```

- Examples:
  - Header file
  - Constant
  - Main function
  - Variables
  - Input and output
  - Process

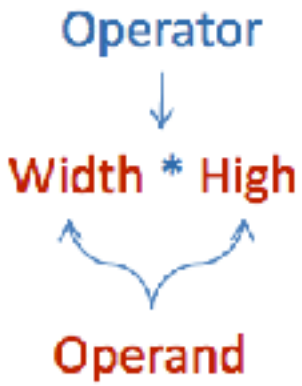
```
#include <stdio.h>  // (preprocessor )

#define  PI   3.14  // PI  constant (preprocessor )

// calculating  area of circle
int main()
{
   /* variable definition */
   float  Radius;
   float  Area = 0;
   // get radius of circle form user
   printf("Enter Radius :\n");
   scanf("%f",  &float );
   // calculating  area of circle
   Area  =  PI  *  Radius  *  Radius;
   printf("Area  =  %f",  Area );

   system("Pause");
   return 0;
}
```

عبارات و عملگرها



ابتدا با دو کلمه آشنا شوید:

- عملگر : نمادهایی هستند که اعمال خاص انجام می‌دهند.
- عملوند : مقادیری که عملگرها بر روی آن‌ها عملی انجام می‌دهند.

مثلاً X+Y: یک عبارت است که در آن X و Y عملوند و علامت + عملگر به حساب می‌آیند.

انواع مختلف عملگر که در این بخش مورد بحث قرار می‌گیرند عبارت‌اند از:

- عملگرهای ریاضی
- عملگرهای تخصیصی
- عملگرهای مقایسه ای
- عملگرهای منطقی
- عملگرهای بیتی

عملگرهای ریاضی

C++ از عملگرهای ریاضی برای انجام محاسبات استفاده می‌کند. جدول زیر عملگرهای ریاضی سی پلاس پلاس را نشان می‌دهد:

عملگر	دسته	مثال	نتیجه
+	Binary	var1 = var2 + var3;	Var1برابر است با حاصل جمع var2 وvar3
-	Binary	var1 = var2 - var3;	Var1برابر است با حاصل تفریق var2 وvar3
*	Binary	var1 = var2 * var3;	Var1برابر است با حاصلضرب var2 درvar3
/	Binary	var1 = var2 / var3;	Var1برابر است با حاصل تقسیم var2 برvar3
%	Binary	var1 = var2 % var3;	Var1برابر است با باقیمانده تقسیم var2 وvar3
+	Unary	var1 = +var2;	Var1برابر است با مقدار var2
-	Unary	var1 = -var2	Var1برابر است با مقدار var2 ضربدر ۱-

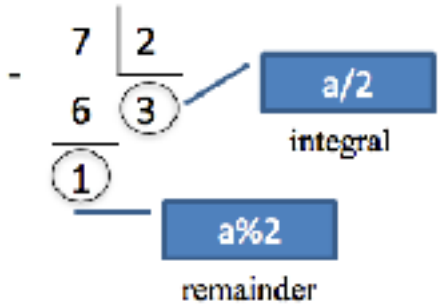
نکته : عملگر %یا باقی مانده فقط برای اعداد صحیح می تواند استفاده شود

- ☒ `z = a % 2.0; // error`
- ☒ `z = a % 0; // error`

- Example

```
int a = 7, b, c;
b = a % 2;
c = a / 2;
```

Modulus will result in the remainder of a/2.



دیگر عملگرهای C++ عملگرهای کاهش و افزایش هستند.این

عملگرها مقدار ۱ را از متغیرها کم یا به آن‌ها اضافه می‌کنند. از این متغیرها اغلب در حلقه‌ها استفاده می‌شود:

عملگر	دسته	مثال	نتیجه
++	Unary	var1 = ++var2;	مقدار var1 برابر است با var2 بعلاوه ۱
- -	Unary	var1 = - -var2;	مقدار var1 برابر است با var2 منهای ۱
++	Unary	var1 = var2++;	مقدار var1 برابر است با var2 به متغیر var2 یک واحد اضافه می‌شود
-	Unary	var1 = var2-;	مقدار var1 برابر است با var2 از متغیر var2 یک واحد کم می‌شود

به این نکته توجه داشته باشید که محل قرار گیری عملگر در نتیجه محاسبات تأثیر دارد. اگر عملگر قبل از متغیر var2 بیاید افزایش یا کاهش var1 اتفاق می‌افتد.چنانچه عملگرها بعد از متغیر var2 قرار بگیرند ابتدا var1 برابر var2می‌شود و سپس متغیر var2افزایش یا کاهش می‌یابد. به مثال‌های زیر توجه کنید:

```
#include <iostream>
using namespace std;

int main()
{
    int x = 0;
    int y = 1;

    x = ++y;

    cout << "x= {0}" << x << endl;
    cout << "y= {0}" << y << endl;
}
```

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

```
#include <iostream>
using namespace std;

int main()
{
    int x = 0;
    int y = 1;

    x = --y;

    cout << "x= {0}" << x << endl;
    cout << "y= {0}" << y << endl;
}
```

```
x=0
y=0
```

همانطور که در دو مثال بالا مشاهده می‌کنید،درج عملگرهای – و ++ قبل از عملوند y باعث می‌شود که ابتدا یک واحد از y کم و یا یک واحد به y اضافه شود و سپس نتیجه در عملوند x قرار بگیرد.

```cpp
#include <iostream>
using namespace std;

int main()
{
    int x = 0;
    int y = 1;

    x = y--;

    cout << "x= {0}" << x << endl;
    cout << "y= {0}" << y << endl;
}
```

```
x=1
y=0
```

```cpp
#include <iostream>
using namespace std;

int main()
{
    int x = 0;
    int y = 1;

    x = y++;

    cout << "x= {0}" << x << endl;
    cout << "y= {0}" << y << endl;
}
```

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

همانطور که در دو مثال بالا مشاهده می‌کنید،درج عملگرهای – و ++ بعد از عملوند y باعث می‌شود که ابتدا مقدار y در داخل متغیر x قرار بگیرد و سپس یک واحد از y کم و یا یک واحد به ان اضافه شود.

عملگرهای تخصیصی

نوع دیگر از عملگرهای C++ عملگرهای جایگزینی نام دارند. این عملگرها مقدار متغیر سمت راست خود را در متغیر سمت چپ قرار می‌دهند. جدول زیر انواع عملگرهای تخصیصی در سی شارپ را نشان می‌دهد:

عملگر	مثال	نتیجه
=	var1 = var2;	مقدار var1 برابر است با مقدار var2
+=	var1 += var2;	مقدار var1 برابر است با حاصل جمع var1 و var2
-=	var1 -= var2;	مقدار var1 برابر است با حاصل تفریق var1 و var2
*=	var1 *= var2;	مقدار var1 برابر است با حاصل ضرب var1 در var2
/=	var1 /= var2;	مقدار var1 برابر است با حاصل تقسیم var1 بر var2
%=	var1 %= var2;	مقدار var1 برابر است با باقیمانده تقسیم var1 بر var2

استفاده از این نوع عملگرها در واقع یک نوع خلاصه نویسی در کد است. مثلاً شکل اصلی کد var1 += var2 به صورت var1 = var1 + var2 می‌باشد. این حالت کدنویسی زمانی کارایی خود را نشان می‌دهد که نام متغیرها طولانی باشد.

عملگرهای مقایسه ای

از عملگرهای مقایسه ای برای مقایسه مقادیر استفاده می‌شود.نتیجه این مقادیر یک مقدار بولی(منطقی) است.این عملگرها اگر نتیجه مقایسه دو مقدار درست باشد مقدار ۱ و اگر نتیجه مقایسه اشتباه باشد مقدار ۰ را نشان می‌دهند. این عملگرها به طور معمول در دستورات شرطی به کار می‌روند به این ترتیب که باعث ادامه یا توقف دستور شرطی می‌شوند.جدول زیر عملگرهای مقایسه ای در C++ را نشان می‌دهد:

عملگر	دسته	مثال	نتیجه
==	Binary	var1 = var2 == var3	var1در صورتی ۱ است که مقدار var2 با مقدار var3 برابر باشد در غیر اینصورت ۰ است
!=	Binary	var1 = var2 != var3	var1در صورتی ۱ است که مقدار var2 با مقدار var3 برابر نباشد در غیر اینصورت ۰ است
<	Binary	var1 = var2 < var3	var1در صورتی ۱ است که مقدار var2 کوچک‌تر از var3 مقدار باشد در غیر اینصورت ۰ است

عملگرهای مقایسه ای

>	Binary	var1 = var2 > var3	var1در صورتی ۱ است که مقدارvar2 بزرگ‌تر ازمقدار var3باشد در غیر اینصورت ۰ است
<=	Binary	var1 = var2 <= var3	var1در صورتی ۱ است که مقدار var2 کوچک‌تر یا مساوی مقدار var3باشد در غیر اینصورت ۰ است
>=	Binary	var1 = var2 >= var3	var1در صورتی ۱ است که مقدار var2 بزرگ‌تر یا مساوی var3 مقدار باشد در غیر اینصورت ۰ است

عملگرهای منطقی

عملگرهای منطقی بر روی عبارات منطقی عمل می‌کنند و نتیجه آن‌ها نیز یک مقدار بولی است. از این عملگرها اغلب برای شرطهای پیچیده استفاده می‌شود. همانطور که قبلاً یاد گرفتید مقادیر بولی می‌توانند `false` یا `true` باشند. فرض کنید که `var2` و `var3`دو مقدار بولی هستند.

عملگر	نام	دسته	مثال
&&	منطقی AND	Binary	`var1 = var2 && var3;`
||	منطقی OR	Binary	`var1 = var2 || var3;`
!	منطقی NOT	Unary	`var1 = !var1;`

عملگر منطقی (&&)AND

اگر مقادیر دو طرف عملگر AND ، trueباشند عملگر AND مقدار true را بر می‌گرداند. در غیر اینصورت اگر یکی از مقادیر یا هر دوی آن‌ها false باشند مقدار false را بر می‌گرداند. در زیر جدول درستی عملگر AND نشان داده شده است :

X	Y	X && Y
true	true	true
true	false	false
false	true	false
false	false	false

برای درک بهتر تأثیر عملگر AND یاد آوری می‌کنم که این عملگر فقط در صورتی مقدار true را نشان می‌دهد که هر دو عملوند مقدارشان true باشد.در غیر اینصورت نتیجه تمام ترکیب‌های بعدی false خواهد شد. استفاده از عملگر ANDمانند استفاده از عملگرهای مقایسه ای است. به عنوان مثال نتیجه عبارت زیر درست (true) است اگر سن (age) بزرگ‌تر از ۱۸ و salary کوچک‌تر از ۱۰۰۰ باشد.

```
result = (age > 18) && (salary < 1000);
```

عملگر منطقی OR(||)

اگر یکی یا هر دو مقدار دو طرف عملگر OR ،درست (true) باشد،عملگر OR مقدار true را بر می‌گرداند. جدول درستی عملگر ORدر زیر نشان داده شده است :

X	Y	X || Y
true	true	true
true	false	true
false	true	true
false	false	false

در جدول بالا مشاهده می‌کنید که عملگر OR در صورتی مقدار false را بر می‌گرداند که مقادیر دو طرف آن false باشند. کد زیر را در نظر بگیرید.نتیجه این کد در صورتی درست (true) است که رتبه نهایی دانش آموز (finalGrade) بزرگ‌تر از ۷۵ یا یا نمره نهایی امتحان آن ۱۰۰ باشد.

```
isPassed = (finalGrade >= 75) || (finalExam == 100);
```

عملگرهای بیتی

عملگرهای بیتی به شما اجازه می‌دهند که شکل باینری انواع داده‌ها را دستکاری کنید.

عملگر	نام	دسته	مثال
&	بیتیAND	Binary	x = y & z;
|	بیتیOR	Binary	x = y | z;
^	بیتیXOR	Binary	x = y ^ z;
~	بیتیNOT	Unary	x = ~y;
&=	بیتیAND Assignment	Binary	x &= y;
|=	بیتیOR Assignment	Binary	x |= y;
^=	بیتیXOR Assignment	Binary	x ^= y;

عملگر بیتی (&)AND

عملگر بیتی AND کاری شبیه عملگر منطقی AND انجام می‌دهد با این تفاوت که این عملگر بر روی بیت‌ها کار می‌کند. اگر مقادیر دو طرف آن ۱ باشد مقدار ۱ را بر می‌گرداند و اگر یکی یا هر دو طرف آن صفر باشد مقدار صفر را بر می‌گرداند. جدول درستی عمگر بیتی AND در زیر آمده است:

X	Y	X AND Y
1	1	1
1	0	0
0	1	0
0	0	0

در زیر نحوه استفاده از عملگر بیتی AND آمده است:

```
int result = 5 & 3;

cout << result;
```

```
1
```

همانطور که در مثال بالا مشاهده می‌کنید نتیجه عملکرد عملگر AND بر روی دو مقدار ۵ و ۳ عدد ۱ می‌شود. اجازه بدهید ببینیم که چطور این نتیجه را به دست می‌آید:

```
5:  00000000000000000000000000000101
3:  00000000000000000000000000000011
-------------------------------------
1:  00000000000000000000000000000001
```

ابتدا دو عدد ۵ و ۳ به معادل باینری‌شان تبدیل می‌شوند. از آنجاییکه هر عدد صحیح 32 (int)بیت است از صفر برای پر کردن بیت‌های خالی استفاده می‌کنیم. با استفاده از جدول درستی عملگر بیتی AND می‌توان فهمید که چرا نتیجه عدد یک می‌شود.

عملگر بیتی OR(|)

اگر مقادیر دو طرف عملگر بیتی OR هر دو صفر باشند نتیجه صفر در غیر اینصورت ۱ خواهد شد. جدول درستی این عملگر در زیر آمده است:

X	Y	X OR Y
1	1	1
1	0	1
0	1	1
0	0	0

نتیجه عملگر بیتی OR در صورتی صفر است که عملوندهای دو طرف آن صفر باشند. اگر فقط یکی از دو عملوند یک باشد نتیجه یک خواهد شد. به مثال زیر توجه کنید:

```
int result = 7 | 9;

cout << result;
```

```
15
```

عملگر بیتی (^) XOR

جدول درستی این عملگر در زیر آمده است:

X	Y	X XOR Y
1	1	0
1	0	1
0	1	1
0	0	0

در صورتیکه عملوندهای دو طرف این عملگر هر دو صفر یا هر دو یک باشند نتیجه صفر در غیر اینصورت نتیجه یک می‌شود. در

مثال زیر تأثیر عملگر بیتی XORرا بر روی دو مقدار مشاده می‌کنید :

```
int result = 5 ^ 7;

cout << result;
```

```
2
```

عملگر بیتی تغییر مکان(shift)

این نوع عملگرها به شما اجازه می‌دهند که بیت‌ها را به سمت چپ یا راست جا به جا کنید. دو نوع عملگر بیتی تغییر مکان وجود دارد که هر کدام دو عملوند قبول می‌کنند. عملوند سمت چپ این عملگرها حالت باینری یک مقدار و عملوند سمت راست تعداد جابه جایی بیت‌ها را نشان می‌دهد.

عملگر	نام	دسته	مثال
>>	تغییر مکان به سمت چپ	Binary	x = y << 2;
<<	تغییر مکان به سمت راست	Binary	x = y >> 2;

عملگر تغییر مکان به سمت راست

این عملگر شبیه به عمگر تغییر مکان به سمت چپ است با این تفاوت که بیت‌ها را به سمت راست جا به جا می‌کند. به عنوان مثال:

```
int result = 100 >> 4;

cout << result;
```

```
6
```

با استفاده از عملگرتغییر مکان به سمت راست بیت‌های مقدار ۱۰۰ را به اندازه ٤ واحد به سمت چپ جا به جا می‌کنیم. اجازه بدهید تأثیر این جا به جایی را مورد بررسی قرار دهیم:

```
100: 00000000000000000000000001100100
------------------------------------
  6: 00000000000000000000000000000110
```

هر بیت به اندازه ٤ واحد به سمت راست منتقل می‌شود،بنابراین ٤ بیت اول سمت راست حذف شده و چهار صفر به سمت چپ اضافه می‌شود .

عملگر شرطی به صورت زیر استفاده می شود :

– Condition ? Expression1 : Expression2;

- The statement above is equivalent to:

```
if (Condition)
    Expression1;
else
    Expression2;
```

- Example:

if/else statement:

```
if (total > 12)
    grade = 'P';
else
    grade = 'F';
```

conditional statement:

```
(total > 12) ? grade = 'P': grade = 'F';
```

OR

```
grade =( total > 12) ? 'P': 'F';
```

عملگر کاما :

- (Expression1 ,Expression2,...);
- Example:
  - int x, y, z;
  - z = (x = 2, y = x + 1);
  - printf("z = %d", z);

```
int x, y, z;
x = 2;
y = x + 1;
z = y;
printf("z = %d", z);
```

عملگر sizeof به صورت زیر استفاده می شود و تعداد بایت های عبارت داده شد را بر می گرداند :

-یک عدد صحیح بر می گرداند

- sizeof *variable_Identifier;*
- sizeof *(variable_Identifier);*
- sizeof *(Data_Taype);*

- Example:
  - int x;
  - printf("size of x = %d", sizeof x);
  - printf("size of long long = %d", sizeof(long long));
  - printf("size of x = %d", sizeof (x));

تقدم عملگرها

تقدم عملگرها مشخص می‌کند که در محاسباتی که بیش از دو عملوند دارند ابتدا کدام عملگر اثرش را اعمال کند. عملگرها در C++ در محاسبات دارای حق تقدم هستند.به عنوان مثال:

```
number = 1 + 2 * 3 / 1;
```

اگر ما حق تقدم عملگرها را رعایت نکنیم و عبارت بالا را از سمت چپ به راست انجام دهیم نتیجه ۹ خواهد شد (۱+۲=۳ سپس ۳×۳=۹ و در آخر ۹/۱=۹).اما کامپایلر با توجه به تقدم عملگرها محاسبات را انجام می‌دهد.برای مثال عمل ضرب و تقسیم نسبت به جمع و تفریق تقدم دارند.بنابراین در مثال فوق ابتدا عدد ۲ ضربدر ۳ و سپس نتیجه آن‌ها تقسیم بر ۱ می‌شود که نتیجه ٦ به دست می‌آید.در آخر عدد ٦ با ۱ جمع می‌شود و عدد ۷ حاصل می‌شود. در جدول زیر تقدم عملگرهای C++ از بالا به پایین آمده است:

تقدم عملگرها

Primary Expression Operators	() [] . ->	left-to-right
Unary Operators	* & + - ! ~ ++expr --expr (typecast) sizeof	right-to-left
Binary Operators	* / %	left-to-right
	+ -	
	>> <<	
	< > <= >=	
	== !=	
	&	
	^	
	|	
	&&	
	||	
Ternary Operator	?:	right-to-left
Assignment Operators	= += -= *= /= %= >>= <<= &= ^= |=	right-to-left
Post increment	expr++ expr--	-
Comma	,	left-to-right

Type Casting

- Explicit Type cast: carried out by programmer using casting

```
int k, i = 7;
float f = 10.14;
char c = 'B';
k = (i + f) % 3; // error
k = (int)(i + f) % 3;
```

- Implicit Type cast: carried out by compiler automatically

```
f = 65.6;
i = f;  // f = (int)f;
c = i;  // c = (int)i;
```

جدول کدهای اسکی

Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char
0	00	Null	32	20	Space	64	40	@	96	60	`
1	01	Start of heading	33	21	!	65	41	A	97	61	a
2	02	Start of text	34	22	"	66	42	B	98	62	b
3	03	End of text	35	23	#	67	43	C	99	63	c
4	04	End of transmit	36	24	$	68	44	D	100	64	d
5	05	Enquiry	37	25	%	69	45	E	101	65	e
6	06	Acknowledge	38	26	&	70	46	F	102	66	f
7	07	Audible bell	39	27	'	71	47	G	103	67	g
8	08	Backspace	40	28	(	72	48	H	104	68	h
9	09	Horizontal tab	41	29	)	73	49	I	105	69	i
10	0A	Line feed	42	2A	*	74	4A	J	106	6A	j
11	0B	Vertical tab	43	2B	+	75	4B	K	107	6B	k
12	0C	Form feed	44	2C	,	76	4C	L	108	6C	l
13	0D	Carriage return	45	2D	-	77	4D	M	109	6D	m
14	0E	Shift out	46	2E	.	78	4E	N	110	6E	n
15	0F	Shift in	47	2F	/	79	4F	O	111	6F	o
16	10	Data link escape	48	30	0	80	50	P	112	70	p
17	11	Device control 1	49	31	1	81	51	Q	113	71	q
18	12	Device control 2	50	32	2	82	52	R	114	72	r
19	13	Device control 3	51	33	3	83	53	S	115	73	s
20	14	Device control 4	52	34	4	84	54	T	116	74	t
21	15	Neg. acknowledge	53	35	5	85	55	U	117	75	u
22	16	Synchronous idle	54	36	6	86	56	V	118	76	v
23	17	End trans. block	55	37	7	87	57	W	119	77	w
24	18	Cancel	56	38	8	88	58	X	120	78	x
25	19	End of medium	57	39	9	89	59	Y	121	79	y
26	1A	Substitution	58	3A	:	90	5A	Z	122	7A	z
27	1B	Escape	59	3B	;	91	5B	[	123	7B	{
28	1C	File separator	60	3C	<	92	5C	\	124	7C	|
29	1D	Group separator	61	3D	=	93	5D	]	125	7D	}
30	1E	Record separator	62	3E	>	94	5E	^	126	7E	~
31	1F	Unit separator	63	3F	?	95	5F	_	127	7F	□

Type Casting

```
char c = 'A';
short s = 1;
int i;
float f;
double d;
```

```
d = (c / i) + (f * d) + (f + i);
    (char) (int) (float) (double) (float) (int)
```

int

double

double

float

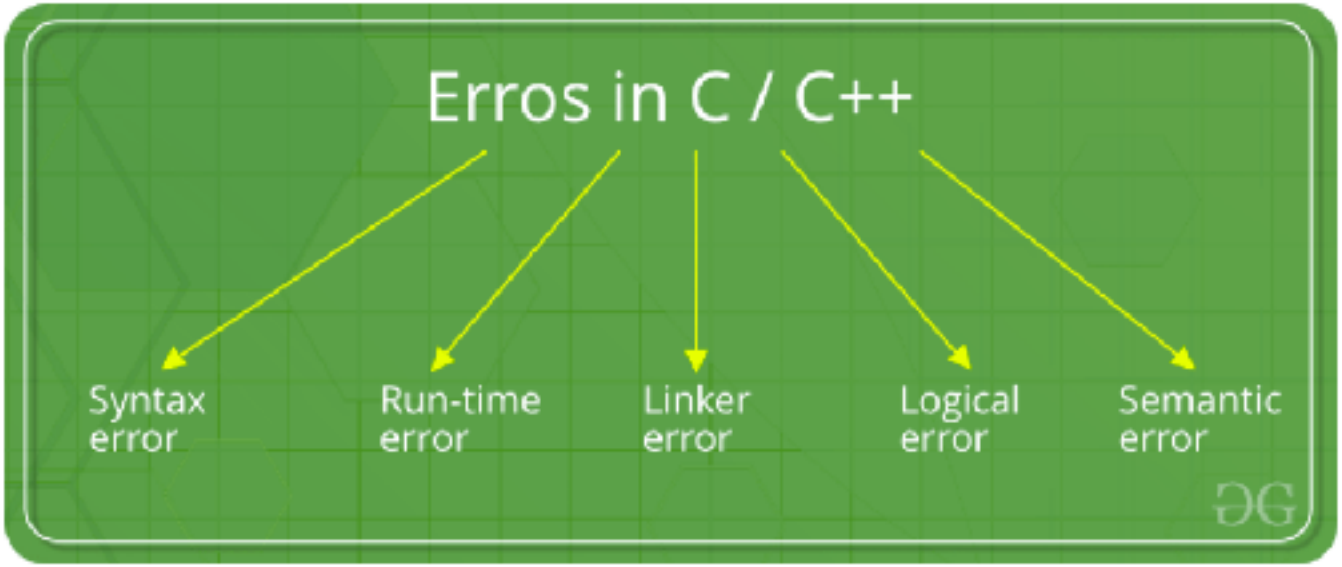
double

```
i = (c + s);
(int) (char) (short)
```

(short)

(int)

انواع خطا


Erros in C / C++
Syntax error
Run-time error
Linker error
Logical error
Semantic error

انواع خطا (ادامه)

- نحوی (Syntax)
- اجرایی (Runtime)
- منطقی (Logical)

خطای منطقی یا Logic error چیست؟

خطای منطقی (Logic error) گونه‌ای رایج از خطاهای برنامه نویسی است که منجر به عملکرد نادرست برنامه می‌شود و حاصل آن، خروجی یا رفتاری نامناسب و مخالف انتظار می‌باشد.

در کد نمونه‌ی زیر که به زبان C++ نوشته شده است نیز به دلیل یک خطای منطقی ناشی از اشتباه برنامه نویس، برای مقادیر زوج n عبارت n is odd (به معنای n فرد است) و برای مقادیر فرد n عبارت n is even (به معنای n زوج است) چاپ می‌شود. با جابجا نمودن دو دستور cout دوم و سوم این خطا برطرف می‌شود.

```
#include <iostream>
using namespace std;
int main() {
  int n;
  cout << "Enter an integer: ";
  cin >> n;
  if ( n%2 == 0) {
    cout << n << " is odd.";
  }
  else {
    cout << n << " is even.";
  }
  return 0;
}
```

خطاهای نحوی یا syntax

برخی از قواعد نحوی به شکل زیر میباشند:

1- دستورات باید به; ختم شوند.

2- رشته ها باید در علامت نقل قول (' ')قرار گیرند.

3- پارامترهای توابع باید با , از هم جدا شوند و در پرانتز قرار گیرند.

4- بستن کروشه

اگر این قواعد نقض شود با خطای نحوی مواجه می شویم:

5- اشتباه نوشتن کلمات کلیدی

6- استفادهی نادرست از نام توابع