



MODUL WORKSHOP

C# PROGRAMMING LANGUAGE FUNDAMENTALS



Laboratorium Pengembangan
Aplikasi dan Pemrograman Komputer
LEMBAGA PENGEMBANGAN KOMPUTERISASI
UNIVERSITAS GUNADARMA



KATA PENGANTAR

Modul ini merupakan panduan bagi peserta C# Programming Language Fundamentals yang diselenggarakan oleh Laboratorium Pengembangan Aplikasi dan Pemrograman LePKom Universitas Gunadarma.

Topik-topik yang dibahas dalam modul adalah Pengenalan Dasar Dasar bahasa C# dan Object Oriented Programming. Modul ini juga dilengkapi dengan contoh-contoh serta studi kasus yang menarik untuk dipelajari.

Penyusun berharap modul ini dapat digunakan sebaik-baiknya oleh seluruh peserta dan dapat membantu peserta dalam mempelajari dan memahami C# .

Tim Penyusun



Daftar Isi

Kata Pengantar.....	i
Daftar Isi.....	ii
Bab 1 Pendahuluan	1
1.1. Pengertian C#.....	1
1.2. Pengenalan Net.Framework.....	1
1.3. Pertandingan Java dan C#.....	5
Bab 2 Pemrograman C#	7
2.1. Variable.....	7
2.2. Operator.....	9
2.3. Keywords (Kata Kunci).....	11
Bab 3 Percabangan(if, if-else, if-elseif-else).....	13
3.1. Pernyataan if.....	13
3.2. Pernyataan if-else.....	14
3.3. Pernyataan if-else if-else.....	15
Bab 4 Percabangan(switch).....	17
4.1. Switch.....	17
Bab 5 Looping(Perulangan).....	20
5.1. Looping.....	20
5.2. For.....	20
5.3. While.....	22
Bab 6 Continue.....	25
6.1. Continue.....	25
Bab 7 Break.....	27
7.1. Break.....	27
Bab 8 Komentar.....	30
8.1. Komentar.....	30
Bab 9 Function.....	31
9.1. Function.....	31
9.2. Call by Value.....	34
9.3. Call by Reference.....	35

9.4. Out Parameter.....	36
Bab 10 Array.....	37
10.1. Array.....	37
10.2. Array Satu Dimensi.....	37
10.3. Array Multi Dimensi.....	38
10.4. Jagged Array.....	49
Bab 11 OOP(Object Oriented Programming).....	41
11.1. Object.....	41
11.2. Class.....	42
11.3. Constructor.....	43
11.4. Destructor.....	44
11.5. This.....	45
11.6. Static.....	46
11.7. Inheritance.....	49
11.8. Polymorphism.....	52
11.9. Abstract.....	54
11.10. Encapsulation.....	55
11.11 File I/O.....	57

1 Pendahuluan

Obyektif :

- Mengetahui Bahasa Pemrograman C#
 - Mengetahui .Net Framework
 - Mengetahui Perbandingan Java, C++, dan C#
-

1.1. Pengertian C#

C# dapat diucapkan dengan "C-Sharp" merupakan bahasa pemrograman berorientasi objek yang disediakan oleh Microsoft yang berjalan pada .NET Framework. Dengan bahasa C#, kita dapat mengembangkan berbagai jenis aplikasi yang aman dan kuat, seperti :

- Aplikasi Window
- Website
- Distributed applications
- Aplikasi Web service
- Aplikasi Database
- Dan masih banyak lagi

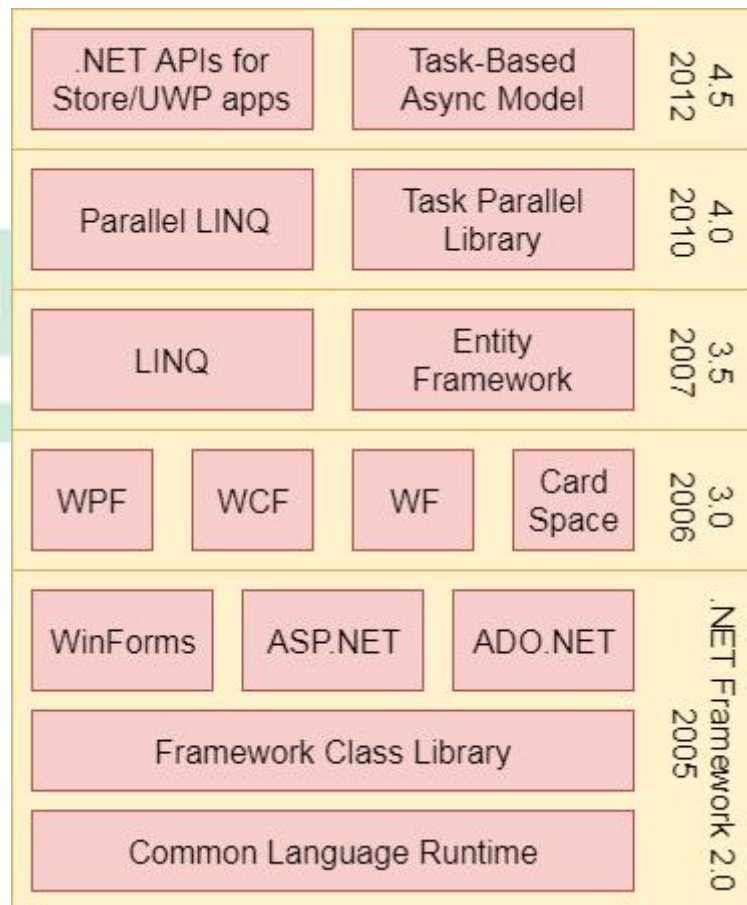
C# disetujui sebagai standar oleh ECMA dan ISO. C# dirancang untuk CLI (Common Line Interface). CLI adalah spesifikasi yang menjelaskan kode yang dapat dieksekusi dan lingkungan runtime. C# dipengaruhi oleh bahasa C++, Java, Eiffel, Modula-3, Pascal, dll.

1.2. Pengenalan Net.Framework

.NET adalah kerangka kerja yang digunakan untuk mengembangkan aplikasi perangkat lunak yang dirancang dan dikembangkan oleh Microsoft dan versi beta pertama dirilis pada 2000.

.Net Framework digunakan untuk membangun aplikasi untuk web, Windows, telepon dan menyediakan berbagai fungsi dan dukungan untuk standar industri. Kerangka kerja ini berisi sejumlah besar perpustakaan kelas yang dikenal sebagai Framework Class Library (FCL).

Program perangkat lunak yang ditulis dalam .NET dijalankan di lingkungan eksekusi, yang disebut CLR (Common Language Runtime) yang merupakan bagian penting dari .NET Framework. Kerangka kerja ini menyediakan berbagai layanan seperti manajemen memori, Jaringan, keamanan, dan keamanan memori. .NET juga mendukung banyak bahasa pemrograman seperti C#, F#, VB, dll. Berikut ini adalah .NET Framework stack yang menunjukkan modul dan komponen kerangka kerja.



Gambar 1.1 modul dan komponen kerangka kerja pada C#

CLR (Common Language Runtime)

CLR adalah mesin eksekusi program yang memuat dan mengeksekusi program dan bertindak sebagai antarmuka antara kerangka kerja dan sistem operasi.

FCL (Framework Language Runtime)

FCL adalah *standard library* yang merupakan kumpulan dari ribuan *class* dan digunakan untuk membangun sebuah aplikasi. BCL (Base Class Library) adalah inti dari FCL yang menyediakan fungsi dasar.

WinForms

Windows Forms adalah teknologi klien pintar untuk .NET Framework, seperangkat Library yang menyederhanakan tugas aplikasi umum seperti membaca dan menulis ke file sistem.

ASP .NET

ASP .NET adalah kerangka web yang dirancang dan dikembangkan oleh Microsoft digunakan untuk mengembangkan situs web, aplikasi web, dan layanan web. ASP.NET sudah terintegrasi dengan HTML, CSS, dan JavaScript. Pertama kali dirilis pada 2002 Januari.

ADO .NET

ADO .NET adalah modul .NET Framework, yang digunakan untuk menghubungkan antara Data Source (sumber data) dengan aplikasi. Data Source dapat berbentuk SQL Server dan XML. ADO .NET terdiri dari kelas yang dapat digunakan untuk menghubungkan, mengambil, memasukkan, dan menghapus data.

WPF (Windows Presentation Foundation)

Windows Presentation Foundation (WPF) adalah subsistem grafis yang dikembangkan oleh Microsoft untuk pembuatan antarmuka pengguna di aplikasi berbasis Windows. Dahulu, WPF dikenal sebagai "Avalon ", awalnya dirilis sebagai bagian dari .NET Framework 3.0 di 2006. WPF menggunakan DirectX.

WCF (Windows Communication Foundation)

WCF merupakan kerangka kerja untuk membangun aplikasi berorientasi layanan. Menggunakan WCF, Anda dapat mengirim data sebagai pesan asinkron dari satu titik akhir layanan lain.

LINQ (Language Integrated Query)

LINQ merupakan bahasa query, diperkenalkan di .NET 3,5 Framework. Hal ini digunakan untuk membuat query untuk sumber data dengan C# atau bahasa pemrograman visual Basics.

Entity Framework

Entity Framework merupakan kerangka open source berbasis ORM yang digunakan untuk bekerja dengan database menggunakan objek .NET. Entity Framework menghilangkan banyak upaya pengembang untuk menangani database. Ini merupakan teknologi yang direkomendasikan Microsoft untuk menangani database.

Parallel LINQ

Parallel LINQ atau PLINQ merupakan implementasi paralel LINQ ke objek. PLINQ menggabungkan kesederhanaan dan pembacaan LINQ dan memberikan kekuatan pemrograman paralel.

Hal ini dapat meningkatkan dan menyediakan kecepatan cepat untuk mengeksekusi query LINQ dengan menggunakan semua kemampuan komputer yang tersedia. Selain dari fitur dan Perpustakaan di atas, .NET mencakup API dan model lainnya untuk meningkatkan .NET Framework.

1.3. Perbandingan Java, C++, dan C#

Tabel 1.1 Tabel Perbandingan C# dan C++

NO	C++	C#
1	C++ adalah bahasa pemrograman case-sensitive, bahasa pemrograman <i>free-form</i> yang mendukung pemrograman berorientasi objek, prosedural dan generik.	C# diucapkan sebagai "C-Sharp ". Ini adalah bahasa pemrograman berorientasi objek yang disediakan oleh Microsoft yang berjalan pada .NET Framework.
2	Dalam C++, beberapa pewarisan dimungkinkan melalui kelas.	Dalam C#, beberapa enkapsulasi (warisan) tidak melalui kelas.
3	Dalam C++, manajemen memori ditangani secara manual.	Dalam C#, manajemen memori ditangani secara otomatis.
4	Dalam C++, pointer dapat digunakan di mana saja dalam sebuah program.	Dalam C#, pointer dapat digunakan hanya dalam mode tidak aman (Unsafe Mode).
5	Pemrograman C++ didasarkan pada konsep OOPs.	Pemrograman C# didasarkan pada konsep komponen dan OOPs.
6	C++ adalah bahasa pemrograman yang berjalan pada semua platform.	C# adalah bahasa pemrograman yang jarang digunakan di luar Windows.
7	Pemrograman C++ dapat digunakan untuk membuat aplikasi konsol.	C# dapat digunakan untuk membuat aplikasi konsol, aplikasi Windows, aplikasi mobile, dll.

Tabel1.2 Tabel Perbandingan C# dengan Java

No	JAVA	C#
1	Java adalah bahasa pemrograman tingkat tinggi, kuat, aman dan berorientasi objek yang dikembangkan oleh Oracle.	C# adalah bahasa pemrograman berorientasi objek yang dikembangkan oleh Microsoft yang berjalan pada .NET Framework.
2	Bahasa pemrograman java dirancang untuk dijalankan pada platform Java, dengan bantuan Java Runtime Environment (JRE).	Bahasa pemrograman C# dirancang untuk dijalankan pada <i>Common Language Runtime (CLR)</i> .
3	Type-safety pada Java bersifat aman.	Type-safety pada C# bersifat tidak aman.
4	Pada Java, tipe data built-in yang dilewatkan oleh nilai disebut tipe data primitif .	Dalam C#, tipe data built-in yang disampaikan oleh nilai disebut <i>simple types</i> .
5	Array pada Java adalah spesialisasi langsung dari object.	Array di C# adalah spesialisasi sistem.
6	Java tidak mendukung kompilasi bersyarat.	C# mendukung kompilasi bersyarat menggunakan arahan preprocessor.
7	Java tidak mendukung pernyataan “goto”.	C# mendukung pernyataan Goto.

2 Pemrograman C#

Obyektif :

- Mengetahui Variabel, Operator dan Keyword

2.1. Variabel

Variabel adalah nama dari lokasi memori. Hal ini digunakan untuk menyimpan data. Nilainya dapat diubah dan dapat digunakan kembali berkali-kali. Variabel adalah cara untuk mewakili lokasi memori melalui simbol sehingga dapat dengan mudah diidentifikasi.

Tipe variabel dasar yang tersedia dalam C# dapat dikategorikan sebagai:

Tabel 2.1. Tabel tipe data

Tipe Variabel	Contoh
Decimal types	decimal
Boolean types	True or false value, as assigned
Integral types	int, char, byte, short, long
Floating point types	float and double
Nullable types	Nullable data types

Mari kita lihat sintaks untuk mendeklarasikan sebuah variabel:

```
type variable_list;
```

Contoh Deklarasi variabel di bawah ini:

```
int i, j;  
double d;  
float f;  
char ch;
```

Di sini, i, j, d, f, ch adalah variabel dan int, double, float, char adalah tipe data.

Kita juga dapat memberikan nilai sambil mendeklarasikan variabel seperti yang diberikan di bawah ini:

```
int i=2,j=4;  
float f=40.2;  
char ch='B';
```

Aturan untuk mendefinisikan variabel

Variabel dapat dibuat dengan huruf abjad, angka dan garis bawah. Sebuah nama variabel dapat dimulai dengan alfabet dan garis bawah saja dan tidak dapat dimulai dengan angka. Tidak boleh ada *whitespace* dalam nama variabel. Nama variabel tidak boleh ada kata atau *Reserved keyword* misalnya char, Float dll.

Nama variabel yang valid:

```
int x;  
int _x;  
int k20;
```

Nama variabel yang tidak diizinkan:

```
int 4;  
int x y;  
int double;
```

2.2. Operator

Operator hanyalah simbol yang digunakan untuk melakukan operasi. Ada banyak jenis operasi seperti aritmatika, logis, bitwise dll.

Berikut adalah jenis operator untuk melakukan berbagai jenis operasi dalam bahasa C#.

- Operator aritmatika
- Operator relasional
- Operator logika
- Bitwise operator
- Operator penugasan
- Unary operator
- Operator ternary
- Operator Misc

	Operator	Type
Binary Operator	+, -, *, /, %	Arithmetic Operators
	<, <=, >, >=, ==, !=	Relational Operators
	&&, , !	Logical Operators
	&, , <<, >>, ~, ^	Bitwise Operators
	=, +=, -=, *=, /=, %=	Assignment Operators
Unary Operator	→ ++, --	Unary Operator
Ternary Operator	→ ?:	Ternary or Conditional Operator

Gambar 2.1. Operator pada C#

Prioritas operator

Prioritas operator menentukan operator mana yang akan dievaluasi terlebih dahulu dan berikutnya. Asosiativitas menentukan arah operator untuk dievaluasi, mungkin dari kiri ke kanan atau kanan ke kiri.

Mari kita pahami contoh yang diberikan di bawah ini:

```
int data = 10 + 5 * 5
```

Variabel "data" akan berisi 35 karena * (operator kali) dievaluasi sebelum + (operator tambah).

Tabel 2.2. Tabel Prioritas Operator

Kategori (menurut prioritas)	Biro wisata	Associativity
Unari	+ - ! ~ + +--(tipe) * & sizeof	Kanan ke kiri
Aditif	+ -	Kiri ke kanan
Perkalian	% / *	Kiri ke kanan
Relasional	< > <= >=	Kiri ke kanan
Shift	<< >>	Kiri ke kanan
Kesetaraan	== !=	Kanan ke kiri
Logical dan	&	Kiri ke kanan
Logical atau		Kiri ke kanan
Logical XOR	^	Kiri ke kanan
Kondisional atau		Kiri ke kanan
Kondisional dan	&&	Kiri ke kanan

Null Coalescing	??	Kiri ke kanan
Ternary	?:	Kanan ke kiri
Assignment	= *= /= %= += -= <<= >>= &= ^= = =>	Kanan ke kiri

2.3. Keywords (kata kunci)

Kata kunci adalah kata yang dicadangkan. Anda tidak dapat menggunakannya sebagai nama variabel, nama konstan, dll.

Dalam C# kata kunci tidak dapat digunakan sebagai pengidentifikasi. Namun, jika kita ingin menggunakan kata kunci sebagai pengidentifikasi, kita mungkin awalan kata kunci dengan karakter @.

Daftar Reserved keyword yang tersedia dalam bahasa pemrograman C# di bawah ini:

abstract	base	as	bool	break	catch	case
byte	char	checked	class	const	continue	decimal
private	protected	public	return	readonly	ref	sbyte
explicit	extern	false	finally	fixed	float	for
foreach	goto	if	implicit	in	in (generic modifier)	int
ulong	ushort	unchecked	using	unsafe	virtual	void
null	object	operator	out	out (generic modifier)	override	params
default	delegate	do	double	else	enum	event
sealed	short	sizeof	stackalloc	static	string	struct
switch	this	throw	true	try	typeof	uint
abstract	base	as	bool	break	catch	case
volatile	while					

Gambar 2.2 Reserved words pada C#

Ada beberapa *identifier* yang memiliki arti khusus dalam konteks kode disebut sebagai **Contextual Keywords**.

Daftar **Contextual Keywords** yang tersedia dalam bahasa pemrograman C# :

add	group	ascending	descending	dynamic	from	get
global	alias	into	join	let	select	set
partial (type)	partial(method)	remove	orderby			

Gambar 2.3 Contextual Keywords



3 Percabangan(if, if-else, if-elseif-else)

Obyektif :

- Mengetahui perbedaan If, If-Else, dan If-Elseif-Else.
-

3.1. Pernyataan if

Pada C# Pernyataan if membutuhkan hasil dengan tipe data boolean, yaitu true atau false. Pada beberapa bahasa pemrograman, beberapa tipe data dapat otomatis diubah ke tipe data boolean, tapi di C# harus membuat hasilnya menjadi tipe boolean.

Syarat statement pada C# :

- Gunakan if untuk kondisi khusus pertama dalam menentukan blok kode yang akan dieksekusi, jika kondisinya benar maka dia akan mengeluarkan nilai *true* atau benar.
- Gunakan else if untuk menentukan kondisi khusus baru yang akan diuji, jika kondisi pertama salah maka kondisi berikut yang akan dieksekusi. Dapat digunakan lebih dari satu else if.
- Gunakan else untuk menentukan blok kode opsi terakhir tanpa kondisi khusus yang akan dieksekusi.

Bentuk Umum Sintaks Pernyataan If

```
if (kondisi) {  
    //Kode yang akan di eksekusi jika kondisi bernilai benar  
}
```

Contoh Pernyataan If

```
using System;
public class IfExample {
    public static void Main(string[] args)
    {
        int num = 10;
        if (num % 2 == 0)
        {
            Console.WriteLine("Ini adalah angka genap");
        }
    }
}
```

Hasil:

```
Ini adalah angka genap
```

3.2. Pernyataan if-else

Pernyataan If-Else digunakan pada saat pernyataan hanya memiliki 2 kondisi yang bernilai benar atau salah. Yang mengharuskan mengeksekusi kondisi khusus If, jika salah maka else yang akan di eksekusi.

Bentuk Umum Sintaks Pernyataan If-Else

```
If (kondisi) {
    //kode yang akan dieksekusi jika kondisi bernilai benar
} else {
    //kode yang akan dieksekusi jika kondisi bernilai salah
}
```

Contoh Pernyataan If-Else

```
using System;
public class IfExample
{
    public static void Main(string[] args)
    {
        int num = 11;
        if (num % 2 == 0)
        {
            Console.WriteLine("Ini adalah angka genap");
        }
        else
```

Hasil :

```
Ini adalah angka ganjil
```

3.3. Pernyataan if-else if-else

Pernyataan else if digunakan jika program yang dibuat memiliki lebih dari 2 kondisi khusus yang akan dieksekusi, sehingga else if tidak hanya satu blok kode. Pernyataan If-Else If-Else dijalankan sesuai urutan sampai sebuah kondisi dinyatakan benar. Jika tidak ada kondisi yang benar dari blok kode if dan else if, maka blok kode else yang dijalankan. Untuk blok kode Else dapat tidak digunakan.

Bentuk Umum Sintaks Pernyataan If-Else If-Else

```
if (kondisi) {
    //kode yang akan dieksekusi jika kondisi pertama bernilai benar
} else (kondisi lain) {
    //kode yang akan dieksekusi jika kondisi kedua bernilai benar
} else {
    //kode yang akan dieksekusi jika semua kondisi bernilai salah
}
```

Contoh Pernyataan If-Else If-Else

```
using System;
public class IfExample
{
    public static void Main(string[] args)
    {
        Console.Write("Masukan Nilai : ");
        int num = Convert.ToInt32(Console.ReadLine());

        if (num >= 85 && num <= 100) {
            Console.WriteLine("Grade A");
        }
        else if (num >= 75 && num < 85) {
            Console.WriteLine("Grade B");
        }
        else if (num >= 65 && num < 75) {
            Console.WriteLine("Grade C");
        }
        else if (num > 1 && num < 65) {
```

Hasil :

Masukkan Nilai : 101
Angka Salah

Masukkan Nilai : 90
Grade A

Masukkan Nilai : 0
Grade E

4 Percabangan (Switch)

Obyektif :

- Menenal Percabangan Switch pada C#
 - Mengetahui Perbedaan Case dan Default.
-

4.1. Switch

Percabangan Switch dapat digunakan menggantikan percabangan else if, dengan mengeksekusi satu pernyataan dari beberapa kondisi.

Bentuk Umum switch

```
switch(ekspresi){  
case value1:  
    //kode yang akan dieksekusi;  
    break;  
case value2:  
    //kode yang akan dieksekusi;  
    break;  
.....  
default:  
    //kode yang akan dieksekusi bila semua case tidak terpenuhi;  
    break;  
}
```

Contoh Pernyataan Switch

```
using System;

public class SwitchExample {
    public static void Main(string[] args) {
        Console.Write("Masukkan alfabet : ");
        char ch = Convert.ToChar(Console.ReadLine());

        switch(Char.ToLower(ch))
        {
            case 'a':
                Console.WriteLine("Huruf Vokal");
                break;
            case 'e':
                Console.WriteLine("Huruf Vokal");
                break;
            case 'i':
                Console.WriteLine("Huruf Vokal");
                break;
            case 'o':
```

Hasil :

```
Masukkan alfabet : a
Huruf Vokal
```

```
Masukkan alfabet : b
Bukan Huruf Vokal
```

Contoh:

```
1  using System;
2  public class LatihanSwitch
3  {
4      public static void Main(string[] args)
5      {
6          Console.WriteLine("Masukan Angka:");
7          int num = Convert.ToInt32(Console.ReadLine());
8
9          switch (num)
10         {
11             case 10:
12                 Console.WriteLine("Ini angka 10");
13                 break;
14             case 20:
15                 Console.WriteLine("Ini angka 20");
16                 break;
17             case 30:
18                 Console.WriteLine("Ini angka 30");
19                 break;
20             default:
21                 Console.WriteLine("Bukan 10, 20 atau 30");
22                 break;
23         }
24     }
25 }
```

Gambar 4.1 contoh switch pada C#

Hasil:

Masukkan Angka: 10

Ini angka 10

5 Looping (Perulangan)

Obyektif :

- Mengetahui perbedaan FOR, WHILE, DO-WHILE
 - Mengenal Looping pada C#
-

5.1. Looping

Looping adalah proses yang dilakukan secara berulang-ulang sampai batas yang ditentukan. Biasanya bila dalam perulangan tersebut tidak disertakan batasnya maka syntax akan error karena proses itu akan berulang terus hingga tak terhingga sementara variabel dalam komputer masih terbatas.

5.2. FOR

For adalah statement perulangan yang paling sering digunakan. Statement for memiliki 3 parameter, yaitu nilai awal (initial value), tes kondisi yang menentukan akhir loop, dan penentu perubahan nilai.

Bentuk Umum:

For (<init-exp> ; <test-exp> ; <inc/dec-exp>)

Contoh :

Tulislah dalam Class

```
1  </pre>
2  namespace Looping_for
3  {
4      public class rata
5      {
6          public void hitung()
7          {
8              byte total = 0;
9              byte angka, totalAngka;
10             float rata=0;
11
12             Console.Write("Masukkan banyak angka: ");
13             angka= Convert.ToByte(Console.ReadLine());
14
15             for (byte i = 0; i < angka; i++)
16             {
17                 Console.Write("angka ke-{0} : ", i+1);
18                 totalAngka= Convert.ToByte(Console.ReadLine());
19
20                 total = Convert.ToByte(total + totalAngka);
21             }
22             rata = Convert.ToSingle(total) / angka;
23
24             Console.WriteLine("Total : " + total);
25             Console.WriteLine("Rata-rata : " + rata);
26
27             Console.ReadLine();
28         }
29     }
30 }
31 }
32 }
33 <pre>
```

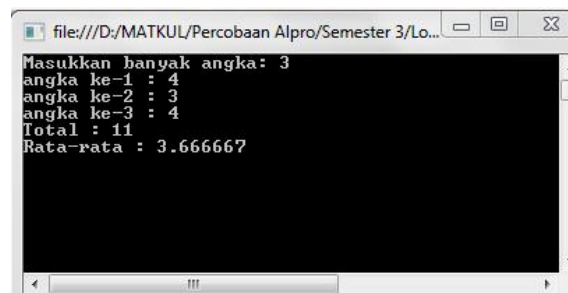
Gambar 5.1 Contoh for pada C# bagian 1

Tulislah dalam Main Program

```
1  namespace Looping_for
2  {
3      class Program
4      {
5          static void Main(string[] args)
6          {
7              rata nilai = new rata();
8              nilai.hitung();
9
10             Console.ReadLine();
11         }
12     }
13 }
14 }
```

Gambar 5.2 Contoh for pada C# bagian 2

Output



Gambar 5.3 Output Contoh for

5.3. WHILE

Pernyataan while adalah pernyataan yang berguna untuk memproses suatu pernyataan atau memproses pernyataan beberapa kali. Pernyataan atau aksi akan di ulang jika kondisi bernilai benar dan jika salah maka keluar dari blok perulangan (loop)

Bentuk umum While :

while (kondisi)

{

Pernyataan ;

}

Contoh :

Buat dalam Class

```
1 namespace Looping_while
2 {
3     public class angka
4     {
5         public void mundur()
6         {
7             byte x = 10;
8
9             Console.WriteLine("Hasilnya adalah : ");
10
11             while (x >= 3)
12             {
13                 Console.WriteLine(" " + x);
14                 x--;
15             }
16             Console.ReadLine();
17         }
18     }
19 }
20 }
```

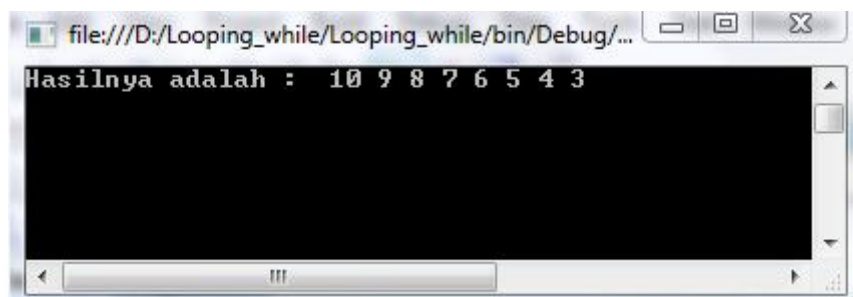
Gambar 5.4 Contoh While bagian 1

Buat pada main program

```
1 namespace Looping_while
2 {
3     class Program
4     {
5         static void Main(string[] args)
6         {
7             angka urut = new angka();
8             urut.mundur();
9
10            Console.ReadLine();
11        }
12    }
13 }
```

Gambar 5.5 Contoh While Bagian 2

Output :



Gambar 5.6 Output Contoh While

C. DO – WHILE

Perulangan akan dilakukan minimal 1x terlebih dahulu, kemudian baru dilakukan pengecekan terhadap kondisi, jika kondisi benar maka perulangan masih akan tetap dilakukan. Perulangan dengan `do...while()` akan dilakukan sampai kondisi false.

Bentuk umum :

do {

pernyataan

}

while (kondisi);

Contoh :

Tulis pada Class

```
1 </pre>
2 namespace Looping_do_while
3 {
4     public class angka
5     {
6         public void maju()
7         {
8             byte x = 1;
9
10            Console.WriteLine("Hasilnya adalah : ");
11
12            do
13            {
14                Console.Write(" " + x);
15                x++;
16            }
17            while (x <= 10);
18            Console.ReadLine();
19        }
20    }
21 }
22 }
23 <pre>
```

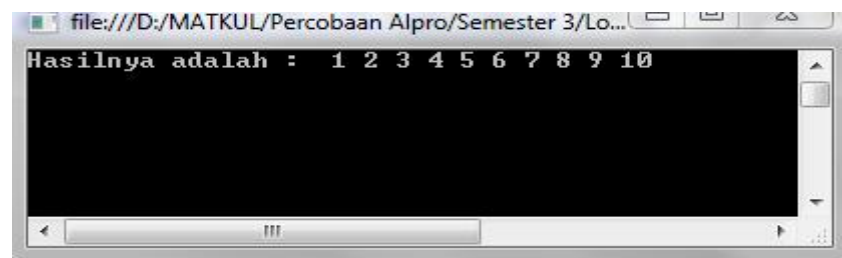
Gambar 5.7 Contoh Do-While bagian 1

Tulis pada Main Program

```
1 </pre>
2 namespace Looping_do_while
3 {
4     class Program
5     {
6         static void Main(string[] args)
7         {
8             angka urut = new angka();
9             urut.maju();
10
11            Console.ReadLine();
12        }
13    }
14 }
15 <pre>
```

Gambar 5.8 Contoh Do-While bagian 2

Output :



Gambar 5.9 Output contoh Do-While

6 Continue

Obyektif :

- Mengenal Continue pada C#
 - Mengetahui Continue pada perulangan WHILE, DO-WHILE dan FOR
-

6.1. Continue

Continue merupakan salah satu statement dalam Control Flow pada C#, Fungsi dari Statement ini adalah untuk melewati proses yang terjadi saat ini ke proses selanjutnya. Umumnya Continue digunakan pada perulangan atau Loop pada program.

Berikut ini adalah implementasi dari Statement Continue pada berbagai perulangan atau Loop:

1. Continue pada Perulangan While

```
using System;
namespace IDCSharp {
    class Program {
        static void Main(string[] args) {
            int i = 0;
            while (i < 10) {
                i++;
                if (i == 4) continue;
                Console.WriteLine("Nilai dari i adalah : {0}", i);
            }
            Console.WriteLine("Keluar dari Program ?");
            Console.ReadLine();
        }
    }
}
```

Gambar 6.1 Continue pada while

2. Continue pada Perulangan Do – While

```
using System;
namespace IDCSharp
{
    class Program
    {
        static void Main(string[] args)
        {
            int i = 0;
            do
            {
                Console.WriteLine(" Nilai dari i adalah : {0}", i );
                i++;
                if (i == 4)
                    continue;
            } while(i < 10);
            Console.WriteLine( "Keluar dari Program ?" );
            Console.ReadLine();
        }
    }
}
```

Gambar 6.2 Continue pada Do-While

3. Continue pada Perulangan For

```
using System;
namespace IDCSharp
{
    class Program
    {
        static void Main(string[] args)
        {
            for (int i = 0; i <= 10; i++)
            {
                if (i == 4)
                    continue;
                Console.WriteLine(" Nilai dari i adalah : {0}", i );
            }
            Console.WriteLine( "Keluar dari Program ?" );
            Console.ReadLine();
        }
    }
}
```

Gambar 6.3 Continue pada For

Pada contoh kode diatas, program akan menampilkan teks **“Nilai dari i adalah : (angka nilai pada i)”**. Namun ketika i bernilai **4**, maka akan dilewati dan teks dengan nilai **i = 4** tidak akan ditampilkan

7 Break

Obyektif :

- Mengetahui penggunaan Break pada C#
 - Mengetahui perbedaan Break pada perulangan WHILE, DO-WHILE, FOR, dan SWITCH-CASE
-

7.1. Break

Break adalah salah satu statement dalam Control Flow, Fungsi dari break ini adalah untuk menghentikan proses perulangan (Loop) pada program. Pada Bahasa C#, break biasa digunakan pada 2 situasi:

- Mengakhiri atau menghentikan Looping.
- Digunakan untuk mengakhiri pernyataan case pada Switch Case.

Berikut ini adalah implementasi Statement Break C# pada berbagai perulangan dan juga percabangan:

1. Break pada Perulangan For

```
using System;
namespace Idossharp
{
    class Program
    {
        static void Main(string[] args)
        {
            for (int i = 0; i <= 10; i++)
            {
                if (i == 5)
                {
                    break;
                }
                Console.WriteLine("Nilai i adalah: {0}", i);
            }
            Console.WriteLine("Keluar dari Program ?");
            Console.ReadLine();
        }
    }
}
```

Gambar 7.1 Break pada for

Pada kode program di atas, break berfungsi untuk menghentikan program ketika nilai dari variable i mencapai angka 5.

2. Break pada Perulangan Do – While

```
using System;

namespace Idcsharp
{
    class Program
    {
        static void Main(string[] args)
        {
            int i = 0;
            do
            {
                Console.WriteLine("Nilai i adalah: {0}", i );
                i++;
                if (i == 5)
                    break;
            }while (i < 10)
            Console.WriteLine("Keluar dari Program?");
            Console.ReadLine();
        }
    }
}
```

Gambar 7.2 Break pada Do-While

Pada kode program di atas, break berfungsi untuk menghentikan program ketika nilai dari variable i mencapai angka 5.

3. Break pada Perulangan While

```
using System;
namespace Idcsharp
{
    class Program
    {
        static void Main(string[] args)
        {
            int i = 0;
            while (i < 5)
            {
                Console.WriteLine(" Nilai i adalah: {0}", i );
                i++;
                if (i == 3)
                    break;
            }
            Console.WriteLine("Keluar dari Program?");
            Console.ReadLine();
        }
    }
}
```

Gambar 7.3 Break pada While

Pada kode program di atas, break berfungsi untuk menghentikan program ketika nilai dari variable i mencapai angka 3.

4. Break pada Percabangan Switch – Case

```
using system;
public class Program
{
    public static void Main()
    {
        int age = 42;

        switch (age) {
            case 16:
                Console.WriteLine("Muda");
                break;

            case 42:
                Console.WriteLine("Remaja");
                break;

            case 70:
                Console.WriteLine("Dewasa");
                break;
        }
    }
}
```

Gambar7.4 Break pada Switch

Break pada Switch Case bertujuan untuk menghindari eksekusi pada case dibawahnya dan menghentikan program ketika value pada case sama dengan yang diharapkan.

8 Komentar

Obyektif :

- Mengetahui perbedaan single-line comment dengan multi-line comment

8.1. Komentar

Komentar adalah pernyataan yang tidak dijalankan oleh kompilator. Komentar dalam pemrograman C# dapat digunakan untuk memberikan penjelasan tentang kode, variabel, metode atau kelas. Dengan bantuan komentar, Anda dapat menyembunyikan kode program juga.

Ada dua jenis komentar di C#.

- Single Line Comment (Komentar Satu Baris), Komentar satu baris dimulai dengan //(garis miring ganda).
- Multi Line Comment (Komentar multi Baris), Komentar multi baris pada C# diawali /* (garis miring bintang) dan diakhiri dengan */ (bintang, garis miring)

Contoh :

```
using System;
public class CommentExample
{
    public static void Main(string[] args)
    {
        /*
        Ini adalah sebuah multi line komentar
        */
        int x = 10; //ini adalah variabel x bernilai 10
    }
}
```

Hasil:

10

9 Function

Obyektif :

- Mengenal function pada C#
 - Mengetahui parameter ref dan out pada C#
-

9.1. Function

Function adalah blok kode yang dieksekusi ketika dipanggil. Function digunakan untuk mengeksekusi pernyataan yang ditentukan dalam blok kode. Tujuannya untuk menggunakan blok kode berkali-kali. Sebuah fungsi terdiri dari beberapa komponen berikut:

- **Function name** : Nama unik yang digunakan untuk membuat fungsi.
- **Return type**: Digunakan untuk menentukan tipe data dari sebuah fungsi.
- **Body**: Blok yang berisi pernyataan executable.
- **Access specifier**: Digunakan untuk menentukan aksesibilitas fungsi dalam aplikasi.
- **Parameters**: Daftar argumen bahwa kita dapat melewati fungsi selama panggilan.

Sintaks:

```
<access-specifier><return-type>FunctionName(<parameters>)  
{  
    // function body  
    // return statement  
}
```

Contoh :

1. Tidak Menggunakan Parameter

```
using System;
namespace FunctionExample
{
    class Program
    {
        // User defined function without return type
        public void Show() // No Parameter
        {
            Console.WriteLine("This is non parameterized function");
            // No return statement
        }
        // Main function, execution entry point of the program
        static void Main(string[] args)
        {
            Program program = new Program(); // Creating Object
            program.Show(); // Calling Function
        }
    }
}
```

Hasil:

This is non parameterized function

1. Menggunakan Parameter

Berikut Contoh function menggunakan parameter:

```
using System;
namespace FunctionExample
{
    class Program
    {
        // User defined function without return type
        public void Show(string message)
        {
            Console.WriteLine("Hello " + message);
            // No return statement
        }
        // Main function, execution entry point of the program
        static void Main(string[] args)
        {
            Program program = new Program(); // Creating Object
            program.Show("Rahul Kumar"); // Calling Function
        }
    }
}
```

Hasil:

Hello Rahul Kumar

9.2. Call By Value

Dalam C#, nilai parameter pada fungsi bila menggunakan value tidak akan mengubah nilai asli

Contoh :

```
using System;
namespace CallByValue
{
    class Program
    {
        // User defined function
        public void Show(int val)
        {
            val *= val; // Manipulating value
            Console.WriteLine("Value inside the show function "+val);
            // No return statement
        }
        // Main function, execution entry point of the program
        static void Main(string[] args)
        {
            int val = 50;
            Program program = new Program(); // Creating Object
            Console.WriteLine("Value before calling the function "+val);
            program.Show(val); // Calling Function by passing value
            Console.WriteLine("Value after calling the function " +
val);
        }
    }
}
```

Hasil:

```
Value before calling the function 50
Value inside the show function 2500
Value after calling the function 50
```

9.3. Call By Reference

Pada C# untuk memanggil menggunakan reference menggunakan kata kunci `ref` . nilai parameter pada fungsi bila menggunakan `ref` bersifat permanen dan mengubah nilai variabel asli.

Contoh :

```
using System;
namespace CallByReference
{
    class Program
    {
        // User defined function
        public void Show(ref int val)
        {
            val *= val; // Manipulating value
            Console.WriteLine("Value inside the show
function "+val);
            // No return statement
        }
        // Main function, execution entry point of the
program
        static void Main(string[] args)
        {
            int val = 50;
            Program program = new Program(); // Creating
Object
            Console.WriteLine("Value before calling the
function "+val);
            program.Show(ref val); // Calling Function by
passing reference
            Console.WriteLine("Value after calling the
function " + val);
        }
    }
}
```

Hasil:

```
Value before calling the function 50
Value inside the show function 2500
Value after calling the function 2500
```

9.4. Out Parameter

Out parameter mirip dengan reference parameter. perbedaannya pada ref kita harus menginisialisasi nilai variable sebelum di lempar ke function. Umumnya digunakan ketika membuat method dengan return yang lebih dari satu.

Contoh :

```
using System;
namespace OutParameter
{
    class Program
    {
        // User defined function
        public void Show(out int val) // Out parameter
        {
            int square = 5;
            val = square;
            val *= val; // Manipulating value
        }
        // Main function, execution entry point of the
        program
        static void Main(string[] args)
        {
            int val = 50;
            Program program = new Program(); // Creating
            Object
            Console.WriteLine("Value before passing out
            variable " + val);
            program.Show(out val); // Passing out argument
            Console.WriteLine("Value after recieving the
            out variable " + val);
        }
    }
}
```

Hasil:

```
Value before passing out variable 50
Value after receiving the out variable 25
```


10 Array

Obyektif :

- Mengetahui penggunaan array pada C#
 - Mengetahui perbedaan Array Satu dimensi, multi-dimensi, dan jagged array
-

10.1. Array

Seperti bahasa pemrograman lain, array dalam C# adalah sekelompok nilai yang memiliki lokasi memori yang berdekatan. Dalam C#, array adalah *objek System.array*. Dalam C#, indeks array dimulai dari 0.

Ada 3 jenis array dalam pemrograman C#:

1. Array satu dimensi
2. Multidimensional array
3. Jagged Array

10.2. Array Satu Dimensi

Untuk membuat array dimensi tunggal, Anda perlu menggunakan kurung siku [] setelah tipe datanya.

```
int[] arr = new int[5]; //creating array
```

Contoh :

```
using System;
public class ArrayExample
{
    public static void Main(string[] args)
    {
        int[] arr = new int[5]; //creating
        array
        arr[0] = 10; //initializing array
        arr[2] = 20;
        arr[4] = 30;

        //traversing array
        for (int i = 0; i < arr.Length; i++)
        {
            Console.WriteLine(arr[i]);
        }
    }
}
```

Hasil:

```
10
0
20
0
30
```

10.3. Array Multidimensi

Array Multidimensional juga dikenal sebagai array persegi panjang di C#. Hal ini dapat dua dimensi atau tiga dimensi. Data yang disimpan dalam bentuk tabel (baris * kolom) yang juga dikenal sebagai matriks.

```
int[,] arr=new int[3,3]; //declaration of 2D array
int[,,] arr=new int[3,3,3]; //declaration of 3D array
```

Contoh :

```
using System;
public class MultiArrayExample
{
    public static void Main(string[] args)
    {
        int[,] arr=new int[3,3];//declaration of 2D
        array
        arr[0,1]=10;//initialization
        arr[1,2]=20;
        arr[2,0]=30;

        //traversal
        for(int i=0;i<3;i++){
            for(int j=0;j<3;j++){
                Console.Write(arr[i,j]+" ");
            }
            Console.WriteLine();//new line at each row
        }
    }
}
```

Hasil:

```
0 10 0
0 0 20
30 0 0
```

10.4. Jagged Array

Jagged Array pada C# merupakan array yang berada didalam array yang dimana anggota arraynya dapat menggunakan panjang array yang berbeda.jagged array dapat digabungkan dengan array multidimensi.

```
// Deklarasi jagged array
jagged_arr[0] = new int[2];
jagged_arr[0] = new int[] {1, 2, 3, 4};
```

Contoh :

```
using System;
public class ArrayJagged
{
    public static void Main(string[] args)
    {
        Int[][] jagged_arr = new int[4][];
        Jagged_arr [0] = new int[]{1,2,3,4};
        Jagged_arr [1] = new int[]{11,34,67};
        Jagged_arr [2] = new int[]{89,23};
        Jagged_arr [3] = new int[]{0, 45, 78, 53, 99};

        for (int n = 0; n < arr.Length; n++)
        {
            System.Console.Write("Row({0}): ", n);
            For (int k = 0; k< jagged_arr[n].length;k++){
                System.Console.Write("{0} ", jagged_arr[n][k]);
            }

            System.Console.WriteLine();
        }
    }
}
```

Hasil:

```
Row(0): 1 2 3 4
Row(1): 11 34 67
Row(2): 89 23
Row(3): 0 45 78 53 99
```

10 OOP(Object Oriented Programming)

Obyektif :

- **Mengenal Object dan Class pada C#**
- **Mengetahui perbedaan Object dan Class**
- **Mengenal Constructor dan Destructor**
- **Mengetahui perbedaan Default Constructor dan Parameterized Constructor**
- **Mengetahui perbedaan Constructor dan Destructor**
- **Mengenal keyword This, Static, Static Class, dan Static Field**
- **Mengenal Inheritance, Polymorphism, dan Encapsulation**
- **Mengetahui perbedaan Single Level Inheritance dan Multi level Inheritance**
- **Mengetahui perbedaan Member Overloading dan Method Overriding**
- **Mengenal Abstaction**
- **Mengenal File I/O**

11.1. Object

Dalam C#, objek adalah entitas dunia nyata, misalnya kursi, mobil, pena, ponsel, laptop, dll. Dengan kata lain, objek adalah entitas yang memiliki status dan perilaku. Di sini, status berarti data dan perilaku berarti fungsionalitas. Objek adalah entitas runtime, objek dibuat saat runtime. Objek adalah turunan dari kelas. Semua anggota kelas dapat diakses melalui objek.

Contoh sintax object.

```
Student s1 = new Student(); //creating an object of Student
```

11.2. Class

Di C#, kelas adalah sekelompok objek yang mirip. Ini adalah tempat dari mana objek dibuat. Itu dapat memiliki bidang, metode, konstruktor, dll.

Contoh syntax class.

```
public class Student
{
    int id; //field or data member
    string name; // field or data member
}
```

Contoh Object dan Class

Mari kita lihat contoh dari kelas yang memiliki dua bidang; id dan name. Ini menciptakan instance dari sebuah kelas, menginisialisasi objek dan mencetak nilai objek.

```
using System;
public class Student
{
    int id; //data member (also instance variable)
    string name; //data member (also instance variable)

    public static void Main(string[] args)
    {
        Student s1 = new Student(); //creating an object of Student
        s1.id = 101;
        s1.name = "Sonoo Jaiswal";
        Console.WriteLine(s1.id);
        Console.WriteLine(s1.name);
    }
}
```

Hasil:

```
101
Sonoo Jaiswal
```

11.3. Constructor

Dalam C#, konstruktor adalah metode khusus yang dipanggil secara otomatis pada saat pembuatan objek. Ini digunakan untuk menginisialisasi data anggota objek baru secara umum. Konstruktor di C# memiliki nama yang sama dengan kelas atau struct.

Ada 2 jenis konstruktor di C#, yaitu:

- **Default Constructor**

Konstruktor yang tidak memiliki argument dikenal sebagai konstruktor default. Itu dikenal sebagai konstruktor default. Itu dipanggil pada saat membuat objek.

Default Constructor memiliki contoh, yaitu Having Main() di dalam kelas.

```
using System;
public class Employee
{
    public Employee()
    {
        Console.WriteLine("Default Constructor Invoked");
    }

    public static void Main(string[] args)
    {
        Employee e1 = new Employee();
        Employee e2 = new Employee();
    }
}
```

Hasil:

```
Default Constructor Invoked
Default Constructor Invoked
```

- **Parameterized Constructor**

Konstruktor yang memiliki parameter disebut konstruktor parameterized. Ini digunakan untuk memberikan nilai yang berbeda untuk objek yang berbeda.

```

using System;
public class Employee
{
    public int id;
    public String name;
    public float salary;
    public Employee(int I, String n, float s)
    {
        id = I;
        name = n;
        salary = s;
    }

    public void display()
    {
        Console.WriteLine(id+" "+name+" "+salary);
    }
}

class TestEmployee
{
    public static void Main(string[] args)
    {
        Employee e1 = new Employee(101,"Sonoo",890000f);
        Employee e2 = new Employee(102,"Mahesh",490000f);
        e1.display();
        e2.display();
    }
}

```

Hasil:

```

101 Sonoo 890000
102 Mahesh 490000

```

11.4. Destructor

Sebuah destruktur bekerja berlawanan dengan konstruktor, ia merusak objek kelas. Ini dapat didefinisikan hanya sekali dalam satu kelas. Seperti konstruktor, ia dipanggil secara otomatis.

Contoh constructor dan destructor

Mari kita lihat contoh dari konstruktor dan destruktur dari C# yang dipanggil secara otomatis

```
using System;
public class Employee
{
    public Employee()
    {
        Console.WriteLine("Constructor Invoked");
    }
    ~Employee()
    {
        Console.WriteLine("Destructor Invoked");
    }
}
class TestEmployee
{
    public static void Main(string[] args)
    {
        Employee e1 = new Employee();
        Employee e2 = new Employee();
    }
}
```

Hasil:

```
Constructor Invoked
Constructor Invoked
Destructor Invoked
Destructor Invoked
```

11.5. This

Dalam pemrograman C#, this adalah kata kunci yang merujuk pada instance kelas saat ini. Ada 3 penggunaan utama kata kunci this di C#.

1. This dapat digunakan untuk merujuk variabel instance kelas saat ini. This digunakan jika nama bidang (variabel instan) dan nama parameter sama, itu sebabnya keduanya dapat dibedakan dengan mudah.

2. Dapat pula digunakan untuk melewati objek saat ini sebagai parameter ke metode lain.
3. This dapat digunakan untuk mendeklarasikan pengindeks.

```
using System;
public class Employee
{
    public int id;
    public String name;
    public float salary;
    public Employee(int id, String name, float salary)
    {
        this.id = id;
        this.name = name;
        this.salary = salary;
    }

    public void display()
    {
        Console.WriteLine(id+""+name+""+salary);
    }
}

class TestEmployee
{
    public static void Main(string[] args)
    {
        Employee e1 = new Employee(101,"Sonoo",890000f);
        Employee e2 = new Employee(102,"Mahesh",490000f);
        e1.display();
        e2.display();
    }
}
```

Hasil:

```
101 Sonoo 890000
102 Mahesh 490000
```

11.6. Static

Dalam C#, static adalah kata kunci atau pengubah yang termasuk dalam jenis bukan turunan. Jadi, instance tidak diperlukan untuk mengakses anggota static. Dalam C#, static dapat berupa field, method, constructor, class, property, operator, dan event.

Keuntungan Kata Kunci Static

Hemat memori: Sekarang kita tidak perlu membuat instance untuk mengakses anggota statis, sehingga menghemat memori. Selain itu, itu termasuk tipe, sehingga tidak akan mendapatkan memori setiap kali ketika instance dibuat.

Static Field

Field yang dinyatakan sebagai static, disebut Static Field. Tidak seperti bidang contoh yang mendapat memori setiap kali kita membuat object, hanya ada satu Salinan static field yang dibuat dalam memori. Ini dibagikan ke semua object. Ini digunakan untuk merujuk property umum dari semua object seperti rateOfInterest dalam hal Akun, nama perusahaan dalam kasus karyawan dll.

```
using System;
public class Account
{
    public int accno;
    public String name;
    public static float rateOfInterest=8.8f;
    public Account(int accno, String name)
    {
        this.accno = accno;
        this.name = name;
    }

    public void display()
    {
        Console.WriteLine(id+""+name+""+rateOfInterest);
    }
}

class TestAccount
{
    public static void Main(string[]args)
    {
        Account a1 = new Employee(101,"Sonoo");
        Account a2 = new Employee(102,"Mahesh");
        a1.display();
        a2.display();
    }
}
```

Hasil:

101 Sonoo 8.8
102 Mahesh 8.8

Static Class

Static Class C# seperti class normal tetapi tidak dapat dipakai. Itu hanya dapat memiliki anggota static. Keuntungan dari static class adalah ia menyediakan jaminan bahwa instance dari static class tidak dapat dibuat.

Poin yang perlu diingat untuk static class C#.

1. C# static class hanya berisi anggota statis
2. C# static class tidak dapat dipakai
3. C# static class disegel
4. C# static class tidak dapat berisi contoh konstruktor

Mari kita lihat contoh dari kelas statis yang berisi bidang statis dan metode statis

```
using System;
public class MyMath
{
    public static float PI=3.14f;
    public static int cube(int n) {return n*n*n;}
}

class TestMyMath
{
    public static void Main(string[]args)
    {
        Console.WriteLine("Value of PI is: "+MyMath.PI);
        Console.WriteLine("Cube of 3 is: "+MyMath.cube(3));
    }
}
```

Hasil:

Value of PI is: 3.14
Cube of 3 is: 27

11.7. Inheritance

Pada C#, inheritance merupakan proses yang dimana satu object menerima semua property dan perilaku / behaviors dari objek parent nya secara otomatis. Sehingga kamu bias menggunakan nya kembali, memperluas atau memodifikasi atribut dan behaviour yang didefinisikan dari kelas lain.

Pada C# kelas yang mewarisi anggota dari kelas lain merupakan kelas turunan / derived class dan kelas yang anggotanya diwariskan dinamakan kelas dasar atau base class. Kelas turunan merupakan kelas khusus dari base class.

```
using System;

public class Employee
{
    public float salary = 40000;
}

public class Programmer: Employee
{
    public float bonus = 10000;
}

class TestInheritance{
    public static void Main(string[] args)
    {
        Programmer p1 = new Programmer();
        Console.WriteLine("Salary: " + p1.salary);
        Console.WriteLine("Bonus: " + p1.bonus);
    }
}
```

Hasil:

```
Salary: 40000
Bonus: 10000
```

- **Single Level Inheritance**

Mari kita lihat contoh lain dari pewarisan di C # yang hanya mewarisi methods.

```
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();
    }
}
```

Hasil:

```
Eating...
Barking...
```

- **Multi Level Inheritance**

Ketika satu class mewarisi class lain yang kemudian diwarisi lagi oleh class lain, ini merupakan multi level inheritance pada C#. Inheritance bersifat transitif sehingga class turunan terakhir mendapatkan semua anggota dari kelas induknya.

```

using System;

public class Animal
{
    public void eat() { Console.WriteLine("Eating..."); }
}

public class Dog: Animal
{
    public void bark() { Console.WriteLine("Barking..."); }
}

public class BabyDog : Dog
{
    public void weep() { Console.WriteLine("Weeping..."); }
}

class TestInheritance2{
    public static void Main(string[] args)
    {
        BabyDog d1 = new BabyDog();
        d1.eat();
        d1.bark();
        d1.weep();
    }
}

```

Hasil:

```

Eating...
Barking...
Weeping...

```

11.8. Polymorphism

Istilah "Polimorfisme" merupakan gabungan dari "poli" + "morf" yang artinya banyak bentuk. Ada dua jenis polimorfisme dalam C#: compile time polymorphism dan runtime polymorphism. Compile time polymorphism dicapai dengan metode overloading dan operator overloading dalam C#. Ia dikenal sebagai pengikatan statis atau pengikatan awal. Polymorphism runtime dicapai dengan metode overriding yang disebut juga dynamic binding atau late binding.

- **Member Overloading**

Jika kita membuat 2 atau lebih anggota yang memiliki nama yang sama tetapi memiliki perbedaan nomor atau tipe dari parameter, ini biasa disebut dengan overloading. Pada C# kita bias overload:

- Method
- Constructor
- Index properties

- **Method Overloading**

Memiliki 2 atau lebih method dengan nama yang sama tetapi dengan parameter yang berbeda ini disebut dengan method overloading. Keuntungan dari method overloading adalah dapat meningkatkan readability dari program karena kamu tidak membutuhkan untuk menggunakan nama yang berbeda untuk aksi yang sama.

Kamu bisa perform method overloading pada C# dengan 2 cara:

- Mengganti nomor pada argument
- Mengganti tipe data pada argument

Contoh method overloading dengan mengganti nomor argument pada method add()

```
using System;
public class Cal{
    public static int add(int a,int b){
        return a + b;
    }
    public static int add(int a, int b, int c)
    {
        return a + b + c;
    }
}
public class TestMemberOverloading
{
    public static void Main()
    {
        Console.WriteLine(Cal.add(12, 23));
        Console.WriteLine(Cal.add(12, 23, 25));
    }
}
```


Hasil:

```
35  
60
```

- **Method Overriding**

Jika kelas turunan di definisikan sama dengan metode yang sama seperti pada kelas basisnya, ini dikenal dengan method overriding pada C#. Ini digunakan untuk mencapai polimorfisme. Ini memungkinkan untuk mengimplementasikan secara spesifik dari method yang sudah disediakan pada base class nya.

```
using System;  
public class Animal{  
    public virtual void eat(){  
        Console.WriteLine("Eating...");  
    }  
}  
public class Dog: Animal  
{  
    public override void eat()  
    {  
        Console.WriteLine("Eating bread...");  
    }  
}  
public class TestOverriding  
{  
    public static void Main()  
    {  
        Dog d = new Dog();  
        d.eat();  
    }  
}
```

Hasil:

```
Eating bread...
```

11.9. Abstract

Kelas abstrak adalah cara untuk mencapai abstraksi dalam C#. Abstraksi dalam C# adalah proses untuk menyembunyikan detail internal dan hanya menunjukkan fungsionalitas. Abstraksi dapat dicapai dengan dua cara:

1. Kelas abstrak
2. Antarmuka

- **Abstract Method**

Metode yang dinyatakan abstrak dan tidak memiliki tubuh disebut metode abstrak. Itu dapat dideklarasikan di dalam kelas abstrak saja. Implementasinya harus disediakan oleh kelas turunan. Sebagai contoh:

```
public abstract void draw();
```

- **Abstract Class**

Dalam C#, kelas abstrak adalah kelas yang dinyatakan abstrak. Itu dapat memiliki metode abstrak dan non-abstrak. Itu tidak bisa dipakai. Implementasinya harus disediakan oleh kelas turunan. Di sini, kelas turunan dipaksa untuk menyediakan implementasi semua metode abstrak.

Mari kita lihat contoh kelas abstrak di C# yang memiliki satu metode menggambar abstrak (). Implementasinya disediakan oleh kelas turunan: Rectangle dan Circle. Kedua kelas memiliki implementasi yang berbeda.

```
using System;
public abstract class Shape
{
    public abstract void draw();
}
public class Rectangle : Shape
{
    public override void draw()
    {
        Console.WriteLine("drawing rectangle...");
    }
}
```

```

public class Circle : Shape
{
    public override void draw()
    {
        Console.WriteLine("drawing circle...");
    }
}
public class TestAbstract
{
    public static void Main()
    {
        Shape s;
        s = new Rectangle();
        s.draw();
        s = new Circle();
        s.draw();
    }
}

```

Hasil:

```

drawing ractangle...
drawing circle...

```

11.10.Encapsulation

Enkapsulasi adalah konsep membungkus data ke dalam satu unit. Ini mengumpulkan anggota data dan fungsi anggota ke dalam satu unit yang disebut kelas. Tujuan enkapsulasi adalah untuk mencegah perubahan data dari luar. Data ini hanya dapat diakses oleh fungsi pengambil kelas. Kelas yang sepenuhnya terenkapsulasi memiliki fungsi pengambil dan penyetel yang digunakan untuk membaca dan menulis data. Kelas ini tidak mengizinkan akses data secara langsung. Di sini, kami membuat contoh di mana kami memiliki kelas yang merangkum properti dan menyediakan fungsi pengambil dan penyetel.

```

namespace AccessSpecifiers
{
    class Student
    {
        // Creating setter and getter for each property
        public string ID { get; set; }
        public string Name { get; set; }
        public string Email { get; set; }
    }
}

```

Contoh :

```

using System;
namespace AccessSpecifiers
{
    class Program
    {
        static void Main(string[] args)
        {
            Student student = new Student();
            // Setting values to the properties
            student.ID = "101";
            student.Name = "Mohan Ram";
            student.Email = "mohan@example.com";
            // getting values
            Console.WriteLine("ID = "+student.ID);
            Console.WriteLine("Name = "+student.Name);
            Console.WriteLine("Email = "+student.Email);
        }
    }
}

```

Hasil:

```
ID = 101  
Name = Mohan Ram  
Email = mohan@example.com
```

11.11.File I/O

FileStream

kelas FileStream menyediakan aliran untuk operasi file. Ini dapat digunakan untuk melakukan operasi baca dan tulis yang sinkron dan asinkron. Dengan bantuan kelas FileStream, kita dapat dengan mudah membaca dan menulis data ke dalam file.

```
using System;  
using System.IO;  
public class FileStreamExample  
{  
    public static void Main(string[] args)  
    {  
        FileStream f = new FileStream("e:\\b.txt",  
        FileMode.OpenOrCreate); //creating file stream  
        f.WriteByte(65); //writing byte into stream  
        f.Close(); //closing stream  
    }  
}
```

Hasil:

```
A
```

Contoh C # FileStream contoh: menulis banyak byte ke dalam file

```
using System;
using System.IO;
public class FileStreamExample
{
    public static void Main(string[] args)
    {
        FileStream f = new FileStream("e:\\b.txt",
        FileMode.OpenOrCreate);
        for (int i = 65; i <= 90; i++)
        {
            f.WriteByte((byte)i);
        }
        f.Close();
    }
}
```

Hasil:

ABCDEFGHIJKLMNOPQRSTUVWXYZ

Contoh C # FileStream contoh: membaca semua byte dari file

```
using System;
using System.IO;
public class FileStreamExample
{
    public static void Main(string[] args)
    {
        FileStream f = new FileStream("e:\\b.txt",
        FileMode.OpenOrCreate);
        int i = 0;
        while ((i = f.ReadByte()) != -1)
        {
            Console.Write((char)i);
        }
        f.Close();
    }
}
```

Hasil:

ABCDEFGHIJKLMNOPQRSTUVWXYZ

FileInfo Contoh: Membuat File

```
using System;
using System.IO;
namespace CSharpProgram
{
    class Program
    {
        static void Main(string[] args)
        {
            try
            {
                // Specifying file location
                string loc = "F:\\abc.txt";
                // Creating FileInfo instance
                FileInfo file = new FileInfo(loc);
                // Creating an empty file
                file.Create();
                Console.WriteLine("File is created Successfully");
            } catch (IOException e)
            {
                Console.WriteLine("Something went wrong: "+e);
            }
        }
    }
}
```

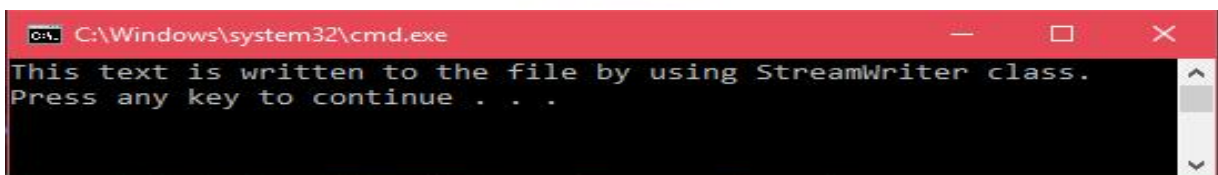
Hasil:

File is created Successfully

FileInfo Contoh: menulis ke file

```
using System;
using System.IO;
namespace CSharpProgram
{
    class Program
    {
        static void Main(string[] args)
        {
            try
            {
                // Specifying file location
                string loc = "F:\\\\abc.txt";
                // Creating FileInfo instance
                FileInfo file = new FileInfo(loc);
                // Creating an file instance to write
                StreamWriter sw = file.CreateText();
                // Writing to the file
                sw.WriteLine("This text is written to the file by using
StreamWriter class.");
                sw.Close();
            } catch (IOException e)
            {
                Console.WriteLine("Something went wrong: "+e);
            }
        }
    }
}
```

Hasil:



```
C:\Windows\system32\cmd.exe
This text is written to the file by using StreamWriter class.
Press any key to continue . . .
```

Gambar 11.1 Output Program CSharpProgram