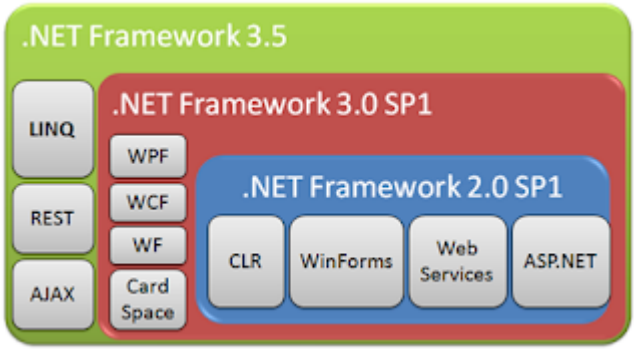
. Net Framework क्या है :-

. Net framework एक **डेवलपमेंट सॉफ्टवेयर फ्रेमवर्क** है,जिसे **माइक्रोसॉफ्ट** ने **2002** में डेवेलप किया था। **. net framework** केवल **windows operating system** में ही रन होता है,डॉट नेट एक ऐसा फ्रेमवर्क है। जिससे की **माइक्रोसॉफ्ट एनवायरनमेंट** के अंदर **web-based,windows based** तथा **console based सॉफ्टवेयर (एप्लीकेशन)** को विकसित किया जाता है। डॉट नेट फ्रेमवर्क में **CLR (Common Language Run-time)** फ्रेमवर्क की आत्मा की तरह काम करता है। डॉट नेट फ्रेमवर्क में बहुत बड़ी class library होती है,जिसे **framework class library (FCL)** कहते है इसमें कोड के बारे में पूरी जानकारी होती है। डॉट फ्रेमवर्क **java language** की तरह **pure object oriented** होता है। लेकिन यह प्लेटफार्म इंडिपेंडेंट नहीं होता है। अर्थात यह केवल **विंडोज प्लेटफार्म** पर ही रन होता है। डॉट नेट फ्रेमवर्क एक ऐसा प्लेटफार्म है,जो बहुत सारी **लैंग्वेज को सपोर्ट** करता है, अर्थात यह **दूसरी लैंग्वेज में लिखे कोड** को भी **सपोर्ट करता** है। डॉट नेट फ्रेमवर्क **Graphical User Interface (GUI)** उपलब्ध करता है।

Components of . Net Framework :-

. Net Framework के निम्मलिखित components होते है।

1.CLR(Common Language run-time)
2.CTS(Common Type System)
3.FCL(. Net Framework Class Library)
4. .Net Languages



1.CLR(Common Language Run-time) :-

यह एक **virtual machine** की तरह कार्य करता है,और सारी लैंग्वेज को **एक्सेक्यूटे** करता है। **CLR source code को byte code में translate** कर देता है,जिसे हम **CIL(Common Intermediate Language) या MSIL(Microsoft Intermediate Language)** कहते है। तथा इसके बाद **run-time में CLR,JIT compiler** के द्वारा इस **CIL कोड या MSIL कोड को native code** में बदल देता है।

2.CTS(Common Type System) :-

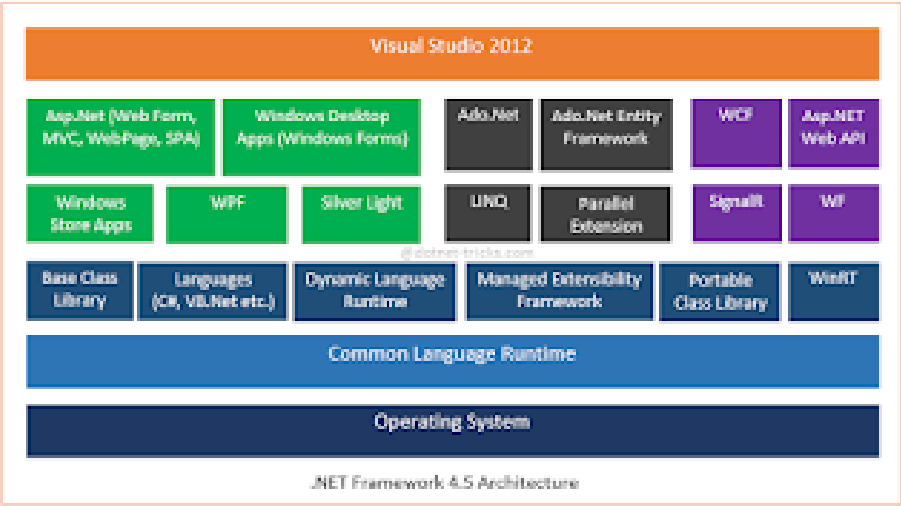
CTS ऐसे टाइप के समूह को **describe** करता है, जिसका की सभी **डॉट नेट लैंग्वेज** में सामान रूप से प्रयोग किया जाता है। उदाहरण के लिए **C#** में एक **क्लास लाइब्रेरी** का प्रयोग किया जाता है,और इस **क्लास लाइब्रेरी** का प्रयोग **VB.Net** में भी किया जाता है। क्योकि यह **क्लास लाइब्रेरी CTS** के द्वारा डिफाइन है। यह टाइप वैल्यू टाइप तथा **रिफरेन्स टाइप** हो सकते है,वैल्यू टाइप को वैल्यू के द्वारा पास किये जाते है। जो कि **stack में स्टोर** रहते है। **रिफरेन्स टाइप** को रिफरेन्स के द्वारा पास किया जाता है,जो कि **heap में स्टोर** होता है।

3. FCS(Framework Class Library) :-

FCL माइक्रोसॉफ्ट की स्टैण्डर्ड लाइब्रेरी है,तथा यह **reusable,class,interfaces** तथा वैल्यू टाइप की लाइब्रेरी है। इसका प्रयोग **web-based,Windows GUI application** तथा **ASP.Net application,console based application** को विकसित करने के लिए किया
जाता है।

4. Net language :-

. Net Framework में **Visual Basic(VB) . Net,Visual C#,ASP. Net,Jscript .Net,ADO.Net** तथा अन्य **. Net Language** है।



Architecture of .NET Framework in Hindi

1. CLR (Common Language Run-Time)
2. CTS (Common Type System)
3. FCL (Framework Class Library)
4. .NET Language
5. Common Language Infrastructure
6. Assemblies
7. C++/CLI

CLR (Common Language Run–Time) यह एक Virtual Machine की तरह काम करता है और सभी Languages को Execute करता है। यह Source Code को Byte Code मे Translate कर देता है जिसे हम **CIL (Common Intermediate Language)** या **MSIL (Microsoft Intermediate Language)** कहते है। बाद मे Run-Time मे CLR, JIT Compiler के द्वारा CIL Code को Native Code मे बदल देता है।

Assemblies अब जो Compile किया हुअ CIL Code होता है वह Code CLI Assembly मे Store होता है। Assemblies को Portable Executable (PE) फाइल फॉर्मैट मे Store किया जाता है जो कि सभी Dynamic Link Library (DLL) और Executable EXE Files के लिए के लिए Windows Platform पर आम बात है।

CTS (Common Type System) Common Type System ऐसे प्रकार के Group को Describe करता है जिसका सभी .NET Language मे समान रूप से प्रयोग किया जाता है। जैसे C# मे एक Class Library का प्रयोग होता है और इसी Class Library का प्रयोग VB.Net मे भी किया जाता है। यह Class Library CTS के द्वारा डिफाइन होती है।

FCS (Framework Class Library) जैसा की हमने अभी पढ़ा कि FCS माइक्रोसॉफ्ट की Standard Library है। इस Library का प्रयोग Windows GUI Application, Web Based, ASP.Net Application और Console Based Application को Develop करने के लिए किया जाता है।

.NET Languages .NET Framework मे Visual Basic (VB), .NET, Visual C#, ADO.Net और Jscript जैसी Languages का इस्तेमाल किया जाता है।

CLI (Common Language Infrastructure) यह Application Development और Execution के लिए एक Language Neutral Platform प्रदान करता है। Common Language Infrastructure मे .NET Framework के

Core Aspects को Implement करने पर सभी Function एक ही Language पर Depend नहीं होंगे बल्कि यह Bahut सारी अन्य Languages को भी Support करेंगे।

App Models Class Library मे बहुत सारी App Models का इस्तेमाल Apps को Create करने के लिए किया जाता है। .NET फ्रेमवर्क Windows Form, Windows Presentation, Foundation, ASP.NET Core और ASP.NET जैसी Application को By Default पहले से ही Support करता है। इसमे अन्य App Models को Develop करने के लिए .NET Framework मे कुछ Alternative Implements करने पड़ते है।

C++/CLI Microsoft ने Visual Studio मे **सन 2005** मे C++/CLI को Introduce किया था। इसके माध्यम से .NET Framework के अंतर्गत Visual C++ Program को Run किया जा सकता था।

Advantages of .NET Framework

- यह Object Oriented सॉफ्टवेयर है।
- इसमे .NET इसकी Monitoring के लिए खुद जिम्मेदार रहता है जब भी आपको इसमे कोई Problem आती है तो यह खुद ही इन Problems को Solve कर देता है।
- इसमे Management और Monitoring का काम भी .NET Framework के द्वारा ही किया जाता है अगर आपकी कोई Ongoing Process को कोई नुकसान हो जाता है या यह Destroy हो जाती है तो नई Process को आसानी से Start किया जा सकता है।
- इसमे Write करना और इसको Maintain करना बहुत आसान होता है।
- यह आपका बहुत समय बचाता है क्योंकि यह .NET Framework मे Coding के Large Part को Remove कर दिया जाता है।
- इसमे किसी भी Task पर Perform करना बहुत Simple होता है।
- .NET Framework मे Developers के लिए बहुत सारे Feature उपलब्ध रहते है।

Disadvantages of .NET Framework in Hindi

- .NET Framework के साथ आप जो भी Code रन करते है वह कोड Native Code की तुलना मे Slower होते है।
- .NET Framework मे Vendor Lock- in को Involve किया गया है जिसका मतलब है कि इसमे Future मे कोई भी Development माइक्रोसॉफ्ट पर निर्भर करती है।
- यह कुछ लोगों के लिए Expensive भी हो सकता है।

C# में सामान्य भाषा रनटाइम (CLR)

सामान्य भाषा रनटाइम (CLR) Microsoft .NET फ्रेमवर्क का एक घटक है जो .NET अनुप्रयोगों के निष्पादन का प्रबंधन करता है। यह C#, VB.NET, F# और अन्य सहित विभिन्न .NET प्रोग्रामिंग भाषाओं में लिखे गए कोड को लोड करने और निष्पादित करने के लिए जिम्मेदार है।

जब एक C# प्रोग्राम संकलित किया जाता है, तो परिणामी निष्पादन योग्य कोड एक मध्यवर्ती भाषा में होता है जिसे कॉमन इंटरमीडिएट लैंग्वेज (CIL) या Microsoft इंटरमीडिएट लैंग्वेज (MSIL) कहा जाता है। यह कोड मशीन-विशिष्ट नहीं है, और यह सीएलआर स्थापित किसी भी प्लेटफ़ॉर्म पर चल सकता है। जब सीआईएल कोड निष्पादित होता है, तो सीएलआर इसे मशीन कोड में संकलित करता है जिसे प्रोसेसर द्वारा निष्पादित किया जा सकता है।

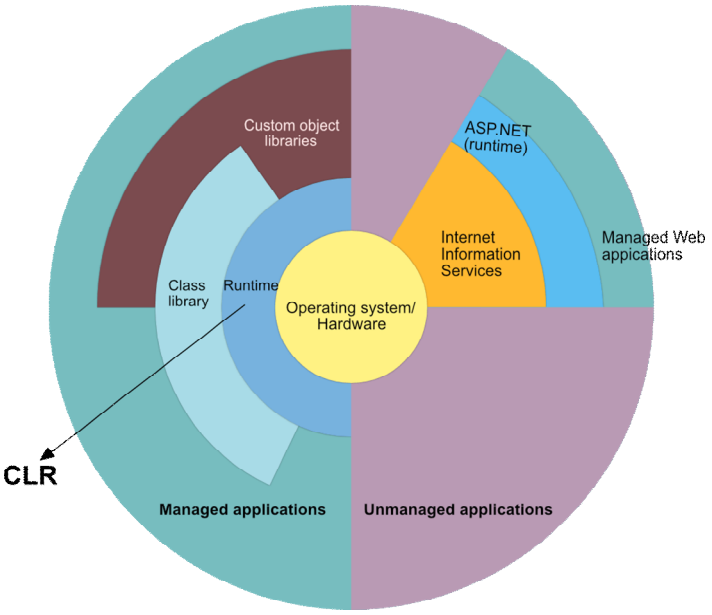
CLR .NET अनुप्रयोगों को कई सेवाएँ प्रदान करता है, जिसमें मेमोरी प्रबंधन, प्रकार सुरक्षा, सुरक्षा और अपवाद हैंडलिंग शामिल हैं। यह जस्ट-इन-टाइम (जेआईटी) संकलन भी प्रदान करता है, जो प्रोग्राम चलने पर सीआईएल कोड को मशीन कोड में संकलित करता है, प्रदर्शन को अनुकूलित करता है।

इसके अतिरिक्त, सीएलआर .NET अनुप्रयोगों को विकसित करने और तैनात करने के लिए एक ढांचा प्रदान करता है, जिसमें पुस्तकालयों का एक सेट शामिल है, जिसे .NET फ्रेमवर्क क्लास लाइब्रेरी कहा जाता है, जो इनपुट/आउटपुट संचालन, नेटवर्किंग, डेटाबेस कनेक्टिविटी जैसी कार्यक्षमता की एक विस्तृत श्रृंखला तक पहुंच प्रदान करता है। , और यूजर इंटरफ़ेस डिज़ाइन।

कुल मिलाकर, सीएलआर .NET फ्रेमवर्क का एक महत्वपूर्ण घटक है और यह सुनिश्चित करने के लिए जिम्मेदार है कि .NET अनुप्रयोगों को सुरक्षित, सुरक्षित और कुशल तरीके से निष्पादित किया जाता है, जो इसे C# प्रोग्रामिंग का एक मूलभूत पहलू बनाता है।

CLR .NET फ्रेमवर्क का मूल और वर्चुअल मशीन घटक है। यह **.NET फ्रेमवर्क में रन-टाइम वातावरण** है जो कोड चलाता है और विभिन्न सेवाएं प्रदान करके विकास प्रक्रिया को आसान बनाने में मदद करता है। मूल रूप से, यह किसी भी *.NET* प्रोग्रामिंग भाषा की परवाह किए बिना *.NET प्रोग्रामों* के निष्पादन के प्रबंधन के लिए जिम्मेदार है । आंतरिक रूप से, सीएलआर *वीईएस (वर्चुअल एक्ज़ीक्यूशन सिस्टम)* को लागू करता है जिसे माइक्रोसॉफ्ट के *सीएलआई (कॉमन लैंग्वेज इंफ्रास्ट्रक्चर) के कार्यान्वयन में परिभाषित किया गया है।* सामान्य भाषा रनटाइम के अंतर्गत चलने वाले कोड को प्रबंधित कोड कहा जाता है। दूसरे शब्दों में, आप कह सकते हैं कि सीएलआर सुरक्षा में सुधार करके *.NET* कार्यक्रमों के लिए एक प्रबंधित निष्पादन वातावरण प्रदान करता है, जिसमें क्रॉस लैंग्वेज एकीकरण और क्लास लाइब्रेरी का एक समृद्ध सेट आदि शामिल है। सीएलआर प्रत्येक .NET फ्रेमवर्क संस्करण में मौजूद है। नीचे दी गई तालिका .NET ढांचे में सीएलआर संस्करण को दर्शाती है।

नीचे दिए गए चित्र में बताया गया है कि सीएलआर क्लास लाइब्रेरी के साथ-साथ ऑपरेटिंग सिस्टम/हार्डवेयर से कैसे जुड़ा है। यहां, रनटाइम वास्तव में सीएलआर है।

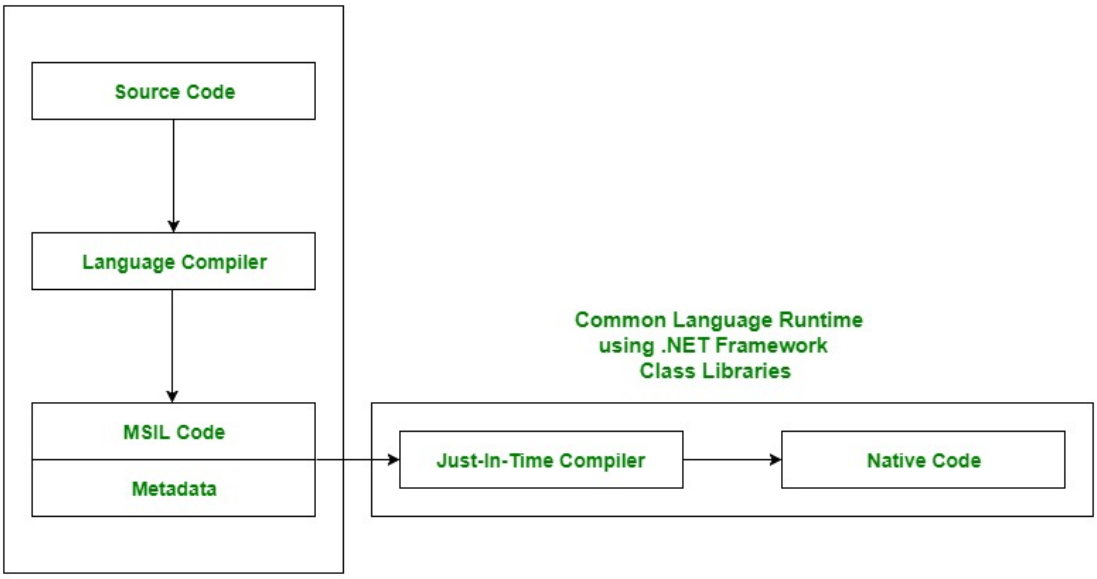


C# प्रोग्राम के निष्पादन में CLR की भूमिका

- मान लीजिए कि आपने एक C# प्रोग्राम लिखा है और इसे एक फ़ाइल में सहेजा है जिसे सोर्स कोड के रूप में जाना जाता है।

- भाषा विशिष्ट कंपाइलर स्रोत कोड को ***MSIL (Microsoft इंटरमीडिएट लैंग्वेज)*** में संकलित करता है , जिसे इसके मेटाडेटा के साथ ***CIL (कॉमन इंटरमीडिएट लैंग्वेज)*** या ***IL (इंटरमीडिएट लैंग्वेज) के रूप में भी जाना जाता है।*** *मेटाडेटा* में प्रोग्राम के सभी प्रकार, प्रत्येक फ़ंक्शन का वास्तविक कार्यान्वयन शामिल है। MSIL मशीन-स्वतंत्र कोड है।

- अब सीएलआर अस्तित्व में आ गया है। सीएलआर एमएसआईएल कोड को सेवाएं और रनटाइम वातावरण प्रदान करता है। आंतरिक रूप से सीएलआर में जेआईटी (जस्ट-इन-टाइम) कंपाइलर शामिल होता है जो एमएसआईएल कोड को मशीन कोड में परिवर्तित करता है जिसे आगे सीपीयू द्वारा निष्पादित किया जाता है। CLR .NET फ्रेमवर्क क्लास लाइब्रेरीज़ का भी उपयोग करता है। मेटाडेटा सीएलआर को प्रोग्रामिंग भाषा, पर्यावरण, संस्करण और क्लास लाइब्रेरी के बारे में जानकारी प्रदान करता है जिसके द्वारा सीएलआर एमएसआईएल कोड को संभालता है। चूंकि सीएलआर सामान्य है, इसलिए यह एक अलग भाषा में लिखे गए वर्ग के उदाहरण को किसी अन्य भाषा में लिखे गए वर्ग की विधि को कॉल करने की अनुमति देता है।



सीएलआर के मुख्य घटक

जैसा कि शब्द निर्दिष्ट करता है, सामान्य का अर्थ है सीएलआर एक सामान्य रनटाइम या निष्पादन वातावरण प्रदान करता है क्योंकि 60 से अधिक .NET प्रोग्रामिंग भाषाएं हैं।

सीएलआर के मुख्य घटक:

सामान्य भाषा विशिष्टता (सीएलएस):

यह विभिन्न .NET प्रोग्रामिंग भाषा वाक्यात्मक नियमों और विनियमों को सीएलआर समझने योग्य प्रारूप में परिवर्तित करने के लिए जिम्मेदार है। मूल रूप से, यह भाषा इंटरऑपरेबिलिटी प्रदान करता है। लैंग्वेज इंटरऑपरेबिलिटी का अर्थ है .NET फ्रेमवर्क में अन्य प्रोग्रामिंग भाषाओं को भी निष्पादन सहायता प्रदान करना।

भाषा अंतरसंचालनीयता दो तरीकों से हासिल की जा सकती है:

1. **प्रबंधित कोड:** MSIL कोड जिसे CLR द्वारा प्रबंधित किया जाता है, प्रबंधित कोड के रूप में जाना जाता है। प्रबंधित कोड के लिए CLR **तीन** .NET सुविधाएँ प्रदान करता है:

2. **अप्रबंधित कोड:** .NET विकास से पहले, प्रोग्रामिंग भाषाएं जैसे .COM कंपोनेंट्स और Win32 API MSIL कोड उत्पन्न नहीं करते हैं। इसलिए इन्हें CLR द्वारा प्रबंधित नहीं किया जाता बल्कि ऑपरेटिंग सिस्टम द्वारा प्रबंधित किया जाता है।

सामान्य प्रकार प्रणाली (सीटीएस):
प्रत्येक प्रोग्रामिंग भाषा का अपना डेटा प्रकार सिस्टम होता है, इसलिए सीटीएस .NET प्रोग्रामिंग भाषाओं के सभी डेटा प्रकार प्रणालियों को समझने और उन्हें सीएलआर समझने योग्य प्रारूप में परिवर्तित करने के लिए जिम्मेदार है जो एक सामान्य प्रारूप होगा।

प्रत्येक .NET प्रोग्रामिंग भाषा में 2 प्रकार के CTS होते हैं:

1. **मूल्य प्रकार:** मूल्य प्रकार मूल्य को सीधे मेमोरी स्थान में संग्रहीत करेगा। ये प्रकार केवल स्टैक तंत्र के साथ काम करते हैं। सीएलआर संकलन समय पर इनके लिए मेमोरी की अनुमति देता है।

2. **संदर्भ प्रकार:** संदर्भ प्रकारों में मूल्य का एक मेमोरी पता शामिल होगा क्योंकि संदर्भ प्रकार वेरिएबल मान को सीधे मेमोरी में संग्रहीत नहीं करेंगे। ये प्रकार हीप तंत्र के साथ काम करते हैं। सीएलआर रनटाइम पर इनके लिए मेमोरी आवंटित करता है।

गार्बेज कलेक्टर:
इसका उपयोग *स्वचालित मेमोरी प्रबंधन* सुविधा प्रदान करने के लिए किया जाता है। यदि कोई कचरा संग्रहकर्ता नहीं होता, तो प्रोग्रामर को मेमोरी प्रबंधन कोड लिखना होगा जो प्रोग्रामर पर एक प्रकार का ओवरहेड होगा।

जेआईटी (जस्ट इन टाइम कंपाइलर):
यह कॉमन लैंग्वेज रनटाइम वातावरण का उपयोग करके सीआईएल (कॉमन इंटरमीडिएट लैंग्वेज) को मशीन कोड या नेटिव कोड में परिवर्तित करने के लिए जिम्मेदार है।

सीएलआर के लाभ:

- यह रन टाइम पर कार्यक्रमों के बीच एक समृद्ध इंटरैक्ट प्रदान करके प्रदर्शन में सुधार करता है।

- किसी प्रोग्राम को समर्थित करने वाले किसी भी ऑपरेटिंग सिस्टम पर पुन: संकलित करने की आवश्यकता को हटाकर पोर्टेबिलिटी बढ़ाएं।

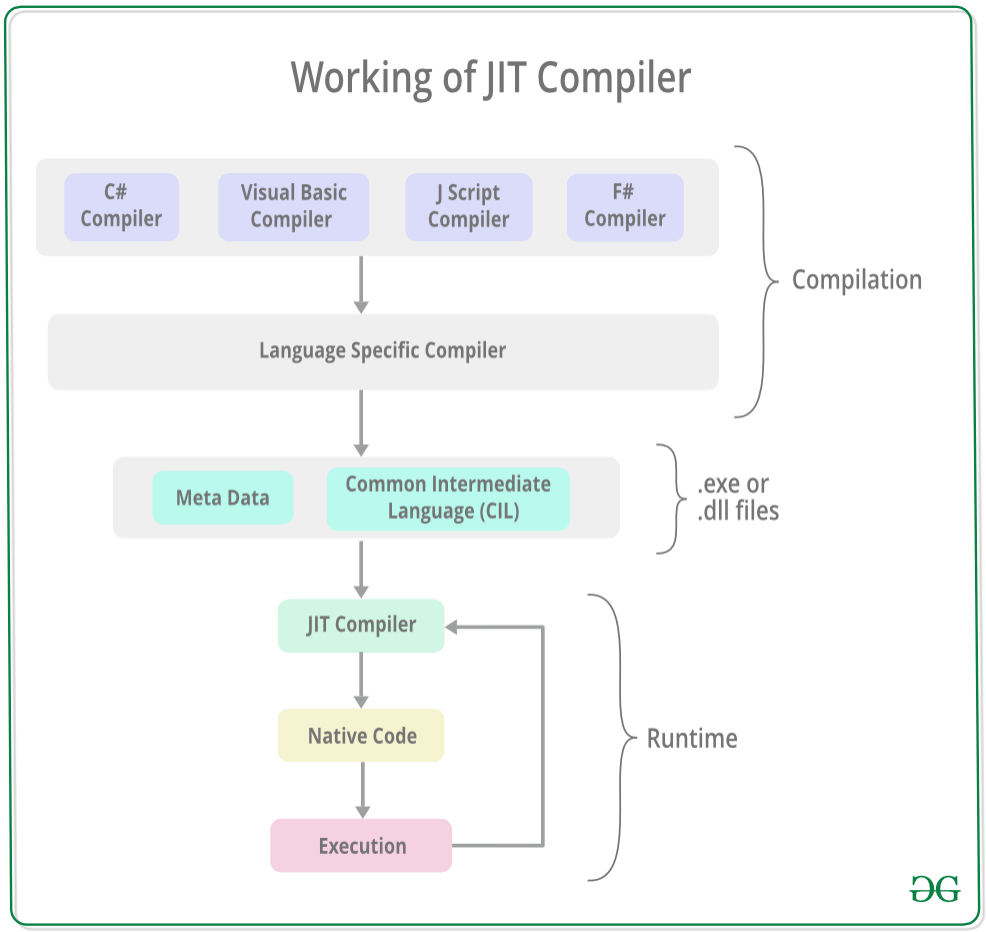
- सुरक्षा भी बढ़ जाती है क्योंकि यह MSIL निर्देशों का विश्लेषण करता है कि वे सुरक्षित हैं या असुरक्षित।साथ ही, फ़ंक्शन पॉइंटर्स के स्थान पर प्रतिनिधियों का उपयोग सुरक्षा और संरक्षा को बढ़ाता है।

- गारबेज कलेक्टर की सहायता से स्वचालित मेमोरी प्रबंधन का समर्थन करें।

- क्रॉस-भाषा एकीकरण प्रदान करता है क्योंकि सीएलआर के अंदर सीटीएस एक सामान्य मानक प्रदान करता है जो विभिन्न भाषाओं को एक-दूसरे की लाइब्रेरी का विस्तार और साझा करने के लिए सक्रिय करता है।
- अन्य .NET प्रोग्रामिंग भाषाओं में विकसित घटकों का उपयोग करने के लिए सहायता प्रदान करता है।
- भाषा, मंच और वास्तुकला को स्वतंत्रता प्रदान करें।
- यह स्केलेबल और मल्टीथ्रेडेड एप्लिकेशन के आसान निर्माण की अनुमति देता है, क्योंकि डेवलपर को मेमोरी प्रबंधन और सुरक्षा मुद्दों के बारे में सोचने की कोई आवश्यकता नहीं

.NET में जस्ट-इन-टाइम (JIT) कंपाइलर क्या है?

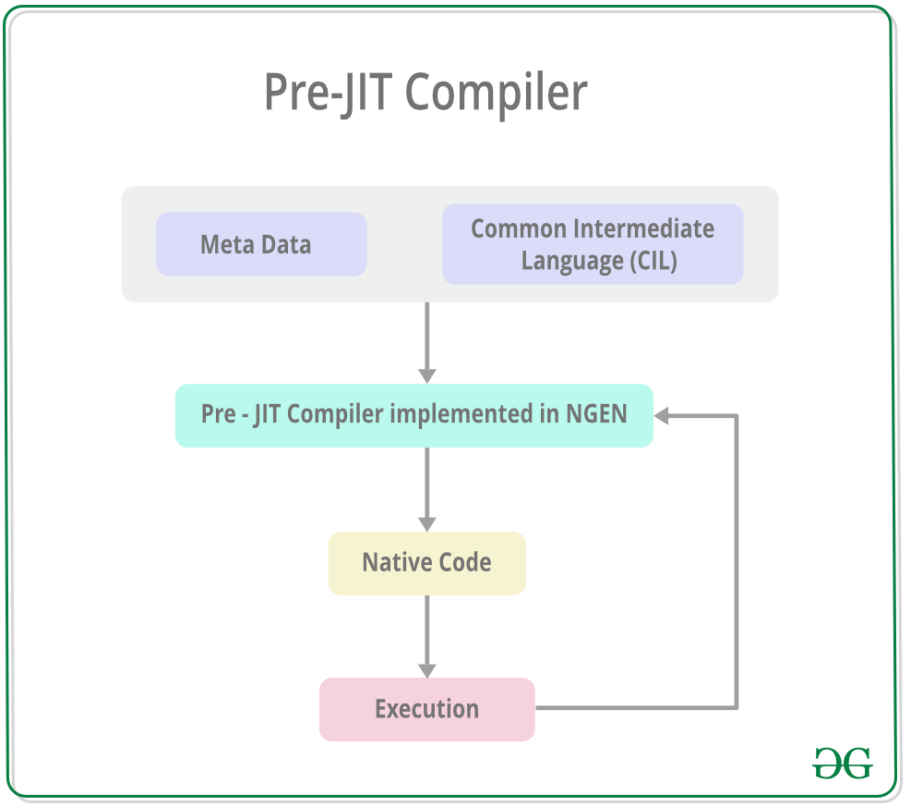
जस्ट-इन-टाइम कंपाइलर (JIT) *.NET* में **कॉमन लैंग्वेज रनटाइम (CLR)** *का एक हिस्सा है जो किसी भी .NET* प्रोग्रामिंग भाषा की परवाह किए बिना *.NET* प्रोग्रामों के निष्पादन को प्रबंधित करने के लिए जिम्मेदार है । एक भाषा-विशिष्ट कंपाइलर स्रोत कोड को मध्यवर्ती भाषा में परिवर्तित करता है। इस मध्यवर्ती भाषा को जस्ट-इन-टाइम (JIT) कंपाइलर द्वारा मशीन कोड में परिवर्तित किया जाता है। यह मशीन कोड उस कंप्यूटर वातावरण के लिए विशिष्ट है जिस पर JIT कंपाइलर चलता है। **जेआईटी कंपाइलर का कार्य:** जेआईटी कंपाइलर को कोड निष्पादन में तेजी लाने और कई प्लेटफार्मों के लिए समर्थन प्रदान करने की आवश्यकता होती है। इसकी कार्यप्रणाली इस प्रकार दी गई है:



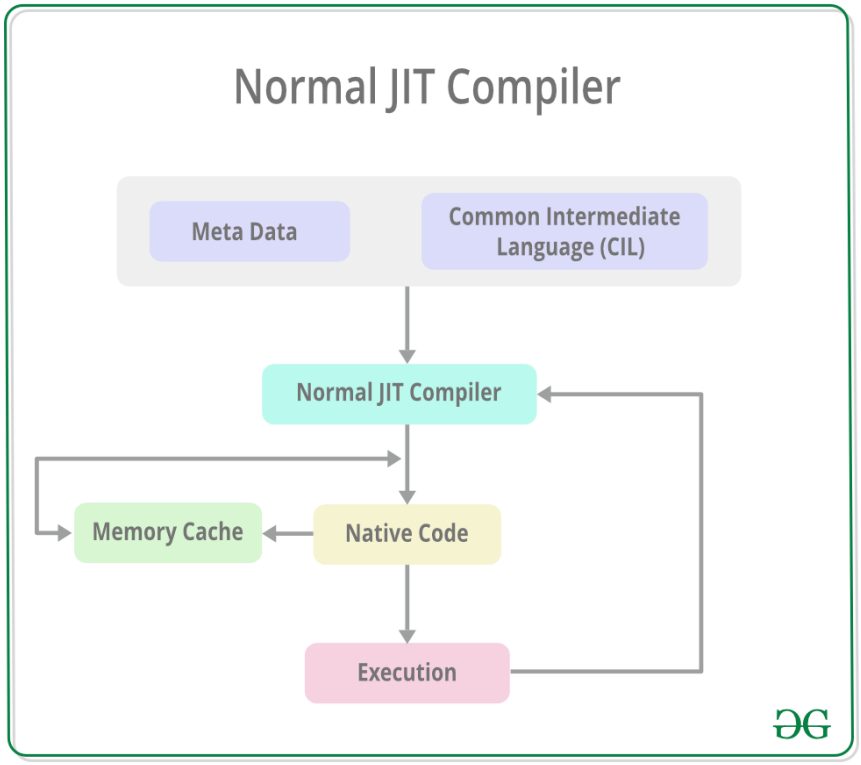
JIT कंपाइलर Microsoft इंटरमीडिएट लैंग्वेज (MSIL) या कॉमन इंटरमीडिएट लैंग्वेज (CIL) को मशीन कोड में परिवर्तित करता है। यह MSIL या CIL के निष्पादित होने से पहले किया जाता है। MSIL को आवश्यकता के आधार पर मशीन कोड में परिवर्तित किया जाता है यानी JIT कंपाइलर संपूर्ण MSIL या CIL के बजाय आवश्यकतानुसार संकलित करता है। संकलित MSIL या CIL को संग्रहीत किया जाता है ताकि आवश्यकता पड़ने पर यह बाद की कॉल के लिए उपलब्ध हो।

जस्ट-इन-टाइम कंपाइलर के प्रकार: JIT कंपाइलर 3 प्रकार के होते हैं जो इस प्रकार हैं:

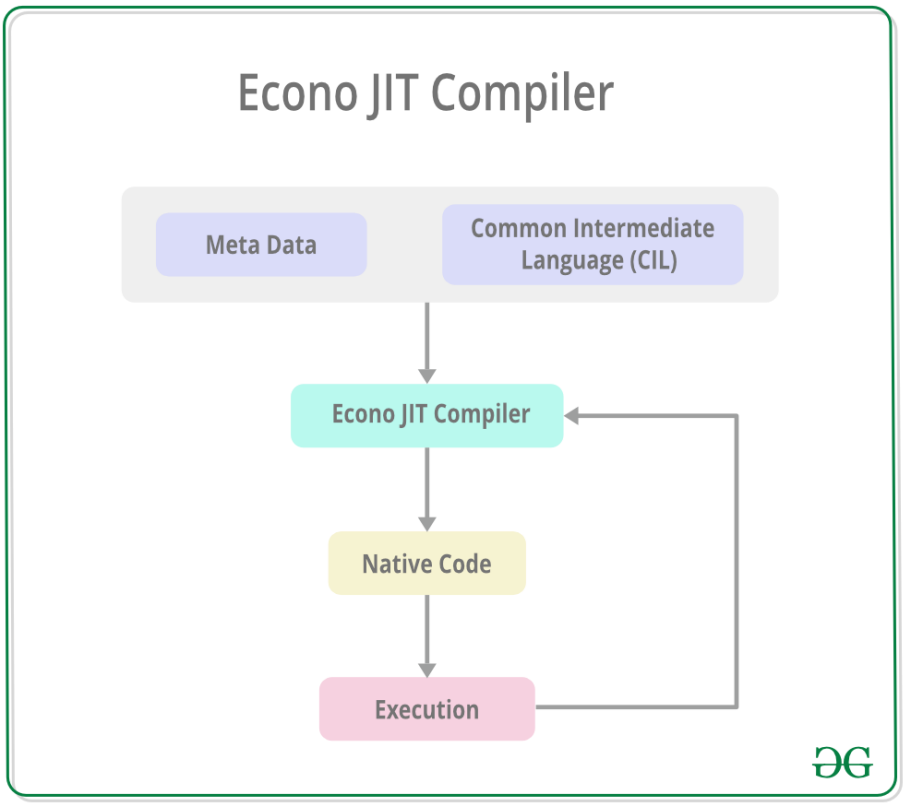
- **प्री-जेआईटी कंपाइलर:** प्री-जेआईटी कंपाइलर का उपयोग करके सभी स्रोत कोड को एक ही संकलन चक्र में एक ही समय में मशीन कोड में संकलित किया जाता है। यह संकलन प्रक्रिया एप्लिकेशन परिनियोजन के समय की जाती है। ***और यह कंपाइलर हमेशा Ngen.exe (नेटिव इमेज जेनरेटर)*** में लागू किया जाता है ।



- **सामान्य जेआईटी कंपाइलर:** रन-टाइम पर आवश्यक स्रोत कोड विधियों को सामान्य जेआईटी कंपाइलर द्वारा पहली बार बुलाए जाने पर मशीन कोड में संकलित किया जाता है। उसके बाद, उन्हें कैश में संग्रहीत किया जाता है और जब भी उन्हें दोबारा बुलाया जाता है तो उपयोग किया जाता है।



- **इकोनो जेआईटी कंपाइलर:** रन-टाइम पर आवश्यक स्रोत कोड विधियों को इकोनो जेआईटी कंपाइलर द्वारा मशीन कोड में संकलित किया जाता है। जब इन विधियों की आवश्यकता नहीं रह जाती, तो उन्हें हटा दिया जाता है। यह जेआईटी कंपाइलर डॉटनेट 2.0 से अप्रचलित है



JIT कंपाइलर के लाभ:

- जेआईटी कंपाइलर को कम मेमोरी उपयोग की आवश्यकता होती है क्योंकि रन-टाइम पर आवश्यक विधियों को जेआईटी कंपाइलर द्वारा मशीन कोड में संकलित किया जाता है।

- जेआईटी कंपाइलर का उपयोग करके पेज दोषों को कम किया जाता है क्योंकि एक साथ आवश्यक विधियां संभवतः एक ही मेमोरी पेज में होती हैं।
- सांख्यिकीय विश्लेषण के आधार पर कोड अनुकूलन, कोड चलने के दौरान JIT कंपाइलर द्वारा किया जा सकता है।

JIT कंपाइलर के नुकसान:

- JIT कंपाइलर को अधिक स्टार्टअप समय की आवश्यकता होती है जबकि एप्लिकेशन प्रारंभ में निष्पादित होता है।
- रन-टाइम पर आवश्यक स्रोत कोड विधियों को संग्रहीत करने के लिए जेआईटी कंपाइलर द्वारा कैश मेमोरी का भारी उपयोग किया जाता है।

C# में कचरा संग्रहण | ।शुद्ध रूपरेखा

कचरा संग्रहण एक मेमोरी प्रबंधन तकनीक है जिसका उपयोग .NET फ्रेमवर्क और कई अन्य प्रोग्रामिंग भाषाओं में किया जाता है। C# में, कचरा संग्रहकर्ता मेमोरी को प्रबंधित करने और स्वचालित रूप से उस मेमोरी को मुक्त करने के लिए जिम्मेदार है जिसका अब एप्लिकेशन द्वारा उपयोग नहीं किया जा रहा है।

कचरा संग्रहकर्ता एप्लिकेशन की मेमोरी को समय-समय पर स्कैन करके यह निर्धारित करने के लिए काम करता है कि कौन सी वस्तुएं अभी भी उपयोग की जा रही हैं और जिनकी अब आवश्यकता नहीं है। जिन वस्तुओं का अब उपयोग नहीं किया जा रहा है उन्हें कचरा संग्रहण के लिए चिह्नित किया जाता है, और उनकी मेमोरी कचरा संग्रहकर्ता द्वारा स्वचालित रूप से मुक्त हो जाती है।

C# में कचरा संग्रहकर्ता की कुछ प्रमुख विशेषताओं में शामिल हैं:

1. स्वचालित मेमोरी प्रबंधन: कचरा संग्रहकर्ता के साथ, डेवलपर्स को मैन्युअल रूप से मेमोरी आवंटित करने या खाली करने के बारे में चिंता करने की आवश्यकता नहीं है। कचरा संग्रहकर्ता स्वचालित रूप से मेमोरी प्रबंधन का ध्यान रखता है, जो मेमोरी लीक और अन्य समस्याओं के जोखिम को कम करने में मदद कर सकता है।
2. एप्लिकेशन प्रदर्शन पर कम प्रभाव: कचरा संग्रहकर्ता पृष्ठभूमि में चलता है और आमतौर पर एप्लिकेशन प्रदर्शन पर कम प्रभाव डालता है। हालाँकि, कुछ मामलों में, कचरा संग्रहण से एप्लिकेशन में संक्षिप्त रुकावट या मंदी आ सकती है, खासकर जब बड़ी मात्रा में मेमोरी को एक ही बार में खाली करने की आवश्यकता होती है।
3. पीढ़ी-आधारित संग्रह: C# में कचरा संग्रहकर्ता मेमोरी प्रबंधन के लिए पीढ़ी-आधारित दृष्टिकोण का उपयोग करता है। वस्तुओं को शुरू में "युवा" पीढ़ी में आवंटित किया जाता है और यदि वे कई कचरा संग्रहण चक्रों में जीवित रहती हैं तो उन्हें "पुरानी" पीढ़ी में ले जाया जाता है। यह दृष्टिकोण कचरा संग्रहण के लिए आवश्यक समय को कम करने में मदद करता है, क्योंकि अधिकांश वस्तुएं युवा पीढ़ी में एकत्र की जाती हैं।

4. अंतिम रूप देना: कचरा संग्रहकर्ता अंतिम रूप देने के लिए भी सहायता प्रदान करता है, जो एक ऐसी प्रक्रिया है जो वस्तुओं को नष्ट होने से पहले सफाई कार्य करने की अनुमति देती है। अंतिमकरणकर्ताओं वाली वस्तुओं को एक अलग अंतिमीकरण कतार में ले जाया जाता है, जिसे अन्य सभी वस्तुओं को एकत्र करने के बाद कचरा संग्रहकर्ता द्वारा संसाधित किया जाता है।

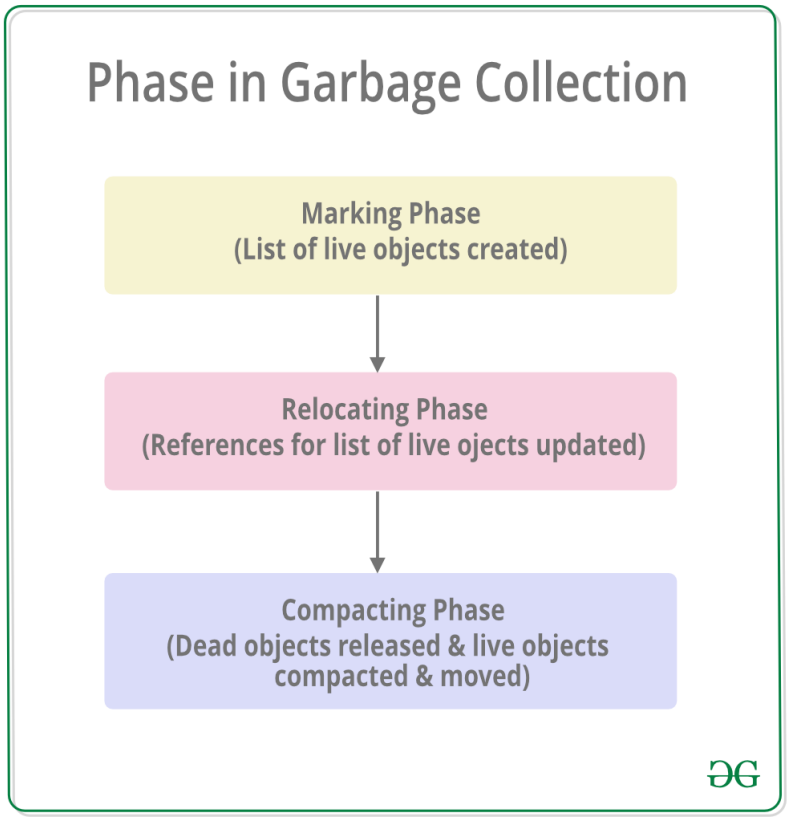
कुल मिलाकर, कचरा संग्रहकर्ता .NET फ्रेमवर्क की एक शक्तिशाली और सुविधाजनक सुविधा है जो C# अनुप्रयोगों में मेमोरी प्रबंधन को सरल बनाने में मदद कर सकती है। हालांकि यह कभी-कभी प्रदर्शन समस्याओं का कारण बन सकता है, इन्हें आम तौर पर एप्लिकेशन के मेमोरी उपयोग के सावधानीपूर्वक डिजाइन और अनुकूलन के साथ प्रबंधित किया जा सकता है।

.NET फ्रेमवर्क में गारबेज कलेक्शन द्वारा स्वचालित मेमोरी प्रबंधन को संभव बनाया गया है। जब रनटाइम पर एक क्लास ऑब्जेक्ट बनाया जाता है, तो उसे हीप मेमोरी में एक निश्चित मेमोरी स्पेस आवंटित किया जाता है। हालाँकि, प्रोग्राम में ऑब्जेक्ट से संबंधित सभी क्रियाएं पूरी होने के बाद, उसे आवंटित मेमोरी स्पेस बेकार है क्योंकि इसका उपयोग नहीं किया जा सकता है। इस मामले में, कचरा संग्रहण बहुत उपयोगी है क्योंकि इसकी आवश्यकता नहीं रहने पर यह स्वचालित रूप से मेमोरी स्पेस को मुक्त कर देता है। **कचरा संग्रहण हमेशा प्रबंधित ढेर** पर काम करेगा और आंतरिक रूप से इसमें एक इंजन होता है जिसे **अनुकूलन इंजन के** रूप में जाना जाता है। कचरा संग्रहण तब होता है जब अनेक शर्तों में से कम से कम एक शर्त पूरी होती है। ये शर्तें इस प्रकार दी गई हैं:

- यदि सिस्टम में भौतिक मेमोरी कम है, तो कचरा संग्रहण आवश्यक है।
- यदि हीप मेमोरी में विभिन्न ऑब्जेक्ट्स को आवंटित मेमोरी पूर्व-निर्धारित सीमा से अधिक हो जाती है, तो कचरा संग्रहण होता है।
- यदि *GC.Collect* विधि को कॉल किया जाता है, तो कचरा संग्रहण होता है। हालाँकि, इस विधि को केवल असामान्य परिस्थितियों में ही बुलाया जाता है क्योंकि आम तौर पर कचरा संग्रहकर्ता स्वचालित रूप से चलता है।

कचरा संग्रहण के चरण

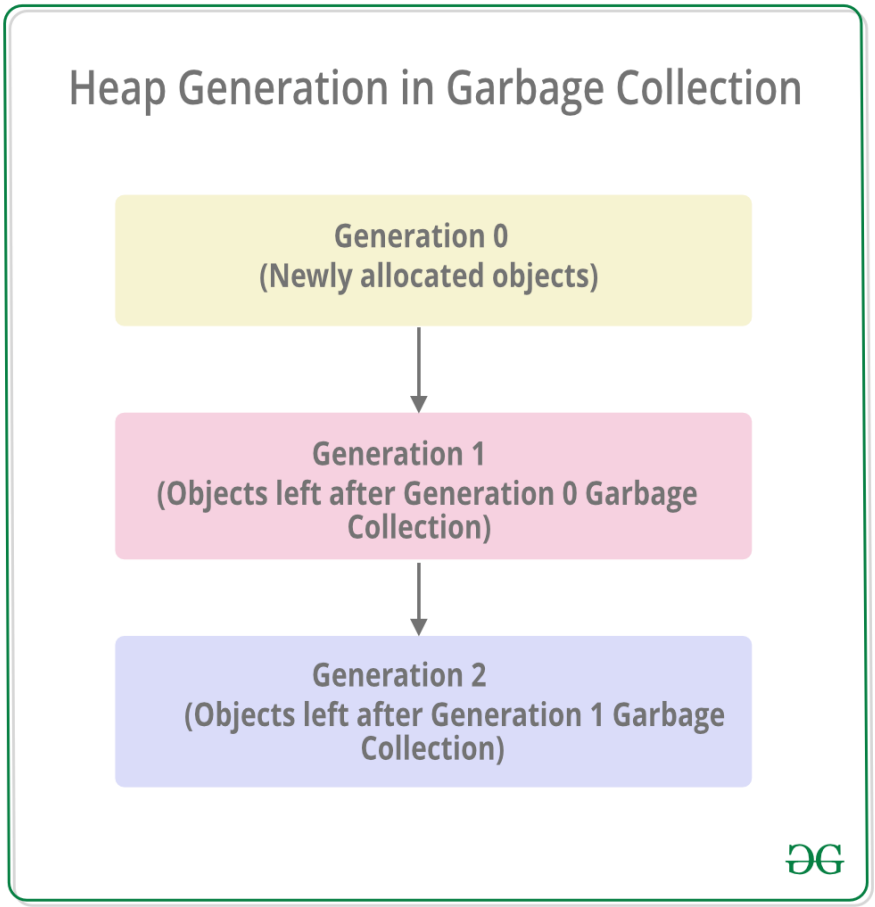
कचरा संग्रहण में मुख्यतः **3 चरण होते हैं।** इनके बारे में विवरण इस प्रकार दिया गया है:



1. **अंकन चरण:** अंकन चरण के दौरान सभी जीवित वस्तुओं की एक सूची बनाई जाती है। यह सभी मूल वस्तुओं के संदर्भों का पालन करके किया जाता है। वे सभी ऑब्जेक्ट जो लाइव ऑब्जेक्ट की सूची में नहीं हैं, संभावित रूप से हीप मेमोरी से हटा दिए जाते हैं।
2. **स्थानांतरण चरण:** सभी जीवित वस्तुओं की सूची में मौजूद सभी वस्तुओं के संदर्भों को स्थानांतरण चरण में अद्यतन किया जाता है ताकि वे नए स्थान की ओर इंगित करें जहां वस्तुओं को कॉम्पैक्टिंग चरण में स्थानांतरित किया जाएगा।
3. **संघनन चरण:** संघनन चरण में ढेर संकुचित हो जाता है क्योंकि मृत वस्तुओं द्वारा कब्जा किया गया स्थान मुक्त हो जाता है और शेष जीवित वस्तुएँ स्थानांतरित हो जाती हैं। कचरा संग्रहण के बाद बची सभी जीवित वस्तुओं को उनके मूल क्रम में ढेर मेमोरी के पुराने छोर की ओर ले जाया जाता है।

कचरा संग्रहण में ढेर पीढ़ी

ढेर मेमोरी को 3 पीढ़ियों में व्यवस्थित किया जाता है ताकि कचरा संग्रहण के दौरान विभिन्न जीवनकाल वाली विभिन्न वस्तुओं को उचित रूप से संभाला जा सके। प्रत्येक पीढ़ी को मेमोरी प्रोजेक्ट आकार के आधार पर **सामान्य भाषा रनटाइम (सीएलआर) द्वारा दी जाएगी।** आंतरिक रूप से, ऑप्टिमाइज़ेशन इंजन यह चयन करने के लिए *कलेक्शन मीन्स मेथड को* कॉल करेगा कि कौन सी वस्तुएं जेनरेशन 1 या जेनरेशन 2 में जाएंगी।



- **जनरेशन 0:** सभी अल्पकालिक वस्तुएँ जैसे अस्थायी चर, हीप मेमोरी के जनरेशन 0 में समाहित होते हैं। सभी नई आवंटित वस्तुएं भी अंतर्निहित रूप से पीढ़ी 0 ऑब्जेक्ट हैं जब तक कि वे बड़ी वस्तुएं न हों। सामान्य तौर पर, कचरा संग्रहण की आवृत्ति पीढ़ी 0 में सबसे अधिक होती है।
- **पीढ़ी 1:** यदि कुछ पीढ़ी 0 वस्तुओं द्वारा कब्जा कर लिया गया स्थान जो कचरा संग्रह में जारी नहीं किया गया है, तो ये वस्तुएं पीढ़ी 1 में स्थानांतरित हो जाती हैं। इस पीढ़ी की वस्तुएं पीढ़ी 0 और में अल्पकालिक वस्तुओं के बीच एक प्रकार का बफर हैं पीढ़ी 2 में लंबे समय तक जीवित रहने वाली वस्तुएँ।
- **पीढ़ी 2:** यदि कुछ पीढ़ी 1 वस्तुओं द्वारा कब्जा कर लिया गया स्थान जो अगले कचरा संग्रह में जारी नहीं किया जाता है, तो ये वस्तुएं पीढ़ी 2 में स्थानांतरित हो जाती हैं। पीढ़ी 2 में वस्तुएं लंबे समय तक जीवित रहती हैं जैसे स्थिर वस्तुएं क्योंकि वे ढेर मेमोरी में रहती हैं पूरी प्रक्रिया अवधि के लिए.

कचरा संग्रहण के लाभ

- कचरा संग्रहण, कचरा संग्रहण की पीढ़ियों का उपयोग करके ढेर मेमोरी पर वस्तुओं को कुशलतापूर्वक आवंटित करने में सफल होता है।
- मेमोरी को मैन्युअल रूप से खाली करने की आवश्यकता नहीं है क्योंकि कचरा संग्रह की आवश्यकता नहीं होने के बाद स्वचालित रूप से मेमोरी स्पेस खाली हो जाता है।
- कचरा संग्रह मेमोरी आवंटन को सुरक्षित रूप से संभालता है ताकि कोई भी वस्तु गलती से किसी अन्य वस्तु की सामग्री का उपयोग न करे।

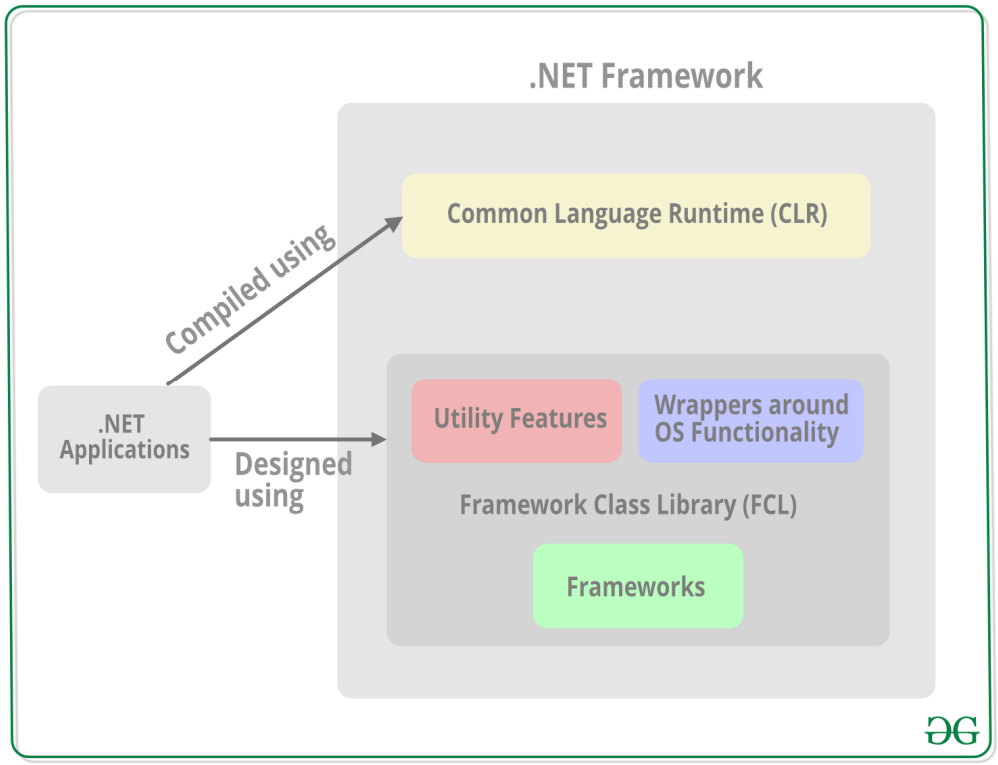
- नव निर्मित ऑब्जेक्ट के कंस्ट्रक्टर को सभी डेटा फ़ील्ड को प्रारंभ करने की आवश्यकता नहीं है क्योंकि कचरा संग्रह उन ऑब्जेक्ट की मेमोरी को साफ़ करता है जो पहले जारी किए गए थे।

.NET फ्रेमवर्क क्लास लाइब्रेरी (FCL)

फ्रेमवर्क क्लास लाइब्रेरी या एफसीएल **.NET फ्रेमवर्क** में सिस्टम कार्यक्षमता प्रदान करता है क्योंकि इसमें कई कार्य करने और डेस्कटॉप एप्लिकेशन, वेब एप्लिकेशन, मोबाइल एप्लिकेशन इत्यादि जैसे विभिन्न प्रकार के एप्लिकेशन बनाने के लिए विभिन्न कक्षाएं, डेटा प्रकार, इंटरफेस इत्यादि हैं। फ्रेमवर्क क्लास लाइब्रेरी को .NET फ्रेमवर्क के कॉमन लैंग्वेज रनटाइम (CLR) के साथ एकीकृत किया गया है और इसका उपयोग सभी .NET भाषाओं जैसे **C#** , F#, विज़ुअल बेसिक .NET, आदि द्वारा किया जाता है।

फ्रेमवर्क क्लास लाइब्रेरी में श्रेणियाँ

फ्रेमवर्क क्लास लाइब्रेरी की कार्यक्षमता को मोटे तौर पर **तीन** श्रेणियों में विभाजित किया जा सकता है यानी *.NET में लिखी गई उपयोगिता सुविधाएँ*, *ओएस कार्यक्षमता* और *फ्रेमवर्क* के आसपास रैपर । इन श्रेणियों को कठोरता से परिभाषित नहीं किया गया है और ऐसे कई वर्ग हैं जो एक से अधिक श्रेणियों में फिट हो सकते हैं।



फ्रेमवर्क-क्लास-लाइब्रेरी-एफसीएल-इन-डॉट-नेट

फ्रेमवर्क क्लास लाइब्रेरी में श्रेणियों के बारे में विवरण इस प्रकार दिया गया है:

- **उपयोगिता सुविधाएँ:** एफसीएल में उपयोगिता सुविधाओं में विभिन्न संग्रह वर्ग जैसे सूची, स्टैक, कतार, शब्दकोश इत्यादि शामिल हैं और नियमित अभिव्यक्तियों के लिए रेगेक्स क्लास जैसे अधिक विविध जोड़तोड़ के लिए कक्षाएं भी शामिल हैं।
- **ओएस कार्यक्षमता के आसपास रैपर्स:** एफसीएल में कुछ विशेषताएं अंतर्निहित विंडोज ओएस कार्यक्षमता के आसपास रैपर्स हैं। इनमें फ़ाइल सिस्टम का उपयोग करने के लिए कक्षाएं, नेटवर्क सुविधाओं को संभालने के लिए कक्षाएं, कंसोल अनुप्रयोगों के लिए I/O को संभालने के लिए कक्षाएं आदि शामिल हैं।
- **फ्रेमवर्क:** कुछ अनुप्रयोगों को विकसित करने के लिए एफसीएल में विभिन्न फ्रेमवर्क उपलब्ध हैं। उदाहरण के लिए, ASP.NET का उपयोग वेब एप्लिकेशन विकसित करने के लिए किया जाता है, विंडोज प्रेजेंटेशन फाउंडेशन (WPF) का उपयोग विंडोज एप्लिकेशन में यूजर इंटरफेस प्रस्तुत करने के लिए किया जाता है, इत्यादि।

फ्रेमवर्क क्लास लाइब्रेरी में नेमस्पेस

फ्रेमवर्क क्लास लाइब्रेरी में नेमस्पेस संबंधित वर्गों और इंटरफेस का एक समूह है जिसका उपयोग सभी *.NET फ्रेमवर्क* भाषाओं द्वारा किया जा सकता है। एफसीएल में कुछ नामस्थान उनके विवरण के साथ इस प्रकार दिए गए हैं:

नाम स्थान	विवरण
सरल उपयोग	एक्सेसिबिलिटी नेमस्पेस COM एक्सेसिबिलिटी इंटरफ़ेस के लिए प्रबंधित रैपर का एक हिस्सा है।
माइक्रोसॉफ्ट.गतिविधियाँ	Microsoft.Activities नामस्थान Windows वर्कफ़्लो फ़ाउंडेशन अनुप्रयोगों के लिए समर्थन प्रदान करता है।
माइक्रोसॉफ्ट.CSharp	Microsoft.CSharp नेमस्पेस में C# स्रोत कोड के संकलन और कोड निर्माण के लिए समर्थन है।
माइक्रोसॉफ्ट.जेस्क्रिप्ट	Microsoft.JScript नेमस्पेस में JScript स्रोत कोड के संकलन और कोड निर्माण के लिए समर्थन है।
माइक्रोसॉफ्ट.विज़ुअलबेसिक	Microsoft.VisualBasic नेमस्पेस में VisualBasic स्रोत कोड के संकलन और कोड निर्माण के लिए समर्थन है।

नाम स्थान	विवरण
प्रणाली	सिस्टम नेमस्पेस में इंटरफेस, डेटा प्रकार, इवेंट, इवेंट हैंडलर, विशेषताएँ, प्रोसेसिंग अपवाद आदि की परिभाषा के लिए आधार वर्ग हैं।
सिस्टम.गतिविधियाँ	System.Activities नेमस्पेस विभिन्न कक्षाओं का उपयोग करके विंडो वर्कफ़्लो फाउंडेशन में गतिविधियों के निर्माण और काम को संभालता है।
सिस्टम.संग्रह	System.Collections नेमस्पेस में कई मानक, विशिष्ट और सामान्य संग्रह ऑब्जेक्ट हैं जिन्हें विभिन्न प्रकारों का उपयोग करके परिभाषित किया गया है।
प्रणाली विन्यास	System.Configuration नेमस्पेस विभिन्न प्रकारों का उपयोग करके कॉन्फ़िगरेशन डेटा को संभालता है। इसमें मशीन या एप्लिकेशन कॉन्फ़िगरेशन फ़ाइलों में डेटा शामिल हो सकता है।
सिस्टम.डेटा	System.Data नेमस्पेस विभिन्न वर्गों का उपयोग करके विभिन्न स्रोतों से डेटा तक पहुंचता है और प्रबंधित करता है।
सिस्टम.ड्राइंग	System.Drawing नेमस्पेस GDI+ बुनियादी ग्राफ़िक्स कार्यक्षमता को संभालता है। विभिन्न चाइल्ड नेमस्पेस वेक्टर ग्राफिक्स कार्यक्षमता, उन्नत इमेजिंग कार्यक्षमता आदि को भी संभालते हैं।
सिस्टम.वैश्वीकरण	System.Globalization नेमस्पेस विभिन्न वर्गों का उपयोग करके भाषा, देश, उपयोग किए गए कैलेंडर, तिथियों के लिए प्रारूप पैटर्न आदि को संभालता है।
सिस्टम.आईओ	System.IO नेमस्पेस IO का समर्थन करता है जैसे डेटा को स्ट्रीम में पढ़ना/लिखना, डेटा संपीड़न, नामित पाइपों का उपयोग करके संचार करना आदि विभिन्न प्रकारों का उपयोग करना।
सिस्टम.लिंक	System.Linq नेमस्पेस विभिन्न प्रकारों का उपयोग करके भाषा-एकीकृत क्वेरी (LINQ) का समर्थन करता है।
सिस्टम.मीडिया	System.Media नेमस्पेस ध्वनि फ़ाइलों को संभालता है और विभिन्न

नाम स्थान	विवरण
सिस्टम.नेट	कक्षाओं का उपयोग करके सिस्टम द्वारा प्रदान की गई ध्वनियों तक पहुंचता है। System.Net नेमस्पेस विभिन्न वर्गों का उपयोग करके नेटवर्क प्रोटोकॉल, वेब संसाधनों के लिए कैश नीतियों, ई-मेल लिखने और भेजने आदि के लिए एक इंटरफ़ेस प्रदान करता है।
सिस्टम.प्रतिबिंब	System.Reflection नेमस्पेस लोड किए गए तरीकों, प्रकारों, फ़ील्ड्स आदि का एक प्रबंधित दृश्य देता है। यह गतिशील रूप से प्रकार भी बना और लागू कर सकता है।
सिस्टम की सुरक्षा	System.Security नेमस्पेस में .NET सुरक्षा प्रणाली और अनुमतियाँ हैं। चाइल्ड नेमस्पेस प्रमाणीकरण, क्रिप्टोग्राफ़िक सेवाएँ आदि प्रदान करते हैं।
सिस्टम.थ्रेडिंग	System.Threading नेमस्पेस विभिन्न प्रकारों का उपयोग करके मल्टीथ्रेडेड प्रोग्रामिंग की अनुमति देता है।
XamlGeneratedNamespace	XamlGeneratedNamespace में कंपाइलर-जनरेटेड प्रकार हैं जिनका उपयोग सीधे कोड से नहीं किया जाता है।

Introduction of C# in Hindi

C# को C-sharp भी कहा जाता है। C# एक type-safe ऑब्जेक्ट-ओरिएंटेड प्रोग्रामिंग लैंग्वेज है जिसे माइक्रोसॉफ्ट कारपोरेशन ने विकसित किया था।

C# लैंग्वेज में अन्य .net लैंग्वेज की तरह ही source कोड को intermediate कोड में बदला जाता है जिसे हम माइक्रोसॉफ्ट इंटरमीडिएट कोड कहते है।

C# का प्रयोग सुरक्षित तथा आकर्षक applications को बनाने में किया जाता है जो कि .net प्लेटफार्म में run होते है।

C# में c तथा c++ और java प्रोग्रामिंग लैंग्वेज की तरह ही गुणधर्म होते है।

c# माइक्रोसॉफ्ट के .net प्लेटफार्म पर कार्य करता है।

c# को CLI (common language runtime) लैंग्वेज की तरह ही specify किया जाता है।

C# को type-safe इसलिए कहा जाता है क्योंकि c# का type, c++ से अधिक सुरक्षित (safe) होता है।

सन् 1999 में, सॉफ्टवेयर इंजीनियर Anders Hejlsberg तथा उनके साथियों ने c# की विकसित किया था।

C# में programs बनाने से पहले ऐसी कुछ terms है जो आपको पता होनी चाहिए। इनके बारे में नीचे दिया जा रहा है।

MSIL (Microsoft Intermediate Language)

जब भी आप C# program को compile करते है तो सीधा machine code जनरेट नहीं होता है। बल्कि एक pseudo code generate होता है, जिसे MSIL कहते है। ये java के byte code की तरह ही होता है। जिसे बाद में convert करके machine code जनरेट किया जाता है।

CLR (Common Language Run-time)

CLR के द्वारा ही MSIL को machine code में convert किया जाता है। ये java में JVM की तरह होता है। ऐसा कोई भी program जो MSIL में convert किया गया है, उसे CLR के साथ machine code में convert किया जा सकता है। CLR आपके program के execution को manage करता है।

JIT (Just In Time) Compiler

JIT compiler के द्वारा ही MSIL को machine executable code में convert किया जाता है। जब भी आप C# program को execute करते है तो CLR के द्वारा JIT को activate किया जाता है। इसके बाद JIT जो है वह MSIL को machine executable code में convert करता है।

CLS (Common Language Specification)

.NET framework में अलग अलग languages को एक साथ काम करने के लिए कुछ common Rules को follow करने होते है। ये rules CLS द्वारा define किये जाते है। लेकिन ये तब ही possible है जब सभी languages .NET compatible (अनुकूल) हो। यदि आप ऐसा program बनाना चाहते है जो दूसरी languages यूज़ कर सके तो आपका प्रोग्राम CLS compatible होना चाहिए।

सी# विशेषताएं

C# ऑब्जेक्ट ओरिएंटेड प्रोग्रामिंग लैंग्वेज है। यह बहुत सारी **सुविधाएँ** प्रदान करता है जो नीचे दी गई हैं।

1. सरल
2. आधुनिक प्रोग्रामिंग भाषा
3. वस्तु के उन्मुख
4. सुरक्षित टाइप करें
5. इंटरोऑपरेबिलिटी
6. स्केलेबल और अद्यतन करने योग्य
7. घटक उन्मुख
8. संरचित प्रोग्रामिंग भाषा
9. समृद्ध पुस्तकालय
10. तेज़ गति

1) सरल

C# इस अर्थ में एक सरल भाषा है कि यह संरचित दृष्टिकोण (समस्या को भागों में तोड़ने के लिए), लाइब्रेरी फ़ंक्शंस का समृद्ध सेट, डेटा प्रकार आदि प्रदान करती है।

2) आधुनिक प्रोग्रामिंग भाषा

C# प्रोग्रामिंग वर्तमान प्रवृत्ति पर आधारित है और यह स्केलेबल, इंटरऑपरेबल और मजबूत अनुप्रयोगों के निर्माण के लिए बहुत शक्तिशाली और सरल है।

3) वस्तु उन्मुखी

C# ऑब्जेक्ट ओरिएंटेड प्रोग्रामिंग लैंग्वेज है। ओओपी विकास और रखरखाव को आसान बनाता है जबकि प्रक्रिया-उन्मुख प्रोग्रामिंग भाषा में यदि प्रोजेक्ट का आकार बढ़ने के साथ कोड बढ़ता है तो इसे प्रबंधित करना आसान नहीं होता है।

4) सुरक्षित टाइप करें

C# प्रकार का सुरक्षित कोड केवल उस मेमोरी स्थान तक पहुंच सकता है जिसे निष्पादित करने की उसे अनुमति है। इसलिए यह प्रोग्राम की सुरक्षा में सुधार करता है।

5) अंतरसंचालनीयता

इंटरऑपरेबिलिटी प्रक्रिया C# प्रोग्राम को लगभग वह सब कुछ करने में सक्षम बनाती है जो एक मूल C++ एप्लिकेशन कर सकता है।

6) स्केलेबल और अपडेट करने योग्य

C# स्वचालित स्केलेबल और अद्यतन करने योग्य प्रोग्रामिंग भाषा है। अपने एप्लिकेशन को अपडेट करने के लिए हम पुरानी फ़ाइलों को हटाते हैं और उन्हें नई फ़ाइलों के साथ अपडेट करते हैं।

7) घटक उन्मुख

C# घटक उन्मुख प्रोग्रामिंग भाषा है। यह प्रमुख सॉफ्टवेयर विकास पद्धति है जिसका उपयोग अधिक मजबूत और उच्च स्केलेबल अनुप्रयोगों को विकसित करने के लिए किया जाता है।

8) संरचित प्रोग्रामिंग भाषा

C# इस अर्थ में एक संरचित प्रोग्रामिंग भाषा है कि हम फ़ंक्शन का उपयोग करके प्रोग्राम को भागों में तोड़ सकते हैं। इसलिए, इसे समझना और संशोधित करना आसान है।

9) समृद्ध पुस्तकालय

C# बहुत सारे इनबिल्ट फ़ंक्शंस प्रदान करता है जो विकास को तेज़ बनाता है।

10) तेज गति

C# भाषा का संकलन और निष्पादन समय तेज़ है।

C# (sharp) data types in hindi

कंप्यूटर साइंस या कंप्यूटर प्रोग्रामिंग मे data type, डाटा का वर्गीकरण (classification) होता है! जो कि कम्पाइलर को यहाँ बताता है कि प्रोग्रामर डाटा को कैसे इस्तमाल करना चहता है!

C# मे हर एक वैरिएबल डाटा टाइप से जुड़ा होता है! हर एक डाटा टाइप को अलग मेमोरी की जररूत होती है!

1:- bool data type
bool डेटा टाइप सबसे simple डेटा टाइप है क्योंकि यह केवल 2 values ही स्टोर करता है- true or false.
c# में इसकी default वैल्यू false होती है.

2. Int Data Type
C# में integer वैल्यू के लिये int data type का प्रयोग किया जाता है!
मतलब अगर आप numeric value स्टोर करवाना चाहते हो तो तो आप को int डाटा टाइप इस्तमाल करना होगा !
Integer डाटा टाइप मे आप दसमलव वाली वैल्यू को स्टोर नहीं करवा सकते मतलब Integer डाटा टाइप, दसमलव वाली वैल्यू को स्टोर नहीं करता यह सिर्फ बिना दसमलव वाली वैल्यू को स्टोर करता है जैसा कि 11 केवल!
3. float Data Type:
अगर आप दसमलव वाली वैल्यू को स्टोर करना चाहते हो तो आप को फ्लोट डाटा टाइप इस्तमाल करना होगा जैसा कि अगर आप 56.77 को वेरिएबल मे स्टोर करना चाहते हो तो आप को फ्लोट (float) डाटा टाइप इस्तमाल करना होगा !

4. string Data Type:

अगर आप स्ट्रिंग(string) टाइप की वैल्यू को किसी वेरिएबल मे स्टोर करना चाहते हो तो आप को स्ट्रिंग डाटा टाइप इस्तमाल करना होगा!

5. short int Data Type:

अब Integer data type की भी बहुत सारी वर्गीकरण होता है जैसे short int मतलब यह केवल छोटी Integer टाइप की वैल्यू को स्टोर करता है जिसकी रेंज -32,768 to 32,767 होती है

अगर हम मेमोरी की बात करै तो यह सिर्फ 2 byte लेता है मेमोरी में सेव होने का लिए!

6. unsigned short int data type:

इसका बाद आता है unsigned short int इसका साइज केवल 0 to 65,535 होता है यह बिना sign वाली वैल्यू को स्टोर करता है

मतलब यह नेगेटिव वैल्यू को स्टोर नहीं करता है इसलिए इसका नाम unsigned डाटा टाइप है!

7. unsigned int data type:

इसके बाद आता है unsigned int यह थोड़ा बड़ा होता है इसकी रेंज 0 to 4,294,967,295 होती है यहाँ भी नेगेटिव वैल्यूज को स्टोर नहीं करता है!

8. long int data type:

इसका बाद आता है long int. यह बड़ी वैल्यू को स्टोर करना का काम मे आता है इसकी रेंज -9,223,372,036,854,775,808 to 9,223,372,036,854,775,807 होती है!

यह नेगेटिव वैल्यूज को भी स्टोर कर लेता है! c# data types

9. unsigned long int data type:

unsigned long int बहुत बड़ी बिना नेगेटिव वाली वैल्यू को स्टोर करना के काम मे आता है इसकी रेंज 0 to 18,446,744,073,709,551,615 होती है!

10. decimal data type:

इसका बाद आता है डेसीमल डाटा टाइप. यह दस्मलव वाली बड़ी वैल्यू को स्टोर करना का काम मे आता है इसकी रेंज –7.9 × 10−28 – 7.9 × 1028 होती है!

यह डाटा टाइप नेगेटिव वाली बहुत बड़ी वैल्यू को स्टोर करवा सकता है जैसा की इसकी साइज से ही पता चल रहा है !

11. User defined data type:

C# मे आप अपना डाटा टाइप भी बना सकता हो जिसको हम यूजर डिफाइंड डाटा टाइप कहते है!c# data types

12. class:

class एक user defined data type है. इस डाटा टाइप को यूजर बनाता है. user इसमें अपने अनूसार डाटा टाइप भी डिफाइन करते है

मेथड भी बनाते है! और इनको इस्तमाल करने का लिये इस class का ऑब्जेक्ट भी बनाते है! इस ऑब्जेक्ट की मदद से हम डाटा टाइप और method (functions) पर काम करते है!

अगर आप एक class बनाना चाहते हो तो आप को class कीवर्ड इस्तमाल करना होगा!c# data types

13. struct data type:

struct केवल वैल्यूज को स्टोर करना के काम मे आता है. यह क्लास से अलग होता है ! क्लास ऑब्जेक्ट

(object) के दवारा काम करती है
जबकि struct वैल्यू पर काम करता है! यही इन दोनों मे अंतर होता है!

14. interface data type:

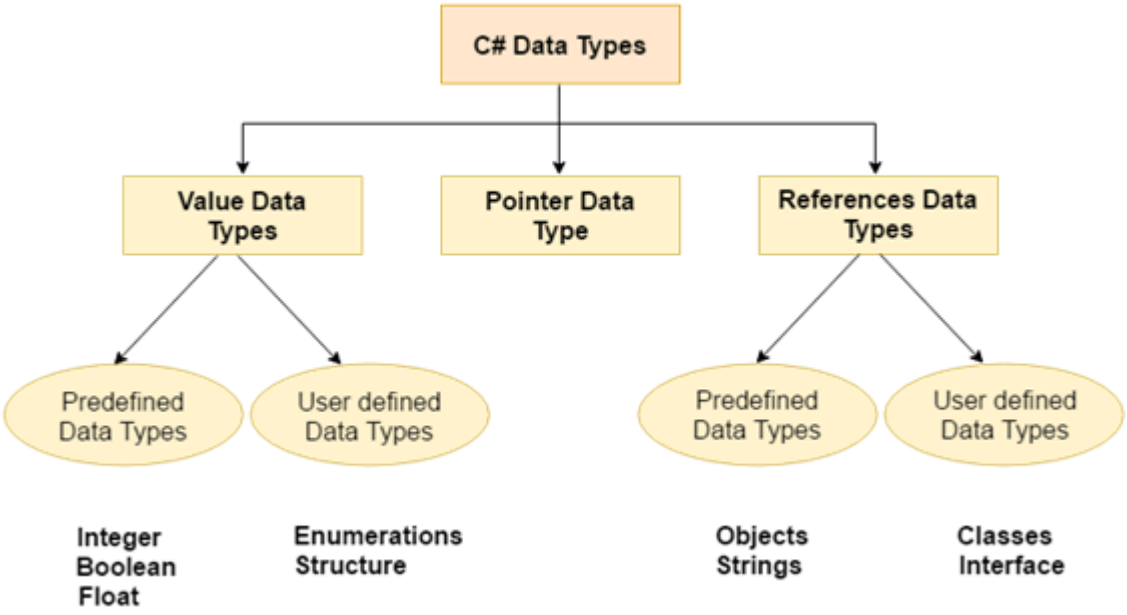
interface डाटा टाइप अनुबंद(contract) करने के काम में आता है. इसके अंदर एक या एक से ज्यादा मेथड (method) होती है जिनका प्रयोग वह class in methods को इन्हेरिट (inherit) करने के लिए करता है. इसडाटा टाइप को भी user defined data type कहते है !
अगर आप अपना प्रोग्राम मे interface इस्तमाल करना चाहते हो तो आपको interface keyword को इस्तमाल करना होगा!c# data types

15. delegate data type:

एक यूजर डिफाइंड डाटा टाइप और होता है जिसका नाम है delegate , यह मेथड के address को स्टोर करता है और फिर उसको कॉल करता है इससे प्रोग्राम की प्रोसेसिंग स्पीड बहुत तेज़ हो जाती है !
यह एक बहुत ही उपयोगी डाटा टाइप है सी शार्प प्रोग्रम्मिंग मे!

सी# डेटा प्रकार

डेटा प्रकार डेटा के प्रकार को निर्दिष्ट करता है जिसे एक वेरिएबल स्टोर कर सकता है जैसे पूर्णांक, फ्लोटिंग, कैरेक्टर इत्यादि।



C# भाषा में 3 प्रकार के डेटा प्रकार होते हैं।

प्रकार	डेटा के प्रकार
मूल्य डेटा प्रकार	शॉर्ट, इंट, चार, फ्लोट, डबल आदि
संदर्भ डेटा प्रकार	स्ट्रिंग, क्लास, ऑब्जेक्ट और इंटरफ़ेस
सूचक डेटा प्रकार	संकेत

मूल्य डेटा प्रकार

मान डेटा प्रकार पूर्णांक-आधारित और फ़्लोटिंग-पॉइंट आधारित हैं। C# भाषा हस्ताक्षरित और अहस्ताक्षरित दोनों शाब्दिकों का समर्थन करती है।

C# भाषा में वैल्यू डेटा टाइप 2 प्रकार के होते हैं।

1) पूर्वनिर्धारित डेटा प्रकार - जैसे इंटीजर, बूलियन, फ्लोट, आदि।

2) उपयोगकर्ता परिभाषित डेटा प्रकार - जैसे संरचना, गणना, आदि।

डेटा प्रकारों की मेमोरी का आकार 32 या 64 बिट ऑपरेटिंग सिस्टम के अनुसार बदल सकता है।

आइए मूल्य डेटा प्रकार देखें। इसका साइज 32 बिट ओएस के हिसाब से दिया गया है।

डेटा के प्रकार	मेमोरी का आकार	श्रेणी
चार	1 बाइट	-128 से 127
हस्ताक्षरित चार	1 बाइट	-128 से 127

अचिन्हित वर्ण	1 बाइट	0 से 127
छोटा	2 बाइट	-32,768 से 32,767
लघु हस्ताक्षरित	2 बाइट	-32,768 से 32,767
अहस्ताक्षरित लघु	2 बाइट	0 से 65,535
int यहाँ	4 बाइट	-2,147,483,648 से -2,147,483,647
हस्ताक्षरित int	4 बाइट	-2,147,483,648 से -2,147,483,647
अहस्ताक्षरित int	4 बाइट	0 से 4,294,967,295
लंबा	8 बाइट	?9,223,372,036,854,775,808 से 9,223,372,036,854,775,807
लंबे समय से हस्ताक्षरित	8 बाइट	?9,223,372,036,854,775,808 से 9,223,372,036,854,775,807
अहस्ताक्षरित लंबा	8 बाइट	0 - 18,446,744,073,709,551,615
तैरना	4 बाइट	$1.5 * 10^{-45} - 3.4 * 10^{38}$, 7-अंकीय परिशुद्धता
दोहरा	8 बाइट	$5.0 * 10^{-324} - 1.7 * 10^{308}$, 15-अंकीय परिशुद्धता

दशमलव	16 बाइट	कम से कम -7.9 * $10^{?28}$ - 7.9 * 10^{28}, कम से कम 28-अंकीय सटीकता के साथ

संदर्भ डेटा प्रकार

संदर्भ डेटा प्रकारों में एक चर में संग्रहीत वास्तविक डेटा नहीं होता है, लेकिन उनमें चर का संदर्भ होता है।

यदि किसी एक वेरिएबल द्वारा डेटा बदला जाता है, तो दूसरा वेरिएबल स्वचालित रूप से मूल्य में इस परिवर्तन को दर्शाता है।

विज्ञापन

C# भाषा में संदर्भ डेटा प्रकार 2 प्रकार के होते हैं।

1) पूर्वनिर्धारित प्रकार - जैसे ऑब्जेक्ट, स्ट्रिंग।

2) उपयोगकर्ता परिभाषित प्रकार - जैसे कक्षाएं, इंटरफ़ेस।

सूचक डेटा प्रकार

C# भाषा में पॉइंटर एक वेरिएबल है, इसे लोकेटर या इंडिकेटर के रूप में भी जाना जाता है जो किसी मान के पते की ओर इशारा करता है।

सूचक में प्रयुक्त प्रतीक

प्रतीक	नाम	विवरण
और (एम्पर्सैंड चिन्ह)	पता संचालक	किसी वेरिएबल का पता निर्धारित करें.
* (तारांकन	अप्रत्यक्ष	किसी पते के मान तक

चिन्ह)	संचालिका	पहुंचें

Type Conversion in C# in Hindi

C# में, Type Conversion एक प्रक्रिया है जिसमें data के एक type को दूसरे type में convert किया जाता है. इसे Type Casting भी कहते है.

Type Conversion तब होता है जब एक data type की value को दूसरे data type को assign की जाती है. यदि data type अनुकूल (compatible) होता है तो C# अपने आप ही Type Conversion कर देता है. परन्तु यदि वह compatible नहीं होता है तो हमें data type को manually convert करना पड़ता है.
उदाहरण के लिए – long variable को int value प्रदान करना.
Type Casting दो प्रकार की होती है:-

1. Implicit Type Conversion
2. Explicit Type Conversion

Implicit Type Conversion

इसे automatic type conversion भी कहते है क्योंकि इसमें conversion अपने-आप होता है. इस conversion को C# के द्वारा type-safe manner में किया जाता है.

यह तब होता है जब:-

- जब दो data types अनुकूल (compatible) होते हैं.
- जब हम छोटे data type की value को बड़े data type को assign करते हैं.

नीचे दी गयी table में C# के द्वारा support किये जाने वाले implicit type conversion दिए गये हैं:-

Convert from Data Type	**Convert to Data Type**
byte	short, int, long, float, double
short	int, long, float, double
int	long, float, double
long	float, double

float	double

इसका example –

```
using System;
namespace Casting{

class Test {

// Main Method
public static void Main(String []args)
{
int i = 50;

// automatic type conversion
long l = i;

// automatic type conversion
float f = l;

// Display Result
Console.WriteLine("Int value "  +i);
Console.WriteLine("Long value "  +l);
Console.WriteLine("Float value "  +f);
        }
}
}
```

इसका आउटपुट –

Int value 50
Long value 50
Float value 50

- C# Exception Handling in Hindi
- C# interface in Hindi

Explicit Type Conversion

Explicit Type Conversion को manual type conversion भी कहते है क्योंकि इसमें data type को manually कन्वर्ट किया जाता है. इसके लिए users के द्वारा pre-defined functions का प्रयोग किया जाता है.

इसे तब किया जाता है जब:-

- जब हम बड़े data type की value को छोटे data type में assign करना चाहते हैं.
- जब डाटा टाइप compatible नहीं होता.

Example –

```
using System;

namespace TypeConversion{
```

```
class ExplicitConversion {
static void Main(string[] args) {
double d = 4321.55;
int i;

// cast double to int.
i = (int)d;
Console.WriteLine(i);
Console.ReadKey();
}
}
}
```

इसका आउटपुट – 4321

Conversion Methods in C# in Hindi

C# में निम्नलिखित conversion methods होते हैं:-

Methods	Description
ToBoolean	यह type को एक boolean value में convert कर देता है.
ToByte	यह type को एक byte में convert करता है.
ToChar	यह एक type को एक single Unicode character में बदलता है..
ToDateTime	यह एक type को date-time structures में बदलता है.
ToDecimal	यह एक floating point या integer type को एक decimal type में convert करता है.
ToDouble	यह एक type को double में convert करता है.
ToInt16	यह एक type को एक 16-bit integer में बदलता है.
ToInt32	यह एक type को एक 32-bit integer में convert करता है.
ToInt64	यह एक type को एक 64-bit integer में convert करता है.

ToSbyte	यह type को एक signed byte type में convert करता है.
ToSingle	यह एक type को एक छोटे floating point number में convert करता है.
ToString	यह एक type को एक string में बदल देता है.
ToType	यह एक type को एक specified type में बदल देता है.
ToUInt16	यह एक type को एक unsigned int type में बदल देता है.
ToUInt32	यह एक type को एक unsigned long type में बदल देता है.
ToUInt64	यह एक type को एक unsigned big integer में बदल देता है.

What is variable in hindi

वेरिएबल एक स्टोरेज एरिया होता है जिसका प्रयोग डेटा को स्टोर करने के लिए किया जाता है. variable डेटा को contain किये रहता है जिसे प्रोग्राम execution के समय कभी भी बदला जा सकता है.

वेरिएबल को declare करने के बाद इसे एक वैल्यू दी जाती है. इस को वैल्यू बराबर के चिन्ह (=) के द्वारा दी जाती है. **जैसे- x = 5; या a = 10;**
कभी भी वेरिएबल का प्रयोग करने से पहले इसे declare करना पड़ता है.
उदाहरण के लिए माना कि हमें किसी छात्र का नाम तथा उसका रोल नंबर वेरिएबल में स्टोर करना है. इसके लिए हम दो variables लेते है एक वेरिएबल में छात्र का नाम स्टोर करते है जबकि दुसरे वेरिएबल में रोल नंबर स्टोर करते है.

types of variable in hindi (वेरिएबल के प्रकार):-

वेरिएबल की अपनी एक सीमा या बाउंड्री होतो है जिसके बाहर वह कार्य नहीं करता है इस सीमा को variable का scope कहते है. वेरिएबल अपने स्कोप के आधार पर दो प्रकार का होता है जो निम्नलिखित है:-

1:- global
2:- local

1:- global variable (ग्लोबल वेरिएबल):-

ग्लोबल वेरिएबल एक ऐसा वेरिएबल होता है जो कि functions के बाहर declare होता है. ग्लोबल वेरिएबल का प्रयोग सभी functions में हो सकता है.

2:- local variable (लोकल वेरिएबल):-

लोकल वेरिएबल एक वेरिएबल है जो कि function के अन्दर declare होता है. लोकल वेरिएबल का प्रयोग केवल वहां पर होता है जहाँ पर फंक्शन declare होता है.

Operators in hindi

किसी भी प्रोग्रामिंग भाषा में प्रयोग किये जाने वाले operators वे संकेत होते हैं जो कि कंप्यूटर कम्पाइलर को गणितीय या लॉजिकल संगणनाएं करने के लिए निर्देश देते है.
सी भाषा में भी operator का प्रयोग गणना करने तथा निर्णय लेने के लिए ही किया जाता है. operator का प्रयोग वेरिएबल अथवा संख्याओं के साथ किया जा सकता है.

types of operators in hindi (ऑपरेटर्स के प्रकार):-

'सी' प्रोग्रामिंग भाषा में operators के निम्नलिखित प्रकार होते है:-

1:- Arithmetic Operator (अरिथमेटिक ऑपरेटर)
2:- Relational Operator (रिलेशनल ऑपरेटर)
3:- Logical Operator (लॉजिकल)
4:- Bitwise Operator (बिटवाइज)
5:– Assignment Operator (असाइनमेंट)
6:– increment & decrement operators (इन्क्रीमेंट तथा डिक्रिमेंट)
7:- अन्य ऑपरेटर्स

1:- arithmetic operators (अंकगणितीय ऑपरेटर):-

arithmetic ऑपरेटर्स का प्रयोग आंकिक गणनाओं के लिए किया जाता है. 'सी' में arithmetic ऑपरेटर + का प्रयोग जोड़ (addition) के लिए, – का प्रयोग घटाने (substraction) के लिए, * का प्रयोग गुणा (multiply) के लिए, / का प्रयोग भाग (dividation) तथा % का प्रयोग भाग-अवशेष (modulo division) के लिए किया जाता है.

2:- Relational operators (रिलेशनल ऑपरेटर):-

जब दो संख्याओं में असमानता अथवा समानता प्रकट करते हुए लिखना होता है तब हम रिलेशनल ऑपरेटर का प्रयोग करते है. 'सी' भाषा में प्रयोग होने वाले रिलेशनल ऑपरेटर निम्नवत हैं:-

3:- logical operators (लॉजिकल ऑपरेटर्स):-

'सी' में लॉजिकल ऑपरेटर का प्रयोग variables में लॉजिकल ऑपरेशन करने के लिए किया जाता है.

operators	Example/Description
&& (logical AND)	(a>6)&&(b<6) यह true दिखाता है यदि दोनों कंडीशन सत्य(true) हो तो.
|| (logical OR)	(a>=12)||(b>=12) यह true दिखाता है यदि एक कंडीशन सत्य हो तो.
! (logical NOT)	!((a>6)&&(b<6)) यह true return करता है जब conditions satisfy नहीं होती है तो

4:- assignment operators (असाइनमेंट ऑपरेटर):-

जब किसी वेरिएबल को मान प्रदान किया जाता है, तो असाइनमेंट operator का प्रयोग किया जाता है. 'सी' भाषा में यह ऑपरेटर (=) है.
int x = 5;

इस ऑपरेटर के साथ अंकगणितीय ऑपरेटर (+, -, *,/ तथा, %) का प्रयोग करके बहुत छोटे स्टेटमेंट द्वारा वेरिएबल को मान प्रदान किया जा सकता है. जैसे, यदि हमें लिखना है-

Int x = x + 5;

इस स्टेटमेंट को हम इस प्रकार भी लिख सकते हैं.

int x += 5;

5:- bitwise operators (बिटवाइज ऑपरेटर):-
bit लेवल के ऑपरेशन करने के लिए c लैंग्वेज में बिटवाइज ऑपरेटर का प्रयोग किया जाता है.

6:- increment & decrement operators (इन्क्रीमेंट तथा डिक्रिमेंट ऑपरेटर):-
ये ऑपरेटर्स किसी एक operand पर ही कार्य करते हैं. इनको unary operator भी कहते है.

जब हमें किसी वेरिएबल में से एक घटाना अथवा एक जोड़ना हो तो हम इन्क्रीमेंट अथवा डिक्रिमेंट ऑपरेटर का प्रयोग करते हैं.

'सी' में यह ऑपरेटर '–' और '++' है. इस operator में यह ध्यान रखना चाहिए कि ऑपरेटर वेरिएबल के दायीं ओर प्रयोग करना है अथवा बायीं ओर क्योंकि दिशा बदलने से इनका स्वभाव बदल जाएगा.

a++;
++a;
a–;

–a;
यदि वेरिएबल के बायीं और इस ऑपरेटर का प्रयोग किया जाता है, तो यह पहले वेरिएबल में एक जोड़ता अथवा घटाता है. यदि ऑपरेटर वेरिएबल के दाई ओर प्रयोग किया जाता है, तो यह ऑपरेटर बाद में घटाता अथवा जोड़ता है इसे इस प्रकार समझा जा सकता है मान लेते हैं कि a = 4 और b = 0 है तो-

b = ++a;

इस स्टेटमेंट में पहले a में एक जुड़ने के बाद वह मान b को भी प्रदान हो जाएगा. अब a और b दोनों वेरिएबल्स का मान 5 हो जाएगा.

यदि इस स्टेटमेंट को इस प्रकार लिखते हैं:-

b = a++;

इस स्टेटमेंट में पहले b को वेरिएबल a का मान प्राप्त होगा और उसके बाद a में एक जुड़ेगा. इस प्रकार b का मान 4 और a का मान 5 हो जाएगा.

7:- अन्य operators:-

सी भाषा में & ऑपरेटर का प्रयोग किसी भी वेरिएबल के एड्रेस को एक्सेस करने के लिए प्रयुक्त होते है.

sizeof ऑपरेटर का प्रयोग वेरिएबल के साइज़ को एक्सेस करने के लिए किया जाता है.

C# में लूप्स

प्रोग्रामिंग भाषा में लूपिंग, स्टेटमेंट निष्पादित करने के लिए मूल्यांकन की जाने वाली स्थिति के परिणाम के आधार पर एक स्टेटमेंट या स्टेटमेंट के सेट को कई बार निष्पादित करने का एक तरीका है। लूप के भीतर कथनों को निष्पादित करने के लिए परिणाम की स्थिति सही होनी चाहिए। लूप्स को मुख्य रूप से दो श्रेणियों में विभाजित किया गया है: **एंट्री कंट्रोल्ड लूप्स:** जिन लूप्स की परीक्षण की जाने वाली स्थिति लूप बॉडी की शुरुआत में मौजूद होती है, उन्हें **एंट्री कंट्रोल्ड लूप्स** के रूप में जाना जाता है । **जबकि लूप** और **फॉर लूप** एंट्री नियंत्रित लूप हैं। **1. जबकि लूप** परीक्षण की स्थिति लूप की शुरुआत में दी गई है और सभी कथन तब तक निष्पादित किए जाते हैं जब तक कि दी गई बूलियन स्थिति संतुष्ट नहीं हो जाती है, जब स्थिति गलत हो जाती है, तो नियंत्रण लूप से बाहर हो जाएगा। **वाक्य - विन्यास:**

```
जबकि (बूलियन स्थिति)
{
    लूप स्टेटमेंट...
```

```
}
```

फ़्लोचार्ट:

उदाहरण:

- सी तेज

```
// C# program to illustrate while loop

using System;



class whileLoopDemo

{

    public static void Main()

    {

        int x = 1;



        // Exit when x becomes greater than 4

        while (x <= 4)

        {

            Console.WriteLine("GeeksforGeeks");



            // Increment the value of x for

            // next iteration

            x++;

        }

    }
```

```
}
```

आउटपुट:

गीक्सफॉरगीक्स

गीक्सफॉरगीक्स

गीक्सफॉरगीक्स

गीक्सफॉरगीक्स

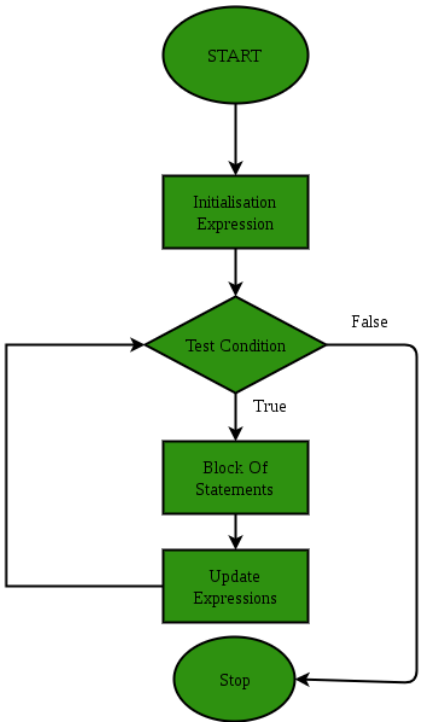
समय जटिलता: O(1)

सहायक स्थान: O(1)

2. फॉर लूप में लूप के समान कार्यक्षमता होती है लेकिन अलग सिंटैक्स के साथ। लूप के लिए प्राथमिकता तब दी जाती है जब लूप स्टेटमेंट को निष्पादित करने की संख्या पहले से ज्ञात हो। लूप वेरिएबल आरंभीकरण, परीक्षण की जाने वाली स्थिति, और लूप वेरिएबल की वृद्धि/कमी लूप के लिए एक पंक्ति में की जाती है जिससे लूपिंग की एक छोटी, डीबग करने में आसान संरचना प्रदान की जाती है।

```
(लूप वैरिएबल इनिशियलाइज़ेशन के लिए; परीक्षण की स्थिति;
                                    वृद्धि/कमी)
{
    // निष्पादित किए जाने वाले कथन
}
```

फ़्लोचार्ट:



1. लूप वेरिएबल का आरंभीकरण: लूप को नियंत्रित करने वाले एक्सप्रेशन/वेरिएबल को यहां आरंभ किया गया है। यह फॉर लूप का शुरुआती बिंदु है। पहले से घोषित

वेरिएबल का उपयोग किया जा सकता है या एक वेरिएबल घोषित किया जा सकता है, केवल स्थानीय रूप से लूप के लिए। **2. परीक्षण की स्थिति:** लूप के बयानों को निष्पादित करने के लिए परीक्षण की स्थिति। इसका उपयोग लूप के लिए निकास स्थिति का परीक्षण करने के लिए किया जाता है। इसे एक बूलियन मान **true या false** लौटाना होगा । जब स्थिति झूठी हो जाती है तो नियंत्रण लूप से बाहर हो जाएगा और लूप समाप्त हो जाएगा। **3. वृद्धि / कमी:** लूप वेरिएबल को आवश्यकता के अनुसार बढ़ाया / घटाया जाता है और नियंत्रण फिर से परीक्षण स्थिति में स्थानांतरित हो जाता है। **नोट:** इनिशियलाइज़ेशन भाग का मूल्यांकन केवल एक बार किया जाता है जब लूप शुरू होता है। **उदाहरण:**

- सी तेज

```
// C# program to illustrate for loop.

using System;



class forLoopDemo

{

    public static void Main()

    {

        // for loop begins when x=1

        // and runs till x <=4

        for (int x = 1; x <= 4; x++)

            Console.WriteLine("GeeksforGeeks");

    }

}
```

आउटपुट:

गीक्सफॉरगीक्स

गीक्सफॉरगीक्स

गीक्सफॉरगीक्स

गीक्सफॉरगीक्स

समय जटिलता: O(1)

सहायक स्थान: O(1)

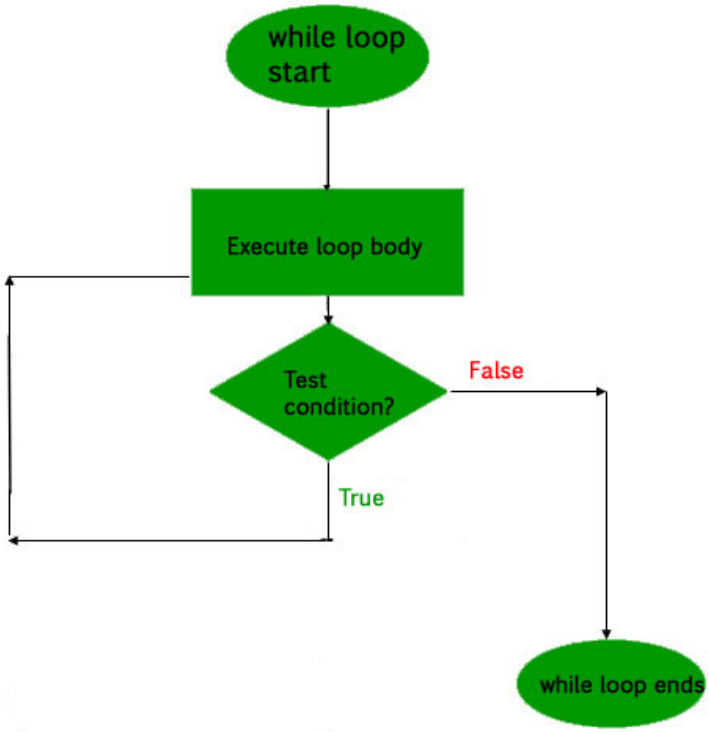
एग्जिट कंट्रोल्ड लूप्स: वे लूप जिनमें परीक्षण की स्थिति लूप बॉडी के अंत में मौजूद होती है, उन्हें **एग्जिट कंट्रोल्ड लूप्स** कहा जाता है । **डू-व्हाइल** एक निकास नियंत्रित लूप है। **नोट:** एग्जिट कंट्रोल्ड लूप्स में, लूप बॉडी का मूल्यांकन कम से कम एक बार किया जाएगा क्योंकि परीक्षण की स्थिति लूप बॉडी के अंत में मौजूद होती है। **1. do-while लूप** do while लूप while लूप के समान है, एकमात्र अंतर यह है कि यह स्टेटमेंट को निष्पादित करने के बाद स्थिति की जांच करता है, अर्थात यह लूप बॉडी को एक बार निश्चित रूप से निष्पादित करेगा क्योंकि यह स्टेटमेंट को निष्पादित करने के बाद स्थिति की जांच करता है। **वाक्य - विन्यास :**

करना

{

बयान..

}जबकि (स्थिति);



फ़्लोचार्ट:

उदाहरण:

- सी तेज

```
// C# program to illustrate do-while loop
```

```
using System;



class dowhileloopDemo

{

    public static void Main()

    {

        int x = 21;

        do

        {

            // The line will be printed even

            // if the condition is false

            Console.WriteLine("GeeksforGeeks");

            x++;

        }

        while (x < 20);

    }

}
```

आउटपुट:
गीक्सफॉरगीक्स

अनंत लूप: वे लूप जिनमें परीक्षण की स्थिति गलत का मूल्यांकन नहीं करती है, वे हमेशा के लिए कथनों को निष्पादित करते हैं जब तक कि इसे समाप्त करने के लिए किसी बाहरी बल का उपयोग नहीं किया जाता है और इस प्रकार उन्हें अनंत लूप के रूप में जाना जाता है। **उदाहरण:**

- सी तेज

```
// C# program to demonstrate infinite loop

using System;



class infiniteLoop

{

    public static void Main()

    {

        // The statement will be printed

        // infinite times

        for(;;)

        Console.WriteLine("This is printed infinite times");

    }

}
```

आउटपुट:

यह अनन्त बार छपा है

यह अनन्त बार छपा है

यह अनन्त बार छपा है

यह अनन्त बार छपा है

यह अनन्त बार छपा है

यह अनन्त बार छपा है

यह अनन्त बार छपा है

.....

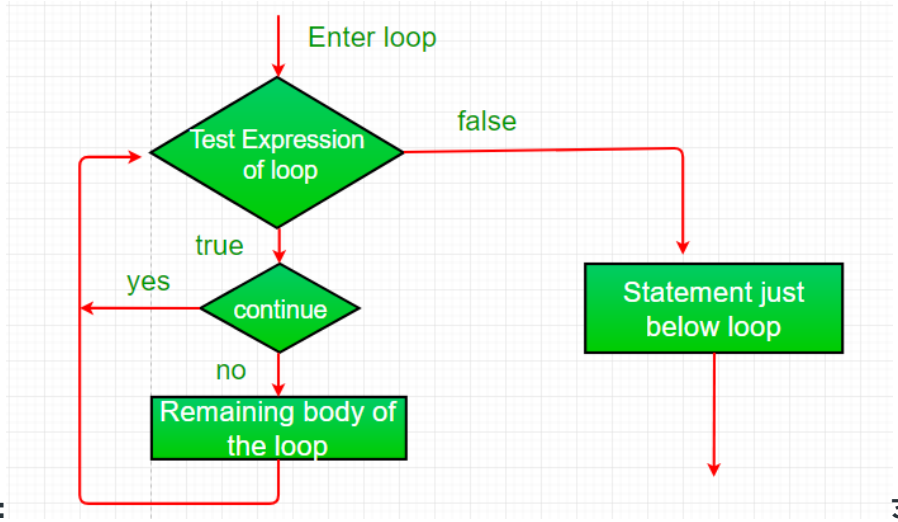
नेस्टेड लूप्स: जब लूप्स अन्य लूप्स के अंदर मौजूद होते हैं, तो इसे नेस्टेड लूप्स के रूप में जाना जाता है। **उदाहरण:**

- सी तेज

```
// C# program to demonstrate nested loops

using System;



class nestedLoops

{

    public static void Main()

    {

        // loop within loop printing GeeksforGeeks

       for(int i = 2; i < 3; i++)

             for(int j = 1; j < i; j++)

                 Console.WriteLine("GeeksforGeeks");

    }

}
```

आउटपुट:
गीक्सफॉरगीक्स

जारी रखें कथन: जारी रखें कथन का उपयोग एक निश्चित स्थिति पर लूप के निष्पादन भाग को छोड़ने और प्रवाह को अगले अद्यतन भाग में ले जाने के लिए किया जाता है। **फ़्लोचार्ट:**



उदाहरण:

- सी तेज

```
// C# program to demonstrate continue statement

using System;



class demoContinue

{

    public static void Main()

    {

        // GeeksforGeeks is printed only 2 times

        // because of continue statement

        for(int i = 1; i < 3; i++)

        {

            if(i == 2)

              continue;



            Console.WriteLine("GeeksforGeeks");

        }

    }

}
```

आउटपुट:
गीक्सफॉरगीक्स

समय जटिलता: O(1)

सहायक स्थान: O(1)

Class and Object in C#

C# एक object-oriented प्रोग्रामिंग है. C# में सब कुछ class और object के साथ जुड़ा हुआ रहता है. एक Class यूजर के द्वारा define किया हुआ एक prototype होता है जिसमें से objects को create किया जाता है. इसमें program को objects और class का प्रयोग करके design किया जाता है.

Class in C# in Hindi – क्लास क्या है?

- C# में, Class एक blueprint होता है जिसमें से objects को create किया जाता है.
- Class समान प्रकार के objects का एक group (समूह) होता है.
- एक क्लास के पास methods, fields और constructors आदि होते हैं.
- एक क्लास properties और methods का encapsulation होता है जिसका प्रयोग real time entity को प्रस्तुत करने के लिए किया जाता है.
- Class को create करने के लिए **class** keyword का प्रयोग किया जाता है.
- C# में, क्लास polymorphism और inheritance को भी support करती है और ये derived class और base class के कांसेप्ट को भी प्रदान करती है.

C# में Class को create करना

C# में एक क्लास को **class** कीवर्ड के द्वारा create या declare किया जाता है और इसके बाद इसमें class का name होता है. इसका syntax निम्नलिखित है:-

```
class Student
{
string name = "Yugal";
}
```

इसमें हमने Student नाम की class को create किया है और इसमें name एक variable है.

- C# Interface in Hindi – इंटरफ़ेस क्या है?
- C# data types in Hindi

Object in C# in Hindi – ऑब्जेक्ट क्या है?

- C# में, Object एक real time entity होता है. **उदाहरण के लिए** – pen, chair, laptop, mobile, car आदि.
- object एक entity है जिसके पास behavior और state होता है. इसमें state का मतलब है data और behavior का मतलब है functionality.

- यह एक runtime entity है क्योंकि इसे run-time में create किया जाता है.
- ऑब्जेक्ट class का एक instance (उदाहरण) होता है. class के सभी members को object के द्वारा access किया जा सकता है.

Object को create करना

object को एक class में से create किया जाता है. हमने ऊपर Student नाम की class को पहले से ही create किया हुआ है. अब हम इस class से object को create करेंगे.

Student के object को create करने के लिए, object name के साथ class name को specify किया जाता है. और new keyword का प्रयोग किया जाता है. इसका example नीचे दिया गया है:-

```
class Student
{
string name = "Yugal";

static void Main(string[] args)
  {
 Student myObj = new Student();
 Console.WriteLine(myObj.color);
  }
}
```

C# Class और Object का Example

नीचे आपको C# में object और class का program दिया गया है:-

```
using System;
public class Student
{
int id;
String name;

public static void Main(string[] args)
{
// creating an object of Student

Student s1 = new Student();
s1.id = 150;
s1.name = "Yugal Joshi";
Console.WriteLine(s1.id);
Console.WriteLine(s1.name);

}
}
```

इसका आउटपुट –

150
Yugal Joshi

C# array in hindi

जैसा की आप जानते है की array को similar type**(int,float,char)** के elements का collection होता है array को आप configuous मेमोरी locations में स्टोर कर सकते है array में सबसे basic और simple data structure होता है arrays में multiple variables को क्रिएट करने की समस्या को solve करता है
array में सभी मेमोरी location को एक index number से identify किया जाता है arrays को **index zero** से start होती है और सभी मेमोरी में location को index numbers के माध्यम से एक independent variables की तरह ही प्रयोग किया जाता है example के लिए यदि आप array को indexes को इस प्रकार से प्रयोग कर सकते है

```
arrayName[0];

arrayName[1];

arrayName[2];

...

...

...

arrayName[n];
```

यदि आपके array की साइज़ 10 है तो आखिर में **elements 9th position** पर ही स्टोर होगा ऐसा इसलिए होता है की array की always indexing 0 से शुरू होती है

- Single Dimensional
- Multi Dimensional
- Jagged Arrays

C# में भी arrays उसी प्रकार से काम करता है जिस प्रकार से दूसरी programming languages में काम करते है लेकिन c# में array को declaration syntax दूसरी programming languages से अलग होता है example के लिए यदि आप c# में **brackets [] type** के बाद लगाए जाते है तो दूसरी popular languages में brackets array name के बाद brackets लगाते है तो प्रोग्राम compile time error दिखता है
c# में arrays में दूसरी languages से एक और major difference यह भी पाया जाता है की c# में arrays में आप चाहे to array की साइज़ डिक्लेअर किये बिना ही directly उसमे वैल्यूज को add कर सकते है इसके बारे में आपको आगे details से बताया गया है

declaring c# arrays

c# में arrays को डिक्लेअर करने का basic syntax आपको निचे बताया जा रहा है

```
type[] arrayName;
```

c# में array को declare करने के लिए सबसे पहले आप जिस type का array को क्रिएट करना चाहते है तो वह type डिफाइन करते है type के बाद में **brackets []** को डिफाइन किये जाते है आखिर में array का unique नाम दिया जाता है इसका example आपको निचे दिया जा रहा है

```
int [] empld;
```

इसका statement से array को सिर्फ डिक्लेअर किया जाता है

आप चाहे तो multi dimensional array भी क्रिएट कर सकते है इस तरह के array में values row और column के format में स्टोर की जाती है multi dimensional array को क्रिएट करने का basic syntax आपको निचे दिया जा रहा है

```
type[,] arrayName
```

multi dimensional array को क्रिएट करते समय आप brackets में एक comma (,) लगाते है ये comma compiler को बताता है की ये एक multi dimensional array है ये comma rows और columns की संख्या को separate करता है इसे आप example के माध्यम से समझ सकते है

```
int[] empid
```

आप jagged arrays को भी क्रिएट कर सकते है इस तरह के arrays में एक array के अन्दर आप दूसरी array को क्रिएट किया जाता है इसका syntax आपको निचे दिया जा रहा है

```
type[][] arrayName;
```

creating c# arrays

आप arrays को डिक्लेअर करने के बाद आप उन्हें क्रिएट करते है की जब आप array को क्रिएट करते है तो तब ही array को मेमोरी allocate की जाती है array को क्रिएट करने के लिए आप new keyword का प्रयोग करते है new keyword के बाद आप brackets में array की साइज़ देते है इसका example आपको निचे दिया जा रहा है

```
int[] empld=new int[5];
```

इसी प्रकार आप multidimensional और jagged arrays को भी क्रिएट कर सकते है

initializing c# arrays

c# में arrays को कई तरह से डिफाइन कर सकते है आप चाहे to compiler syntax का प्रयोग कर सकते है और आप चाहे to short syntax का भी प्रयोग कर सकते है complete syntax का प्रयोग करते हुए आप array को इस प्रकार से initialize कर सकते है

```
int[] empld=new int[5] {1,2,3,4,5,6};
```

आप चाहे तो बिना array की साइज़ को डिफाइन किये भी array को initialize कर सकते है ऐसी सिचुएशन में आप array में जितने भी एलिमेंट डालेंगे तो array की साइज़ उतनी ही हो जाएगी

```
int[] empld=new int[] {1,2,3,4,5,6};
```

यदि आप चाहे तो बिना new ऑपरेटर के भी array को initialize कर सकते है इसका आपको example निचे दिया जा रहा है

```
int empld={1,2,3,4,5,6};
```

पुरे array को एक साथ initialize करने के बजाय आप चाहे तो एक एक index को भी initialize कर सकते है जिसका आपको example निचे दिया जा रहा है

```
empld[0]=1;

empld[1]=2;

empld[2]=3;

empld[3]=4;
```

accessing array members

array members को आप index number से भी एक्सेस कर सकते है example के लिए यदि आप किसी variables को एक array elements को assign करना चाहते है तो ऐसा आप इस प्रकार कर सकते है

```
int num=emplD[3];
```

यदि आप किसी array एलिमेंट को प्रिंट करना चाहते है तो आप ऐसा भी आप index number के द्वारा भी ही कर सकते है इसका example आपको निचे दिया जा रहा है

```
Console.WriteLine(empld[3]);
```

using foreach loop

c# में आपको ऐसा loop को provide करती है की आप जो खासकर arrays के लिए बनाया गया है की इस loop को foreach loop कहते है इस loop में आप loop कण्ट्रोल variables और array का नाम pass करते है ये loop pass किये गया array को end तक traverse करता है इस प्रकार loop में in keyword को भी प्रयोग किये जाते है जो बताता है की आप कौन सा array को traverse किया जायेगा इस loop में आपको variables को increment करने की आवश्यकता नहीं होती है

इस loop के द्वारा आप array में values को स्टोर भी कर सकते है और array की values को print भी करवा सकते है इस loop का basic syntax आपको निचे दिया जा रहा है

```
foreach(variable in arrayName)

{

}
```

तो आएये आपको loop को एक example से समझने का प्रयास करते है

```
foreach(int i in empld)

{

    Console.WriteLine(i);

}
```

Array डेटा सारणी का प्रयोग डेटा ऑब्जेक्टिव्स के समूह को एकत्रित करने के लिए किया जाता है। "सरणी एक स्थिर डेटा स्थिरांक है अर्थात हम केवल संकलन समय में ही स्मृति को आवंटित कर सकते हैं और इसे रन-टाइम में बदल नहीं सकते हैं।"

सारणी के प्रकार हिंदी में:-

सारणी निम्नलिखित तीन प्रकार की होती है:-

1:- एक आयामी सरणियाँ।

2:- द्विविमीय सारणियाँ।

3:- बहु आयामी सारणियाँ।

1:- एक आयामी(1-डी) सारणी:-

वह arrays उदाहरण केवल एक सबस्क्रिप्ट होता है उसे एक आयामी arrays कहता है। इसका प्रयोग रैखिक रूप में डेटा को स्टोर करने के लिए किया जाता है।

2:- द्वि-आयामी(2-डी) सारणी:-

वह सारणी जिसमें दो सबस्क्रिप्ट होती है, उसे द्वि-आयामी सारणी कहा जाता है। द्वि-आयामी सरणियों को मैट्रिक्स और तालिका भी कहते हैं।

3:- बहु आयामी (एमडी) सारणी: -

वह सारणी जिसमें दो से अधिक सबस्क्रिप्ट होती है, वह बहु-आयामी सारणी होती है।

Concepts of OOPS in Hindi – OOPS

OOP का पूरा नाम object-oriented programming (ऑब्जेक्ट ओरिएंटेड प्रोग्रामिंग) है। OOPS concepts निम्नलिखित होते है:-

1. Object
2. Class
3. Encapsulation
4. Abstraction
5. Inheritance
6. Polymorphism
7. Message Passing

Object

ऑब्जेक्ट, class का instance होता है जो कि variable के स्थान पर वास्तविक value को contain किये रहता है।

ऑब्जेक्ट एक बेसिक run-time entity होती है।

सामन्यतया, Object वह प्रत्येक वस्तु होती है जिसको कि पहचाना जा सकें। हमारी आसपास की सारी वस्तुएँ जैसे:-पेन, किताब, कुर्सी, गाडी, टीवी आदि सभी ऑब्जेक्ट्स है।

Class

क्लास एक ही तरह के objects का समूह होता है। जैसे:- आम, अमरुद तथा सेब आदि ये सभी फल है, और ये सभी class fruit के सदस्य हुए।
क्लास यूजर-डिफाइंड डेटा टाइप होता है तथा क्लास data तथा functions का समूह होता है।

इसे पूरा पढ़ें:- class और object क्या है?

Encapsulation

डेटा तथा फंक्शन को एक ही यूनिट में सम्मिलित करना (जोड़ना) Encapsulation कहलाता है। इसमें class के variables प्राइवेट होते हैं और इन्हें class के बाहर direct access नहीं किया जा सकता. Encapsulation को class के रूप में use किया जाता है. एक class में हम data और methods को एक यूनिट के रूप में एक साथ रख सकते हैं.
Java bean पूरी तरह से एक encapsulated class होती है क्योंकि इसमें सभी data members प्राइवेट होते हैं.

Abstraction

Abstraction का अर्थ है कि object के केवल आवश्यक जानकारी को प्रदर्शित करना तथा background की जानकारी को छुपाये रखना।
उदाहरण के लिए– जब हम कोई car चलाते है तो हमें यह पता होता है कि जब accelarator को दबायेंगे तो speed बढ़ेगी और जब break दबायेंगे तो Car रुक जायेगी. लेकिन हमें यह नहीं पता होता कि break दबाने से car क्यों रुक जाती है.
इसी प्रकार OOPS में complex (कठिन) चीजों को छुपा दिया जाता है और केवल आवश्यक तथा simple चीजों को show किया जाता है. Java में, abstraction को प्राप्त करने के लिए abstract class और interface का प्रयोग किया जाता है.

Inheritance

inheritance का अर्थ है 'विरासत'।
जावा में एक क्लास के द्वारा दूसरी क्लास के properties(गुणों) तथा methods को inherit कर लेना inheritance कहलाता है।

वह क्लास जो दूसरी क्लास से derived होती है वह subclass कहलाती है तथा वह क्लास जिससे subclass derived हुई होती है वह super class कहलाती है।
Superclass को हम base class भी कहते है तथा subclass को हम derived class भी कहते है।

inheritance को पूरा पढने के लिए क्लिक करें.

Polymorphism

polymorphism ग्रीक भाषा से लिया गया शब्द है जिसमें poly का अर्थ है many और morphism का अर्थ है forms. तो polymorphism का अर्थ हुआ many forms.

Polymorphism एक ऐसा concept है जिसमें हम एक ही काम को दो भिन्न तरीके से कर सकते है।

जावा में Polymorphism दो तरह की होती है जो निम्न है:-

1:- Compile-time polymorphism (static polymorphism)
2:- Run-time polymorphism (Dynamic polymorphism)

1:- Compile time polymorphism:- Compile time polymorphism को हम method overloading या early binding भी कहते है।
इस polymorphism का अर्थ है कि हम समान नाम के methods को different signatures के साथ declare करते है क्योंकि हम अलग-अलग task को एक ही method name के साथ perform क्र सकते है।

2:- Run-time polymorphism:-इस प्रकार के polymorphism को late binding या dynamic binding या method overriding कहते है।
इस polymorphism का अर्थ है कि हम समान नाम के methods को समान signature के साथ declare करते है।

Message Passing

Objects एक दूसरे को information (सूचना) send तथा receive करके आपस में communicate करते हैं. objects उसी प्रकार message को send तथा receive करते है जिस प्रकार हम लोग करते हैं.
एक ऑब्जेक्ट के लिए message एक procedure (प्रक्रिया) के लिए request होता है और इसलिए receiving object में एक method को invoke किया जाता है.

इसे पूरा पढ़ें:- message passing क्या है?

Advantage of OOP in Hindi

इसके लाभ निम्नलिखित होते हैं:-

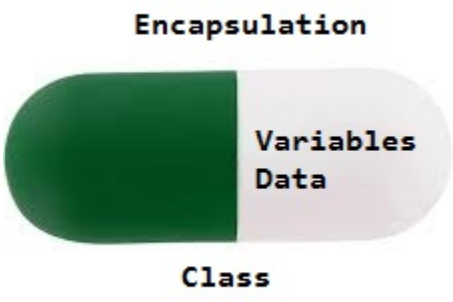
1. इसमें program का structure बहुत ही simple होता है जिससे complexity कम होती है.
2. हमें इसमें सिर्फ एक बार code को लिखने की जरूरत होती है और उसे हम बार-बार use कर सकते हैं.
3. यह data redundancy प्रदान करता है.
4. इसमें हम आसानी से code को maintain किया जा सकता है जिससे time की बचत होती है.

5. object-oriented programming में data hiding और abstarction का प्रयोग किया जाता है जिससे इसमें security बेहतर बन जाती है.
6. इसमें debugging करनी हो तो इसे आसानी से किया जा सकता है.

सी# | कैप्सूलीकरण

एनकैप्सुलेशन को एक इकाई के तहत डेटा और सूचना को लपेटने के रूप में परिभाषित किया गया है। यह वह तंत्र है जो डेटा और उनमें हेरफेर करने वाले कार्यों को एक साथ बांधता है। एक अलग तरीके से, एनकैप्सुलेशन एक सुरक्षा कवच है जो डेटा को इस ढाल के बाहर कोड द्वारा एक्सेस होने से रोकता है।

- तकनीकी रूप से एनकैप्सुलेशन में, किसी वर्ग के चर या डेटा किसी अन्य वर्ग से छिपे होते हैं और केवल अपने वर्ग के किसी भी सदस्य फ़ंक्शन के माध्यम से उन तक पहुंचा जा सकता है जिसमें उन्हें घोषित किया जाता है।
- जैसा कि एनकैप्सुलेशन में, एक क्लास का डेटा अन्य क्लासों से छिपा होता है, इसलिए इसे **डेटा-हिडिंग** के रूप में भी जाना जाता है ।
- **इनकैप्सुलेशन को इसके द्वारा प्राप्त किया जा सकता है:** क्लास में सभी वेरिएबल्स को निजी घोषित करना और वेरिएबल्स के मान सेट करने और प्राप्त करने के लिए क्लास में **C# प्रॉपर्टीज का उपयोग करना।**



एनकैप्सुलेशन के लाभ:

- **डेटा छिपाना:** उपयोगकर्ता को कक्षा के आंतरिक कार्यान्वयन के बारे में कोई जानकारी नहीं होगी। यह उपयोगकर्ता को दिखाई नहीं देगा कि क्लास वेरिएबल्स में मानों को कैसे संग्रहीत करता है। वह केवल यह जानता है कि हम एक्सेसर्स को मान दे रहे हैं और वेरिएबल को उस मान पर आरंभीकृत किया जा रहा है।
- **बढ़ी हुई लचीलापन:** हम अपनी आवश्यकता के आधार पर कक्षा के वेरिएबल्स को केवल पढ़ने या लिखने के लिए बना सकते हैं। यदि हम वेरिएबल्स को केवल-पढ़ने के लिए बनाना चाहते हैं तो हमें कोड में केवल Get Accessor का उपयोग करना होगा। यदि हम वेरिएबल्स को राइट-ओनली बनाना चाहते हैं तो हमें केवल सेट एक्सेसर का उपयोग करना होगा।
- **पुन: प्रयोज्यता:** एनकैप्सुलेशन पुन: प्रयोज्यता में भी सुधार करता है और नई आवश्यकताओं के साथ बदलने में आसान होता है।

- **परीक्षण कोड आसान है:** इकाई परीक्षण के लिए इनकैप्सुलेटेड कोड का परीक्षण करना आसान है।

एनकैप्सुलेशन ऑब्जेक्ट-ओरिएंटेड प्रोग्रामिंग (ओओपी) में एक मौलिक अवधारणा है जो डेटा के बंडलिंग और उन तरीकों को संदर्भित करता है जो एक इकाई के भीतर उस डेटा पर काम करते हैं। C# में, यह आमतौर पर कक्षाओं के उपयोग के माध्यम से हासिल किया जाता है।

```
public class BankAccount {
    private decimal balance;

    public BankAccount(decimal initialBalance)
    {
        balance = initialBalance;
    }

    public void Deposit(decimal amount)
    {
        balance += amount;
    }

    public void Withdraw(decimal amount)
    {
        if (balance >= amount) {
            balance -= amount;
        }
        else {
            Console.WriteLine("Insufficient funds.");
        }
    }

    public decimal GetBalance() { return balance; }
}
```

```
class Program {
        static void Main(string[] args)
        {
                BankAccount myAccount = new BankAccount(1000);


                myAccount.Deposit(500);
                Console.WriteLine("Balance: "
                                        + myAccount.GetBalance());


                myAccount.Withdraw(2200);
                Console.WriteLine("Balance: "
                                        + myAccount.GetBalance());
        }
}
```

C# Inheritance in Hindi

C# में, Inheritance एक ऐसी प्रक्रिया है जिसके द्वारा एक old class से new class को create किया जाता है। इस के द्वारा old class की properties को new class में प्रयोग किया जा सकता है।

Old class की properties को new class में use करने के लिए old class को inherit करना पड़ता है और किसी class को inherit करने के लिए public , private और protected एक्सेस मॉडिफाएर का प्रयोग किया जाता है।

Inheritance मुख्य रूप से inherit शब्द (word) से बना हुआ है जिसका अर्थ है विरासत में पाना.

Inheritance में old class को base class या parent class या super class कहा जाता है, और new class को child class या derived class या sub class कहा जाता है।

C# में Inheritance के फायदे

इसके फायदे निम्नलिखित हैं:-

1. R**eusability –**inheritance के द्वारा आप एक class के code को दूसरी class में प्रयोग कर पाते है इससे आपके code की reusability बढती है.

2. R**eadability–** inheritance को प्रयोग करने से extra code लिखने की आवश्यकता नहीं होती है जिससे आपके प्रोग्राम में code कम हो जाती है जो readability को बढ़ता है

3. S**ave programmers time –**inheritance का उपयोग करने से programmer का time बचता है क्योकि एक ही code को बार बार लिखना नहीं पढ़ता है.

C# Inheritance का syntax

```
class derived-class : base-class
{
  // methods and fields
  .
  .
}
```

Types of Inheritance in C# in Hindi – C# में इनहेरिटेंस के प्रकार

C# में, INHERITANCE तीन प्रकार का होता है:-

1. Single inheritance
2. multilevel inheritance
3. hierachical inheritance

Single inheritance

इस इनहेरिटेंस में, एक class को सिर्फ एक ही class के द्वारा inherit किया जा सकता है कोई भी class एक से अधिक classes को inherit नहीं कर सकती है. नीचे आप इसके चित्र को देख सकते हैं:-

```
using System;
  public class Animal
   {
      public void eat() { Console.WriteLine("Eating..."); }
  }
  public class Dog: Animal
  {
      public void bark() { Console.WriteLine("Barking..."); }
  }
  class TestInheritance2{
      public static void Main(string[] args)
       {
          Dog d1 = new Dog();
          d1.eat();
          d1.bark();
       }
   }
```

Multilevel inheritance

वह इनहेरिटेंस जिसमें एक से अधिक class एक level में एक दूसरे को inherit करते है तो उस इनहेरिटेंस को multilevel inheritance कहते है। इसमें एक sub class दुसरे class के लिए base class की तरह कार्य करती है.

इसका चित्र नीचे आप देख सकते हैं:-

Hierarchical inheritance

जब एक base class को एक से अधिक sub class द्वारा inherit किया जाता है तो उस इनहेरिटेंस को hierarchical inheritance कहते है।

इसका चित्र –

इसका example –

```
class A  //base class
{
   public string msg()
   {
      return "this is A class Method";
   }
}
class B : A
{
   public string info()
   {
      msg();
      return "this is B class Method";
   }
   class C : A
   {
      public string getinfo()
      {
         msg();
         return "this is B class Method";
      }
   }
}
```