```
// +----------------------------+
// |  C# Important commands     |
// |  G.Raf v1.0                |
// +----------------------------+

// Datentypen
    bool x;     //1 Bit 0/1; true/false;

    byte x;     //8 Bit 0-255;
    sbyte x;    //8 Bit 127-(-128);
    ushort x;   //16 Bit 0-65535
    short x;    //16 Bit 32767-(-32768)

    uint x;   UInt16 x; //16 Bit 0-65535
              UInt32 x; //32 Bit
              UInt64 x; //64 Bit
    int x;    Int16 x;  //16 Bit 32767-(-32768)
              Int32 x;  //32 Bit
              Int64 x;  //64 Bit

    ulong x;    //64 Bit 0-18446744073709551615
    long x;     //64 Bit 922337203685477507-(-922337203685477508)

    float x;    //32  Bit 3.402823*10^38-(-3.402823*10^38)
    double x;   //64  Bit 1.79769313486232*10^308-(-1.79769313486232*10^308)
    decimal x;  //128 Bit ±7,9 × 10^28 - ±1,0 × 10^-28

    char x;     //16 Bit Unicode Text
    string x;   //unbestimmt Text

// Standard Operationen
  x += y    //x = x + y
  x -= y    //x = x - y
  x *= y    //x = x * y

// Werte umformen, kürzen
  double x = 10.25;   // x-Wert
  int y = (int)(x);   // x-Wert in INT konvertieren

// Datentyp konvertierungs Funktionen
  x = Convert.ToBoolean(y);
  x = Convert.ToByte(y);
  x = Convert.ToSByte(y);
  x = Convert.ToChar(y);
  x = Convert.ToUIntXX(y);
  x = Convert.ToIntXX(y);
  x = Convert.ToDouble(y);
  x = Convert.ToDecimal(y);
  x = Convert.ToString(y);

// ++String Funktionen
    // ++Klein/Großbuchstaben umwandeln
      text = text1.ToUpper();
      text = text1.ToLower();

    // ++Kopieren ab bestimmter Position
      text = text1.Substring(10);
      text = text1.Substring(10, 6);

    // ++Suche nach Zeichenfolge
      text = text1.IndexOf("Hallo").ToString();

    // ++Ersetzen einer Zeichenfolge
      text = text1.Replace("a", "A");
      text = text1.Replace("Hilfe", "Erledigt");

    // ++Letztes Zeichen entfernen
      text = text1.SubString(text1.Length - 1);
```

```
    // ++Auftrennen
      string[] text;
      string text = "Ich bin ein Text mit Leerzeichen";
      text = s.Split(' ');          // Text wird bei Leerzeichen getrennt und in Array gelegt

    // ++Vergleichen
      int i = String.Compare(text1, text2, false);
        switch(i) { case -1 : ausgabe = "String 1 << String 2"; break;
                    case  0 : ausgabe = "String 1 == String 2"; break;
                    case  1 : ausgabe = "String 1 >> String 2"; break;  }

    // ++String in Array kopieren
      string text = "Test";
      char [] arr = text.ToCharArray();
        for (int i = 0; i <= arr.GetUpperBound(0); i++);
          Console.WriteLine(arr[i].ToString());           // Zeigt "T", "e", "s", "t"

    // ++String zusammenführen (aus Array)
    text = String.Join("-", text1); //Trennzeichen -

// Grafik Device
  Graphics dev = e.Graphics;
  dev.SmoothingMode = System.Drawing.Drawing2D.SmoothingMode.AntiAlias; // Kantenglättung

// Eigene(r) Font, Stift, Pinsel, Dialog
  Font  def = new Font("Lucida Console", 11);
  Pen   def = Pens.Black;
  Pen   def = new Pen(def_farbe);
  Brush def = Brushes.Black;
  Brush def = new Brush(def_farbe);

  // ++Eigene Farbe (Alpha(Transparenz), Rot, Grün, Blau)
    Color std_farbe = Color.Black;
    Color def_farbe = Color.FromArgb(100, 20, 20, 20);

  // ++ColorDialog (Windows.Forms)
    colorDialog.ShowDialog();       //Dialog anzeigen (jede Farbe wählbar)
    def_farbe = colorDialog.Color;  //Definierte Farbe speichern

// Grafik Befehle
  // ++Text
    dev.DrawString("Hallo", Font, Brush, x, y); //Text ausgeben
  // ++Linien
    dev.DrawLine(Pen, x, y, width, height);
    dev.DrawEllipse(Pen, x, y, width, height);

  // ++Füllung
    dev.FillRectangle(Brush, x, y, width, height);
    dev.FillEllipse(Brush, x, y, width, height);


// Schleifen
  for (int x=0; x <= 10; x++)   //Zählen aufwärts
  for (int x=10; x > 0; x--)    //Zählen abwärts
  for (int x=0; x <= 10; x+=2)  //Zählen aufwärts +2 (x+=2 == x = x + 2)
  for (int x=10; x > 0; x-=2)   //Zählen abwärts  -2 (x-=2 == x = x - 2)

  while (true)  { if(err == true) break;  }   // Durchlauf solange err =! true
  while (x =! 2 && y == 1)  {   }             // Durchlauf solange x =! 2 UND y == 1

  foreach (int stop in x) { MessageBox.Show(stop.ToString());
                            if (stop == 5) break; }

  do
  {
    Console.Write("Endet mit x != 1 ODER y != 2");
  } while (x == 1 || y == 2);
```

```
// Switch
  switch (note) { case 1: Console.Write("Fall1"); break;  // Fall 1 und aussprung
                  case 2: Console.Write("Fall2"); break;  // Fall 2 und aussprung
                  default:Console.Write("VOID!"); }       // Fall nicht gefunden!

// Array
  array.lenght                                  // Anzahl der angeforderten Speicherplätze
  Array.Sort(array1);                           // Array Sortieren
  // ++Eindimensional
    double [] array;                            // Referenzvariable auf dem Stack
    array0 = new double[4] { 1,2,3,4  };        // 4 Werte werden Variable Array zugeordnet
    array1 = new double[4]; array1[0] = 10; // Werte einzeln zuweisen
                            array1[1] = 20;
                            array1[2] = 30;
                            array1[3] = 40;
    // Kurzform
    int [] wochentage = { 1,2,3,4,5,6,7 };
    string [] tagname = { "SO", "MO", "DI", "MI", "DO", "FR", "SA"  };

    // letzter Speicherplatz
    int last = array.Length - 1;                // letzter Speicherplatz im Array
    a[last] = 100;

    // Schleifenform
    int[] array = new int[100];                 // 101 Felder erzeugen
      for ( int i = 0; i != array.Lenth; i++)
        array[i] = i;                           // Felder jeweilt mit Wert i

    // Suchmodus
    for (i = 0; i < array1.Length; i++) {
      i = Array.BinarySearch(array1, 2);        // Suche nach Zahl 2
        if( i > 0 )
          Console.WriteLine("2 an " +i.ToString() + " gefunden");
        else
          Console.WriteLine("nichts gefunden!");  }


  // ++Zweidimensional
    double [,] 2darray;          // Referenz
    2darray = new double[1,6];  // Größe des Arrays definieren

    int[,] 2darray1 = new int[2, 3];    // Größe des Arrays definieren
    int[,] 2darray1 = { { 1, 2, 3 },
                        { 4, 5, 6 } };  // Werte direkt zuweisen
    int out = 2darray1[1,2];

// Strukturvariablen

  struct Telefon
    { public string name;
      public int anrufe;
      public string nummer;
      public bool intern;   }

    // Deklaration
    Telefon[] Benutzer = new Telefon[3];    // 4 Nutzer
      Benutzer[0].name  = "Max Musterman";
      Benutzer[0].nummer = "004312345678";
      Benutzer[1].name  = "Peter Dkovski";
      Benutzer[1].nummer = "004312345678";
      Benutzer[2].name  = "Dieter Poller";
      Benutzer[2].nummer = "004312345678";
      Benutzer[3].name  = "Hans Hannsons";
      Benutzer[3].nummer = "004312345678";
```

```
  struct Telefonbuch
    { public string bezirk;
      public Telefon[] Benutzer;  // Verweis auf obere Struktur
      public string betreuer; }

      Telefonbuch Telefonbuch1;
      Telefonbuch1.Benutzer = new Telefon[3];
        Telefonbuch1.bezirk = "Feldkirch";
        Telefonbuch1.Benutzer[3].name = "Hans Hannsons";
        Telefonbuch1.Telefonbuch.betreuer = "Ich";

    for (int i = 0; i < Telefonbuch1.Benutzer.Length; i++)  {
      if(Telefonbuch1.Benutzer[i].anrufe > 10)
        Telefonbuch1.Benutzer[i].intern = true; }

// Random
  Random generate = new Random();
  int x = generate.Next(1,7); // 1 < 7

// Funktionen und Methoden
  // ++Textausgabe
    public static void writetext()  {
      Console.WriteLine("Message 1"); } // Ausgabe von Message 1

    writetext();  // Aufruf in MAIN Routine

  // ++Zeichenausgabe
    public static void writechar()  {
      for (int i = 0; i <= 80; i++)     // Schleifenkörper
        Console.Write("+");             // Ausgabe von +
      Console.WriteLine();  }}          // Absatz einfügen

    writechar();

  // ++Externe Eingabe
    public static void writechar_ext( int start, int stop ) {
      for (int i = start; i <= stop; i++) // Schleifenkörper
        Console.Write("+");               // Ausgabe von +
      Console.WriteLine();  }}            // Absatz einfügen

    writechar_ext(0, 20);

  // ++Return
    public static double return_double( int i ) {
      double summe;
        for (i = 1; i <= i; i++)
          summe = summe + i;
      return summe; }             // return Wert = double (double return_double)

    double ergebnis = return_double(10);

  // ++ref Variablen
    public static void work_with( ref int x, ref int y ) {
        x = x * 2000;
        y = y * 2000; }

    int a = 10, b = 10;           // Definition der Globalen Variablen
    work_with( ref a, ref b);     // Variable wird überschrieben
    Console.Write( a + "," + b ); // Wert von a,b gleich 20000;

  // ++Mittelwert
    public static void double center(double[] datarow)  {
      double sum = 0;
      int z = datarow.Length;
        for (int i = 0; i < z; i++) {
          sum += datarow[i];
      return(sum / z);  }
```

```
// File Instructions

      string directory = @"C:\directory";
      string sub_directory = System.IO.Path.Combine(folderName, "sub_directory");
      string com_directory = @"C:\directory\subdirectory";
      string datafile = "NewFile.txt";

      System.IO.Directory.CreateDirectory(directory);
      sub_directory = System.IO.Path.Combine(sub_directory, datafile);  // Vorgegebener Dateinamen
      sub_directory = System.IO.Path.Combine(sub_directory, nothing);   // Random String für den Dateinamen

      Console.WriteLine("Dateipfad: {0}\n", sub_directory);
        if(!System.IO.File.Exists(sub_directory)) {                                        // Prüfen ob Datei
        bereits vorhanden
          using (System.IO.FileStream streaming = System.IO.File.Create(sub_directory)) // Dateisystem Stream
          erzeugen
          { streaming.WriteByte(x); }}                                                     // Bytes in Datei
          Schreiebn

      try { byte[] read_Buffer = System.IO.File.ReadAllBytes(sub_directory);             // Buffer erzeugen
              foreach (byte b in read_Buffer)                                            // Datei auslesen
                Console.Write(b + " ");
              Console.WriteLine();  }
      catch (System.IO.IOException e) {
        Console.WriteLine(e.Message); }

        if(!System.IO.File.Exists(sub_directory)) {                                        // Prüfen ob Datei
        bereits vorhanden
          using (System.IO.FileStream streaming = System.IO.File.Create(sub_directory))   // Dateisystem
          Stream erzeugen
          { Byte[] info = new UTF8Encoding(true).GetBytes("Text in a File");               // String in Datei
          speichern
            Filestream.Write(info, 0, info.Lenth);  }

          using (System.IO.StreamReader reading = System.IO.File.OpenText(sub_directory))
          { string readout = "";
            while ((readout = reading.ReadLine()) != null)
            { Console.WriteLine(readout); }}}

        if(File.Exists(sub_directory))  {
            File.Delete(sub_directory); }

// Zugriff auf Komponenten
    // ++Serielle Schnitstelle UART/USART

    //  ++USB Schnitstelle

    // ++Soundkarte
```