Back End Development 1
sesi 4

C# Data

C# Data - Sesi 04

Konsep Memori dalam Pemogramman

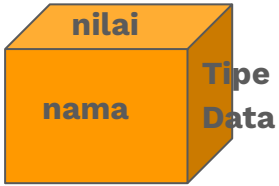
Pada C# Data ini kita akan belajar bagaimana dan jenis data apa saja yang bisa di handle oleh C#,

Nah sebelum lebih jauh ada sub materi yang cukup bagus kita bahas terkait alokasi memory pada C#,

Ingat pada pembahasan Sesi 2 pada pembahasan tipe data ada yang namanya :
Tipe Nilai dan Referensi, dan dimana perbedaannya ?

Semua aplikasi menyimpan dan memanipulasi data di dalam memori komputer.
Ketika kita mendeklarasikan sebuah variabel, sebuah potongan memori akan dialokasikan untuk menyimpan dan memanipulasi variabel tersebut.

Potongan memori ini menyimpan tiga buah informasi, yaitu nama variable, tipe data dan nilai dari variable tersebut



Penggalan Memori

HACKTIV8

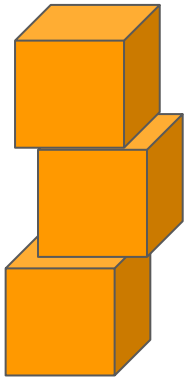
Konsep Memori dalam Pemogramman

Ada dua jenis pengalokasian memori berdasarkan dari tipe data yang disimpan, yaitu memori stack di mana data akan disimpan dalam alamat memori secara berurutan, serta memori heap di mana data akan disimpan tanpa mempertimbangkan urutan alamat memorinya.

Untuk mempermudah pemahaman Anda mengenai dua jenis pengalokasian memori tersebut,

coba Anda bayangkan memori stack seperti sebuah tumpukan kardus dan memori heap seperti sebuah loker.

Ketika menumpuk sebuah kardus, Anda biasanya memulainya dengan menumpuk dari bawah, kemudian menumpuk kardus yang lain di atasnya.

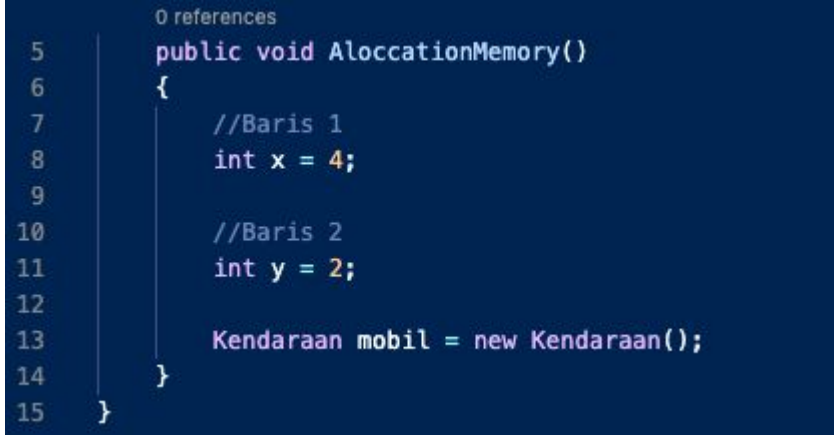


Konsep Memori dalam Pemogramman

Berbeda ketika Kita ingin menyimpan barang ke dalam sebuah loker, Kita bebas memilih pintu loker yang mana saja selama loker tersebut kosong.

Ketika ingin mengeluarkan barang, Kita juga bisa langsung menuju pintu loker tersebut tanpa harus mengeluarkan barang yang disimpan di dalam pintu loker yang lain.

Gambar oleh OpenClipart-Vectors dari Pixabay



```
0 references
5   public void AloccationMemory()
6   {
7       //Baris 1
8       int x = 4;
9
10      //Baris 2
11      int y = 2;
12
13      Kendaraan mobil = new Kendaraan();
14  }
15 }
```

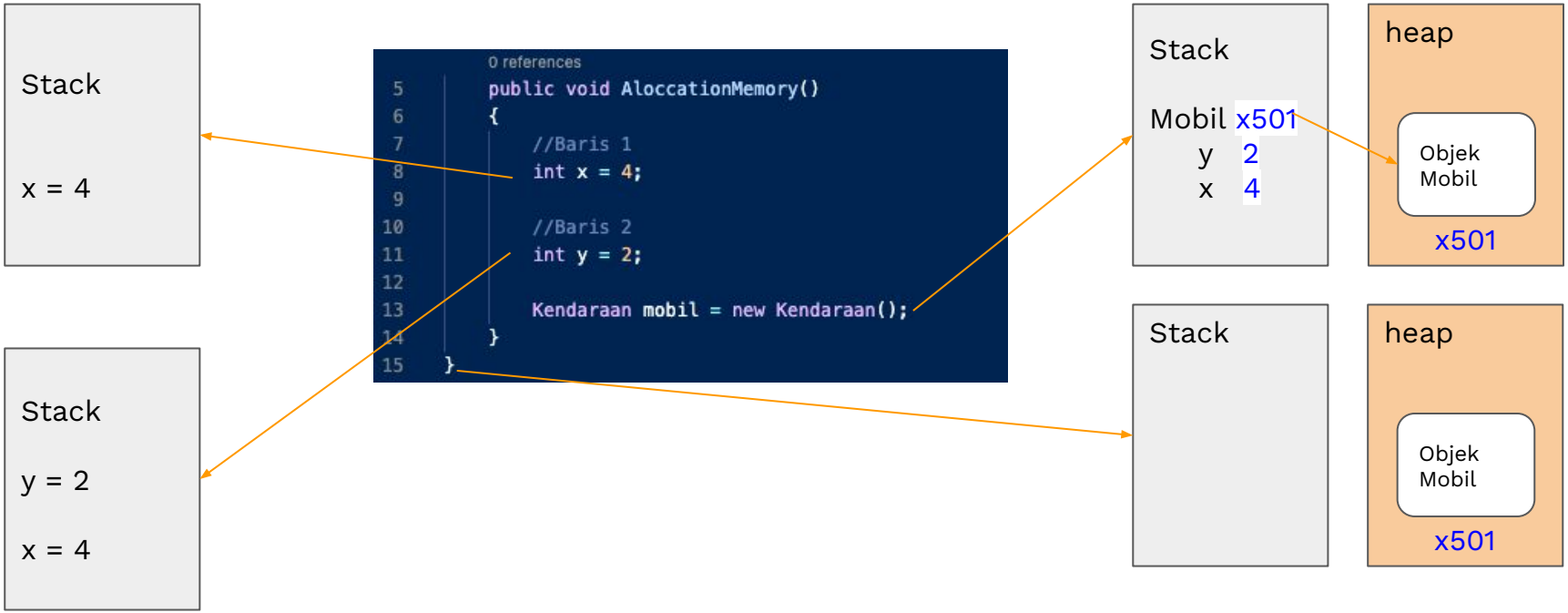
Mari kita pahami bagaimana kode program di atas dieksekusi baris demi baris.

Baris 1 : Ketika baris kode program ini dieksekusi, kompilator akan mengalokasikan 4 byte atau 32 bit memori di stack untuk variabel ‘x’. (Lihat kembali tabel tipe data, variabel int memiliki ukuruan 4 byte atau 32 bit)

Baris 2 : Baris kedua selanjutnya dieksekusi. Kompilator akan mengalokasikan 4 byte untuk variabel ‘y’ dan diletakkan di “atas” alokasi memori untuk variabel ‘x’. Itulah mengapa memori ini bernama stack atau tumpukan.

Baris 3 : Pada eksekusi baris ketiga, kita membuat sebuah instance bernama ‘mobil’ dari sebuah class bernama ‘Kendaraan’. Ketika baris ini dieksekusi, sebuah pointer (referensi) akan dibuat di memori stack dan obyek yang sebenarnya akan disimpan di tipe memori yang berbeda bernama “Heap”. Pointer yang disimpan di memori stack, berisi alamat yang menunjukkan lokasi object di memori heap.

Keluar dari method ‘AlokasiMemori’ : Ketika kontrol eksekusi keluar dari method di atas, semua variabel bertipe int yang disimpan di memori stack, termasuk referensi ke lokasi obyect ‘mobil’ akan dihapus satu per satu (didealokasi) dengan konsep LIFO (Last In First Out).



HACKTIV8

Konsep Memori dalam Pemogramman

Kapan Sebaiknya menggunakan memori heap ?

Ketika kita memerlukan kapasitas memori yang menyimpan banyak data objek dan data tersebut ingin kita simpan dalam waktu yang lama (global variabel).

C# Data - Sesi 04

Array

Pemrograman membutuhkan variabel untuk menyatakan suatu kegiatan proses tertentu. Dalam kondisi tertentu, terkadang membutuhkan kumpulan data yang sama dalam tipe.
Di dalam pemrograman dikenal dengan nama array.

Array mempunyai keterbatasan, yaitu harus memdefinisikan banyaknya data yang dibutuhkan.

Array dideklarasikan dengan tipe data diikuti dengan sepasang kurung siku. Sebagai contoh:

```
17    int[] aAngka;
18    string[] aNama;
19    object[] aObjek;
```

Deklarasi array di atas hanya menghasilkan variabel array kosong. Untuk memberi nilai awal pada array, lakukan salah satu dari cara berikut:

```
17    int[] aAngka = new int[5];
18    string[] aNama = new string[] { "Joni", "Meri", "David" };
19    object[] aObjek = { 20.33, "Lorem ipsum", DateTime.Now, true, 'D' };
```

Contoh pertama membuat 5 int kosong, sedangkan contoh kedua membuat array dengan ukuran 3 dan langsung menugaskan nilai-nilai kepada elemen-elemen array tersebut.

Contoh ketiga menunjukkan versi lain dari contoh ke dua yang mengabaikan keyword "new".

Contoh ketiga juga menunjukkan bahwa kita dapat menugaskan nilai bertipe apa saja kepada suatu objek.

Pada contoh ketiga, kita menugaskan double, string, DateTime, bool dan char sekaligus ke dalam array object tersebut.

Bagaimana cara menggunakan array ?

Assignment

Assignment adalah proses memasukkan data ke dalam *Array* dan selain itu dilakukan juga pemberian ukuran *Array* , contohnya seperti

```
13          string[] contoh = new string [4]
14          {
15              "Motherboard", "Processor", "Power Supply", "Video Card"
16          };
```

C# Data - Sesi 04

Cara Akses Array

Accessing adalah proses atau cara untuk mengakses data dalam *Array* , namun yang harus diingat adalah awalan atau elemen pertama dari *Array* bukanlah indek ke 1 melainkan index ke 0, selain itu *Accessing* juga termasuk merubah data dan menyimpannya, contohnya *Accessing* yaitu,

```
13      string[] contoh = new string [4]
14      {
15          "Motherboard", "Processor", "Power Supply", "Video Card"
16      };
17
18      string contoh1 = contoh[3];
19      Console.WriteLine(contoh1);
20      Console.Write("Press any key to continue . . . ");
```

C# Data - Sesi 04

Cara Menampilkan Array

Untuk menampilkan semua data kita dapat menggunakan bantuan perulangan foreach, jadi dalam memakai *Array* untuk implementasi atau penerapan yang sebenarnya, kita akan sering menggunakan bantuan perulangan seperti for, while dan juga foreach.

Perhatikan kode program dibawah ini,

```
0 references
public static void Main(string[] args)
{
    string[] contoh = new string [4]
    {
        "Motherboard", "Processor", "Power Supply", "Video Card"
    };

    Console.WriteLine("Menampilkan semua data dalam array");
    Console.WriteLine("");

    foreach(string contoh1 in contoh)
    {
        Console.WriteLine(contoh1);
    }

    Console.Write("Press any key to continue . . . ");
}
```

Dan Hasilnya :

```
Menampilkan semua data dalam array

Motherboard
Processor
Power Supply
Video Card
Press any key to continue . . . Maliks-MacBook-Air:Sesi4 swijay
```

Array Multidimensi

Bentuk paling sederhana dari array multidimensi adalah array dua dimensi. Dalam array dua dimensi, lokasi dari elemen tertentu dispesifikasi dengan dua indeks. Anda bisa berimajinasi bahwa array dua dimensi seperti tabel informasi, dimana satu indeks mengindikasikan baris dan indeks lainnya mengindikasikan kolom.

```
1   using System;
2
    0 references
3   class data3 {
        0 references
4       static void Main() {
5           int t, i;
6           int[,] tabel = new int[3, 4];
7
8           for (t = 0; t < 3; ++t)
9           {
10              for (i = 0; i < 4; ++i)
11              {
12                  tabel[t, i] = (t * 4) + i + 1;
13                  Console.Write(tabel[t, i] + " ");
14              }
15
16              Console.WriteLine();
17          }
18      }
19  }
```

```
1 2 3 4
5 6 7 8
9 10 11 12
```

Pada contoh ini, **tabel[0, 0]** akan memiliki nilai 1, **tabel[0, 1]** nilai 2, **tabel[0, 3]** nilai 3, dan seterusnya. Nilai dari **tabel[3, 2]** adalah 12.

	0	1	2	3
0	1	2	3	4
1	5	6	7	8
2	9	10	11	12

C# Data - Sesi 04

Inisialisasi Array MultiDimensi

Array multidimensi dapat diinisialisasi dengan mengapit daftar penginisialisasi pada tiap dimensi menggunakan kurung kurawal.

```
using System;

0 references
class data4
{
    0 references
    static void Main()
    {
        int[,] data4 = {
          { 1, 1 },
          { 2, 4 },
          { 3, 9 },
          { 4, 16 },
          { 5, 25 },
          { 6, 36 },
          { 7, 49 },
          { 8, 64 },
          { 9, 81 },
          { 10, 100 }
    };
        int i, j;

        for (i = 0; i < 10; i++)
        {
            for (j = 0; j < 2; j++)
                Console.Write(data4[i, j] + " ");
            Console.WriteLine();
        }
    }
}
```

```
Maliks-MacBook-Air:Se
1 1
2 4
3 9
4 16
5 25
6 36
7 49
8 64
9 81
10 100
```

HACKTIV8

Array Jagged

Pada beberapa contoh sebelumnya, ketika Kita menciptakan sebuah array dua dimensi, Kita menciptakan apa yang dinamakan dengan array persegi-panjang. Array persegi-panjang seperti tabel, dimana panjang setiap baris sama.

Namun, C# juga mengijinkan Anda untuk menciptakan suatu tipe spesial dari array dua dimensi, yang dikenal dengan array jagged.

Array jagged merupakan array yang memuat beberapa array, dimana di dalamnya panjang setiap array bisa berbeda satu sama lain.

Jadi, array jagged dapat dipakai untuk menciptakan tabel dimana di dalamnya panjang baris tidak sama.

Array jagged dideklarasikan menggunakan beberapa pasang kurung siku untuk mengindikasikan setiap dimensi.

```cs
using System;

0 references
class Jagged {
  0 references
  static void Main() {
    int[][] jagged = new int[3][];
    jagged[0] = new int[4];
    jagged[1] = new int[3];
    jagged[2] = new int[5];
    int i;

    // Menyimpan nilai-nilai dalam array pertama.
    for(i=0; i < 4; i++)
      jagged[0][i] = i;

    // Menyimpan nilai-nilai dalam array kedua.
    for(i=0; i < 3; i++)
      jagged[1][i] = i;

    // Menyimpan nilai-nilai dalam array ketiga.
    for (i = 0; i < 5; i++)
      jagged[2][i] = i;

    // Menampilkan nilai-nilai pada array pertama.
    for (i = 0; i < 4; i++)
      Console.Write(jagged[0][i] + " ");
    Console.WriteLine();

    // Menampilkan nilai-nilai pada array kedua.
    for (i = 0; i < 3; i++)
      Console.Write(jagged[1][i] + " ");
    Console.WriteLine();

    // Menampilkan nilai-nilai pada array ketiga.
    for (i = 0; i < 5; i++)
      Console.Write(jagged[2][i] + " ");
    Console.WriteLine();
  }
}
```

output.

```
Maliks-MacBook-Air:Sesi4 swijaya$
0 1 2 3
0 1 2
0 1 2 3 4
```

Array jagged tidak digunakan pada semua terapan, tetapi bisa efektif pada beberapa situasi. Sebagai contoh, jika membutuhkan sebuah array dua dimensi yang berukuran sangat besar, tetapi berpopulasi jarang, maka array jagged bisa menjadi solusi tepat.

Satu hal penting terakhir: Karena array jagged adalah array yang memuat array, maka tidak ada batasan bahwa array berupa array satu dimensi.

Properti Length

Sejumlah keuntungan didapatkan karena C# mengimplementasikan array sebagai objek.
Salah satu keuntungan yang didapatkan berkaitan dengan properti **Length** yang memuat jumlah elemen yang dapat ditampung suatu array.

```
using System;

0 references
class Length
{
    0 references
    static void Main()
    {
        int[] angka = new int[10];

        Console.WriteLine("Panjang array angka adalah " + angka.Length);

        // Menggunakan Length untuk menginisialisasi angka.
        for (int i = 0; i < angka.Length; i++)
            angka[i] = i * i;

        // Sekarang menggunakan Length untuk menampilkan angka.
        Console.Write("Berikut adalah array angka: ");
        for (int i = 0; i < angka.Length; i++)
            Console.Write(angka[i] + " ");

        Console.WriteLine();
    }
}
```

```
Panjang array angka adalah 10
Berikut adalah array angka: 0 1 4 9 16 25 36 49 64 81
```

Perhatikan cara dimana angka.Length digunakan oleh beberapa loop for dalam mengatur jumlah iterasi yang akan dilakukan. Karena setiap array membawa panjangnya sendiri, Anda bisa menggunakan informasi ini, daripada harus secara manual menjejak ukuran array. Ingat bahwa nilai dari Length tidak ada hubungannya dengan jumlah elemen aktual yang sedang digunakan.

Length memuat jumlah elemen yang mampu ditampung oleh sebuah array.

Total jumlah elemen juga bisa dijadikan nilai balik. Sebagai contoh:

```
1   using System;
2   
    0 references
3   class Panjang3D {
        0 references
4       static void Main() {
5           int[, ,] angka = new int[10, 5, 6];
6   
7           Console.WriteLine("Panjang array angka adalah " + angka.Length);
8       }
9   }
```

Hasil :

```
Maliks-MacBook-Air:Sesi4 swijaya$
Panjang array angka adalah 300
```

C# Data - Sesi 04

String

String merupakan salah satu tipe data terpenting dalam C# karena mendefinisikan dan mendukung string karakter. Dalam banyak bahasa pemrograman, string merupakan sebuah array karakter. Ini tidak terjadi pada C#.

Dalam C#, string adalah objek. Jadi string bertipe referensi. Meskipun string merupakan salah satu tipe built-in dalam C#, diskusi tentang string harus disajikan setelah kelas dan objek dibahas.

```
1   using System;
2   
    0 references
3   class string1 {
        0 references
4       static void Main() {
5           char[] charray = { '1', ' ', 's', 't', 'r', 'i', 'n', 'g', '.' };
6           string str1 = new string(charray);
7           string str2 = "String lain.";
8   
9           Console.WriteLine(str1);
10          Console.WriteLine(str2);
11      }
12  }
```

Metode Class String

Metode	Deskripsi
static int Compare(string strA, string strB, StringComparison tipePerbandingan)	Menghasilkan nilai balik kurang dari nol jika strA kurang dari strB, lebih dari nol jika strA lebih besar dari strB, dan nol jika kedua string sama. Bagaimana komparasi dilakukan dispesifikasi oleh tipePerbandingan.
bool Equals(string nilai, StringComparison tipePerbandingan)	Menghasilkan nilai balik true jika string pemanggil sama dengan nilai. Bagaimana komparasi dilakukan dispesifikasi oleh tipePerbandingan.
int IndexOf(char nilai)	Melakukan pencarian atas kemunculan pertama karakter yang dispesifikasi oleh nilai dalam string pemanggil. Pencarian ordinal adalah pendekatan yang dipakai. Indeks dari kecocokan pertama dijadikan nilai balik, atau -1 dijadikan nilai balik jika tidak ditemukan kecocokan.
int IndexOf(string nilai, StringComparison tipePerbandingan)	Melakukan pencarian atas kemunculan pertama substring yang dispesifikasi oleh nilai dalam string pemanggil. Bagaimana komparasi dilakukan dispesifikasi oleh tipePerban-dingan. Indeks dari kecocokan pertama dijadikan nilai balik, atau -1 dijadikan nilai balik jika tidak ditemukan kecocokan.

Metode Class String

int LastIndexOf(**char** *nilai*)	Melakukan pencarian atas kemunculan terakhir karakter yang dispesifikasi oleh *nilai* dalam string pemanggil. Pencarian ordinal adalah pendekatan yang dipakai. Indeks dari kecocokan pertama dijadikan nilai balik, atau -1 dijadikan nilai balik jika tidak ditemukan kecocokan.
int LastIndexOf(**string** *nilai*, StringComparison *tipePerbandingan*)	Melakukan pencarian atas kemunculan terakhir substring yang dispesifikasi oleh *nilai* dalam string pemanggil. Bagaimana komparasi dilakukan dispesifikasi oleh *tipePerban-dingan*. Indeks dari kecocokan pertama dijadikan nilai balik, atau -1 dijadikan nilai balik jika tidak ditemukan kecocokan.
string ToLower(CultureInfo.CurrentCulture *kultur*)	Menghasilkan versi huruf kecil dari string pemanggil. Bagaimana konversi dilakukan dispesifikasi oleh *kultur*.
string ToUpper(CultureInfo.CurrentCulture *kultur*)	Menghasilkan versi huruf besar dari string pemanggil. Bagaimana konversi dilakukan dispesifikasi oleh *kultur*.

String2 - LIVE CODING

```
using System;
using System.Globalization;

0 references
class string2 {
  0 references
  static void Main() {
    string str1 = "Untuk pemrograman .NET, C# adalah #1.";
    string str2 = "Untuk pemrograman .NET, C# adalah #1.";
    string str3 = "string C# sangat tangguh.";
    string strAtas, strBawah;
    int hasil, idx;

    Console.WriteLine("str1: " + str1);
    Console.WriteLine("Panjang str1: " + str1.Length);

    // Menciptakan versi huruf besar/kecil dari str1.
    strBawah = str1.ToLower(CultureInfo.CurrentCulture);
    strAtas = str1.ToUpper(CultureInfo.CurrentCulture);
    Console.WriteLine("Versi huruf kecil dari str1:\n " +
                      strBawah);
    Console.WriteLine("Versi huruf besar dari str1:\n " +
                      strAtas);
    Console.WriteLine();

    // Menampilkan str1, karakter demi karakter.
    Console.WriteLine("Menampilkan str1, char demi char.");
    for(int i=0; i < str1.Length; i++)
      Console.Write(str1[i]);
    Console.WriteLine("\n");

    // Membandingkan string menggunakan == dan !=. Perbandigan ordinal.
    if (str1 == str2)
      Console.WriteLine("str1 == str2");
    else
      Console.WriteLine("str1 != str2");

    if (str1 == str3)
      Console.WriteLine("str1 == str3");
    else
      Console.WriteLine("str1 != str3");

    // Perbandingan ini sensitif-kultur.
    hasil = string.Compare(str1, str3, StringComparison.CurrentCulture);
    if (hasil == 0)
      Console.WriteLine("str1 dan str3 sama");
    else if (hasil < 0)
      Console.WriteLine("str1 kurang dari str3");
    else
      Console.WriteLine("str1 lebih besar dari str3");

    Console.WriteLine();

    // Menugaskan string baru ke str2.
    str2 = "Satu Dua Tiga Satu";

    // Pencarian string.
    idx = str2.IndexOf("Satu", StringComparison.Ordinal);
    Console.WriteLine("Indeks kemunculan pertama dari Satu: " + idx);

    idx = str2.LastIndexOf("Satu", StringComparison.Ordinal);
    Console.WriteLine("Indeks kemunculan terakhir dari Satu: " + idx);
  }
}
```

HACKTIV8

Hasil :

```
str1: Untuk pemrograman .NET, C# adalah #1.
Panjang str1: 37
Versi huruf kecil dari str1:
 untuk pemrograman .net, c# adalah #1.
Versi huruf besar dari str1:
 UNTUK PEMROGRAMAN .NET, C# ADALAH #1.

Menampilkan str1, char demi char.
Untuk pemrograman .NET, C# adalah #1.

str1 == str2
str1 != str3
str1 lebih besar dari str3

Indeks kemunculan pertama dari Satu: 0
Indeks kemunculan terakhir dari Satu: 14
```

Seperti yang telah dijelaskan, karena **Compare** dideklarasikan **static**, ia dipanggil dengan nama kelasnya, bukan nama instans dari kelasnya. Anda bisa menyambungkan dua string menggunakan operator +.

Satu hal penting lainnya: Katakunci string merupakan alias bagi kelas System.String yang didefinisikan oleh class library Framework .NET.

Jadi, field data dan metode yang didefinisikan oleh string adalah dari kelas System.String.

C# Data - Sesi 04

Memuat String dalam Array

```
1   using System;
2
    0 references
3   class string3 {
        0 references
4       static void Main() {
5           string[] str = { "Ini", "adalah", "sebuah", "test." };
6           Console.WriteLine("Array asli: ");
7
8           for (int i = 0; i < str.Length; i++)
9               Console.Write(str[i] + " ");
10
11          Console.WriteLine("\n");
12
13          // Mengubah string.
14          str[1] = "merupakan";
15          str[3] = "test, juga!";
16          Console.WriteLine("Array termodifikasi: ");
17
18          for (int i = 0; i < str.Length; i++)
19              Console.Write(str[i] + " ");
20      }
21  }
```

Hasil :

```
Maliks-MacBook-Air:Sesi4 swijaya$
Array asli:
Ini adalah sebuah test.

Array termodifikasi:
Ini merupakan sebuah test, juga! N
```

String Immutable

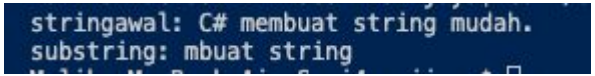
Jika Anda memerlukan sebuah string yang perlu divariasi atau diubah, Anda hanya perlu menciptakan sebuah string baru yang memuat perubahan yang diinginkan.

Tentu saja, Anda perlu memahami bahwa variabel referensi string dapat mengubah objek mana yang ditunjuk olehnya. Yang tidak bisa diubah adalah isi dari objek string.

Untuk memahami mengapa string tidak bisa diubah (immutable), berikut diberikan satu metode string: Substring(). Metode Substring() menghasilkan nilai balik berupa sebuah string baru yang memuat penggalan tertentu dari string pemanggil. String asli tidak berubah.

```
1   using System;
2   
    0 references
3   class string4 {
        0 references
4       static void Main() {
5           string stringawal = "C# membuat string mudah.";
6   
7           // Menciptakan sebuah substring.
8           string substr = stringawal.Substring(5, 12);
9   
10          Console.WriteLine("stringawal: " + stringawal);
11          Console.WriteLine("substring: " + substr);
12      }
13  }
```

Hasil :



Seperti yang dapat Anda lihat, string awal tidak berubah, dan substr memuat substring yang diinginkan.

Satu lagi hal penting: Meskipun isi objek string tidak bisa diubah, tetapi ada beberapa situasi dimana Anda perlu mengubah atau memodifikasi sebuah string.

Untuk melakukan hal ini, C# menawarkan sebuah kelas yang dikenal dengan StringBuilder, yang berada dalam namespace System.Text.

Kelas ini menciptakan objek string yang bisa dimodifikasi. Namun, pada banyak kasus, Anda hanya perlu string, bukan StringBuilder.