**بسم الله الرحمن الرحیم**

**درس اول : انواع داده ها ، انواع عملگرها ، دستورات cin , cout**

کتابخانه های اصلی مورد استفاده در C++ عبارتنداز iostream

یک برنامه سی پلاس پلاس شامل این کتابخانه در هدر خود می باشد

برای شروع یک برنامه سی پلاس پلاس باید به صورت زیر برنامه را تنظیم نمود

#include <iostream>

using namespace std;

int main(int argc, char\*\* argv)

{

return 0;

}

برای چاپ خروجی در سی پلاس پلاس از دستور cout استفاده می کنیم

Cout<<

برای اینکه بعد از اجرا عباراتی در انتهای خروجی برنامه نیاید از دستور Getch() استفاده می کنیم که با کتابخانه conio.h تعریف می‌شود در حقیقت این دستور منتظر دریافت یک کاراکتر از ورودی می باشد

دستور endl همانند \n عمل می‌کند با این تفاوت که به این صورت <<endl نوشته می‌شود

کاراکتر کنترلی \b یک کاراکتر به عقب می رود

به عنوان مثال اگر به صورت زیر باشد

cout<<"Hello"<<'\b'<<"ali";

خروجی آن به صورت زیر است

Hellali

اگر عبارت ali نباشد علامت cursor زیر o قرار می گیرد

کاراکتر کنترلی \a بوق سیستم را در می‌آورد

با دستور cin از برنامه ورودی میگیریم

cin>>

دستور sizeof() تعداد بایت های یک متغیر را نشان می‌دهد

انواع متغیرها :

|  |
| --- |
|  |
| **Group** | **Type names\*** | **Notes on size / precision** |
| Character types | **char** | Exactly one byte in size. At least 8 bits. |
| **char16\_t** | Not smaller than char. At least 16 bits. |
| **char32\_t** | Not smaller than char16\_t. At least 32 bits. |
| **wchar\_t** | Can represent the largest supported character set. |
| Integer types (signed) | **signed char** | Same size as char. At least 8 bits. |
| *signed* **short** *int* | Not smaller than char. At least 16 bits. |
| *signed* **int** شامل اعداد مثبت و منفی است که صحیح هستند | Not smaller than short. At least 16 bits. |
| *signed* **long** *int* | Not smaller than int. At least 32 bits. |
| *signed* **long long** *int* | Not smaller than long. At least 64 bits. |
| Integer types (unsigned) | **unsigned char** | (same size as their signed counterparts) |
| **unsigned short** *int* |
| **unsigned** *int فقط اعداد مثبت* |
| **unsigned long** *int* |
| **unsigned long long** *int* |
| Floating-point types | **float** |  |
| **double** | Precision not less than float |
| **long double** | Precision not less than double |
| Boolean type | **bool** | False ou true |
| Void type | **void** | no storage |
| Null pointer | **decltype(nullptr)** |  |

متغیر double اعشاری بزرگتر از float است

عملگر بیتی << و << که عملگر بیتی >> مقدار را دو برابر کرده و عملگر بیتی << مقدار را نصف می‌کند

عملگر , به این صورت عمل می‌کند که در آن داریم به عنوان مثال :

Y = (x = 10 , x + 4)

در این حالت یعنی برای y عبارت x+4 قرار می‌گیرد توجه به این نکته ضروری است که اگر چند , داشته باشیم آخرین مقدار بعد از , آخر به عنوان مبنا قرار می‌گیرد

**Casting :**

در این حالت داده کوچکتر به داده بزرگتر تبدیل می‌شود

Int >ch

Double>float

Double>int

Float>int

**دو متد width و fill :**

این دو متد بعد از cout می‌آیند و width مقدار خانه خالی و fill پر کردن فضای خالی را نشان می‌دهد

**متد setf :**

این متد بعد از cout می‌آید و می‌تواند تنظیم پر کردن از سمت چپ یا راست را انجام دهد

Cout.setf(ios::left/right)

**توابع ریاضی در C++ :**

این تابع توسط کتابخانه math.h ایجاد می‌شوند

**بسم الله الرحمن الرحیم**

**درس دوم : ساختار تصمیم if و switch**

**بسم الله الرحمن الرحیم**

**درس سوّم : ساختارهای تکرار for while do-while :**

در حلقه for سه حالت قسمت داریم

For(مقدار اوّلیه ; شرط;تکرار)

**بسم الله الرحمن الرحیم**

**درس هشت : تابع :**

**از دقیقه 41:09**

**Template :**

با استفاده از Template می توان یک تابع نوشت وانواع و اقسام متغیر ها را در خروجی گرفت برای اینکار از روش زیر استفاده می‌کنیم

**Templates**

**Function templates**

Function templates are special functions that can operate with *generic types*. This allows us to create a function template whose functionality can be adapted to more than one type or class without repeating the entire code for each type.  
  
In C++ this can be achieved using *template parameters*. A template parameter is a special kind of parameter that can be used to pass a type as argument: just like regular function parameters can be used to pass values to a function, template parameters allow to pass also types to a function. These function templates can use these parameters as if they were any other regular type.  
  
The format for declaring function templates with type parameters is:  
  
template <class identifier> function\_declaration;  
template <typename identifier> function\_declaration;  
  
The only difference between both prototypes is the use of either the keyword class or the keyword typename. Its use is indistinct, since both expressions have exactly the same meaning and behave exactly the same way.  
  
For example, to create a template function that returns the greater one of two objects we could use:

|  |  |
| --- | --- |
| 1 2 3 4 | template <class myType>  myType GetMax (myType a, myType b) {  return (a>b?a:b);  } |

می توان در استفاده از template از casting نیز استفاده نمود برای اینکار بعد از نام تابع از روش <type cast int float ,…> استفاده کرد

می توان از template دو متغیر را استخراج کرد در این حالت در هنگام خروجی گرفتن حتماّ باید به صورت زیر در نظر گرفت

cout<<f<int ,int>(a);

در استفاده از template می‌‌توان دو نوع متغیر را تعریف در Class نمود که از دو نوع مختلف باشد

template <class T,class U>

void f(T a,U b)

**Over loading function Template:**

در این حالت چند تابع هم نام داریم که تابع ها را با قالب درست کردیم

**تابع بازگشتی :**

تابعی است که خودش را فراخوانی می‌کند

اگر در تابع بازگشتی جای فراخوانی تابع را با چاپ مقدار عوض کنیم عناصر در پشته قرار میگیرند و بعد از اتمام فراخوانی از پشته فراخوانی و چاپ می‌شود

توجه شود که دستور آخر در پشته قرار میگیرد

به عنوان مثال در همین درس به تمرین rec3.cpp در branch pr3 مراجعه شود

Namespace :

برای جلوگیری از تداخل و ایجاد اسکوپ از namespace استفاده می‌کنیم

می توانیم خودمان یک namespace داشته باشیم و برای استفاده از آن باید به صورت زیر عمل کنیم

Namespace::variable name

**درس هشتم :آرایه :**

آرایه محل نگهداری داده های هم نوع تحت یک نام است

تعریف آرایه

Type variable arrayname[Array number]

خانه های آرایه از صفر اندیس گذاری می شوند

grade[0] = number,string,…

grade[1] =

.

.

روش دیگر نوشتن آرایه به صورت زیر است

int a[3] = {12 , 18 , 35};

می توان تعداد عناصر آرایه را مشخص نکرد

int a[] = {12 , 18 , 35};

اگر به تعداد عناصر آرایه مقدار نگذاریم آن مقادیر را صفر می گذارد

اگر تعداد عناصر آرایه بیشتر از اندیس ها بود پیغام خطا می دهد

**What is setw Function?**

setw C++ is a method of iomaip library present in C++. setw function is a C++ manipulator which stands for set width. The manipulator sets the ios library field width or specifies the minimum number of character positions a variable will consume. In simple terms, the setw C++ function helps set the field width used for output operations. The function takes member width as an argument and needs a stream where this field has to be manipulated or inserted. The function also sets the width parameter of the stream in or stream out exactly n times. The parameter it takes will be the new value that needs to be set as the width.

Syntax of setw C++

The syntax of the function is:

setw(int number)

where number is the number of characters to be set as the field width.

The setw C++ function is used with an expression generally. The output can be set using out<<setw(num), and input can be taken by using in>>setw(num).

## **Parameters of setw Function**

As shown above, the function only takes one parameter, which is an integer. This integer corresponds to the field width to be set.

As the function acts only as a stream manipulator, it does not return anything.

## **Examples of setw Function**

The following code will help you better understand the concept of setting the width of the field:

#include <iostream>

#include <iomanip>

int main () {

std::cout << std::setw(5);

std::cout <<123<< std::endl;

return 0;

}

**توابع در آرایه ها :**

اگر در یک تابع بخواهیم آرایه را برگردانیم حتماّ باید تعداد خانه ها را نیز برگردانیم و همچنین آرایه را