APLIKASI
Visual Basic

Visual Basic for Applications (VBA) untuk Pemula

Dr. Budi Raharjo, S.Kom, M.Kom, MM

YAYASAN PRIMA AGUS TEKNIK

APLIKASI
Visual Basic
Visual Basic for Applications (VBA) untuk Pemula

Dr. Budi Raharjo, S.Kom, M.Kom, MM

BIODATA PENULIS

Dr. Budi Raharjo, S.Kom, M.Kom, MM lahir di Semarang, tanggal 22 Februari 1985. Beliau adalah Alumni dari Universitas Bina Nusantara (BINUS University) Jakarta dan juga alumni Universitas Kristen Satya wacana (UKSW) Salatiga. Dr. Budi Raharjo telah menjadi Dosen pada Universitas STEKOM pada mata kuliah Kepemimpinan (Leadership), mata kuliah Pengantar Akuntansi, mata kuliah Pengantar Ekonomi Mikro, Manajemen Proses, Manajemen Akuntansi dan Manajemen Resiko Bisnis. Selain sebagai dosen di Universitas STEKOM.

Dr. Budi Raharjo, S.Kom., M.Kom., MM juga mempunyai bisnis sendiri dalam bidang perhotelan dan juga sebagai wirausaha dalam bidang pemasok unggas (ayam) beku ke berbagai kota besar, khususnya Jakarta dan sekitarnya.

Pengalaman beliau berwirausaha menjadi bekal utama dalam penulisan buku ajar yang diterbitkan oleh Yayasan Prima Agus Teknik (YPAT) Semarang. Oleh sebab itu bukunya berisi langkah langkah praktis yang mudah diikuti oleh para mahasiswa, saat mahasiswa mengikuti proses perkuliahan pada Universitas Sains dan Teknologi Komputer (Universitas STEKOM). Jabatan struktural yang di embannya saat ini adalah Wakil Rektor 1 (Akademik) Universitas STEKOM Semarang.

YAYASAN PRIMA AGUS TEKNIK

PENERBIT :
YAYASAN PRIMA AGUS TEKNIK
Jl. Majapahit No. 605 Semarang
Telp. (024) 6723456. Fax. 024-6710144
Email : penerbit_ypat@stekom.ac.id

APLIKASI
Visual Basic
Visual Basic for Applications (VBA) untuk Pemula

Dr. Budi Raharjo, S.Kom, M.Kom, MM

YAYASAN PRIMA AGUS TEKNIK

PENERBIT :
YAYASAN PRIMA AGUS TEKNIK
Jl. Majapahit No. 605 Semarang
Telp. (024) 6723456. Fax. 024-6710144
Email : penerbit_ypat@stekom.ac.id

Aplikasi Visual Basic, Visual Basic for Application (VBA) untuk Pemula

Penulis :
Dr. Budi raharjo, S. Kom., M.Kom., MM

ISBN : 9 786239 504281

Editor :
Dr. Joseph Teguh Santoso, S.Kom., M.Kom.

Penyunting :
Mars Caroline Wibowo, S.T., M.Mm.Tech

Desain Sampul dan Tata Letak :
Irdha Yunianto, S.Ds

Penerbit :
Yayasan Prima Agus Teknik Bekerja sama dengan
Universitas Sains & Teknologi Komputer (Universitas STEKOM)

Redaksi :
Jl. Majapahit no 605 Semarang
Tel. (024) 6723456
Fax. 024-6710144
Email : **penerbit_ypat@stekom.ac.id**

Distributor Tunggal :
Universitas STEKOM
Jl. Majapahit no 605 Semarang
Tel. (024) 6723456
Fax. 024-6710144
Email : **info@stekom.ac.id**

KATA PENGANTAR

Buku ini merupakan panduan, tentang langkah demi langkah untuk belajar bahasa pemrograman Excel bagi pemula. Jika kita sudah mengetahui cara menggunakan Microsoft Excel, namun masih banyak hal yang tidak dapat kita lakukan, sekarang saatnya kita untuk mempelajari fungsi dari Visual Basic for Applications (VBA).

Visual Basic for Applications adalah bahasa pemrograman yang tergabung dalam Microsoft Excel, Access, PowerPoint, dan bahkan Word, yang memungkinkan kita untuk melakukan semua hal. Misalnya, kita ingin setiap kali kita membuka file Microsoft Word tertentu, kita ingin menulis secara otomatis tanggalnya, pada dua baris di bawah tempat yang kita tinggalkan terakhir kali. Barangkali kita menginginkan seluruh spreadsheet Excel tanpa rumus di atasnya dan tetap menerapkannya seolah-olah ada di sana. Bagaimana kita dapat melakukannya? Semua hal ini dapat dilakukan dengan Visual Basic for Applications for Microsoft Office.

Buku ini akan mengupas secara tuntas, sehingga kita dapat belajar lebih banyak dari itu dan kita akan dapat membuat program kita sendiri dengan menggunakan Visual Basic for Applications (VBA). Jika kita memerlukan program yang sangat spesifik untuk analisis bisnis kita, atau sesuatu yang bersifat untuk penggunaan pribadi, atau bahkan hanya untuk bersenang-senang, maka kita memerlukan Aplikasi Visual Basic seperti itu sekarang. Semoga buku ini memberi manfaat bagi kita semua.

Semarang, 15 Februari 2021

Dr. Budi Raharjo, S.Kom, M.Kom, MM

Daftar Isi

Bab 1
Visual Basic Tab Developer

Pendahuluan

Bagi orang yang berhubungan langsung dengan komputer pasti pernah mendengar dan terlibat dengan Visual Basic, baik itu di sekolah, kantor maupun tempat kerja. Visual Basic adalah bahasa pemrograman komputer berupa perintah atau instruksi-instruksi yang dapat dimengerti komputer untuk menjalankan tugas tertentu.

Microsoft Visual Basic (sering disingkat sebagai VB) merupakan bahasa pemrograman yang bersifat event driven dan menawarkan Integrated Development Environment (IDE) visual untuk membuat program aplikasi berbasis sistem operasi Microsoft Windows dengan menggunakan model pemrograman Common Object Model (COM).

Kegunaan Visual Basic adalah untuk membuat program berbasis Windows mulai yang sederhana sampai pemrograman yang lebih kompleks. Contohnya adalah pembuatan aplikasi kasir atau perpustakaan. Untuk membuat aplikasi sederhana dengan visual basic maka kita harus menguasai bahasa pemrograman C++. Visual Basic yang paling banyak digunakan adalah Microsoft Visual Basic.

Microsoft pertama kali mengembangkan bahasa pemrograman Visual Basic ini pada tahun 1991 yang sebenarnya adalah pengembangan dari program terdahulu disebut dengan nama Bahasa Pemrograman Basic atau juga dikenal dengan istilah Beginner of All Purpose Symbol yang fungsinya untuk dijadikan alat bantu pembuatan berbagai jenis program komputer khususnya program komputer yang menggunakan Windows sebagai sistem operasinya.

Pada dasarnya, bahasa pemrograman Visual Basic ini asalnya dari program BASIC yang merupakan jenis bahasa pemrograman yang termasuk user-friendly sebab memang sengaja dirancang untuk pemula. Oleh sebab itu, Visual Basic juga memungkinkan adanya pengembangan aplikasi grafis antarmuka yang bukan hanya cepat namun juga memiliki akses ke berbagai database.

VBA adalah singkatan dari Visual Basic for Applications. Adalah bahasa pemrograman dan pengembangan yang dibuat oleh Microsoft. Ini adalah perpanjangan dari bahasa pemrograman BASIC yang menggabungkan fungsi BASIC dan perintah dengan kontrol visual. Visual Basic menyediakan antarmuka pengguna grafis GUI yang memungkinkan pengembang untuk drag dan drop objek ke dalam program maupun program dengan menulis kode secara manual.

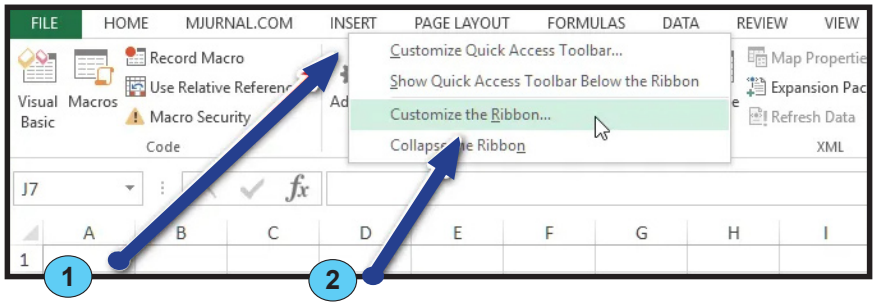
Visual Basic, juga disebut sebagai "VB," dirancang untuk membuat pengembangan perangkat lunak yang mudah dan efisien, dan masih sangat tangguh untuk membuat program tingkat lanjut. Sebagai contoh, bahasa Visual Basic dirancang untuk menjadi "human readable" yang berarti kode sumber dapat dipahami tanpa memerlukan banyak komentar. Program Visual Basic juga termasuk fitur seperti "IntelliSense" dan "Code Snippets," yang secara otomatis menghasilkan kode untuk objek visual yang ditambahkan oleh programmer. Fitur lain, yang disebut "AutoCorrect," bisa debug kode ketika program berjalan.

Beberapa program besar seperti Microsoft Office, Corel Draw dan Photoshop juga menambahkan fitur Visual Basic dalam programnya sepaket dengan aplikasi. Visual Basic dalam Microsoft Excel atau bisa juga disingkat VBA sangat populer digunakan, ini karena banyak sekali manfaat untuk mengolah aplikasi yang dibuat dengan Microsoft Excel agar aplikasinya lebih interaktif dan pembuatannya lebih cepat.

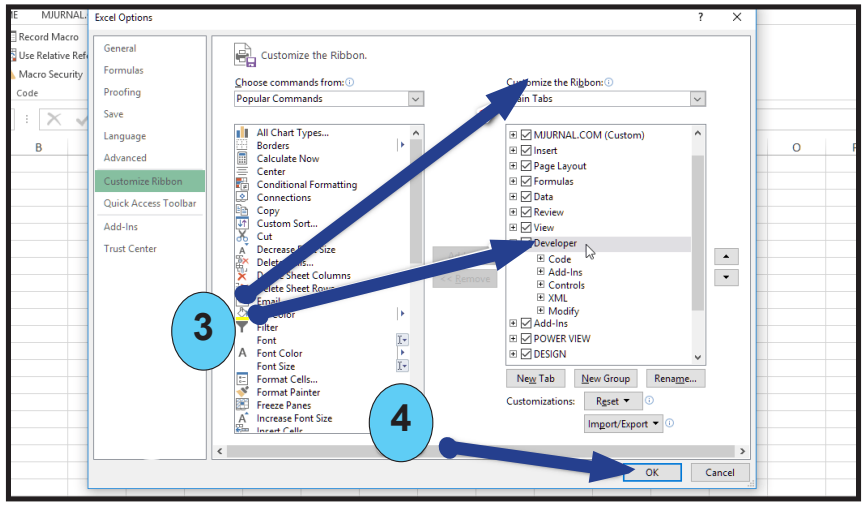
1.1 Cara Mengakses Tab Developer

Secara garis besar cara untuk menampilkan Tab Developer pada microsoft excel 2010, excel 2013 dan excel 2016 sebenarnya sama, yakni dengan membuka excel option pada kategori Customize Ribbon.

Jalankan program Microsoft Office Excel.
Langkah Pertama, Klik Kanan pada bagian Ribbon manapun.
Langkah Kedua, Klik Customize The Ribbon. Kemudian, muncul Window Excel Option Gambar 1.1 dan lakukan langkah – langkah berikut:



Gambar 1.1



Gambar 1.2

Langkah Ketiga, pada bagian Customize the Ribbon, silahkan centang box Developer (seperti gambar).

Langkah Terakhir, Klik Ok untuk melanjutkan dan hasilnya bisa dilihat pada gambar 1.2.

BAB 2
Macro VBA

2.1 Membuat Folder Khusus Belajar VBA Excel

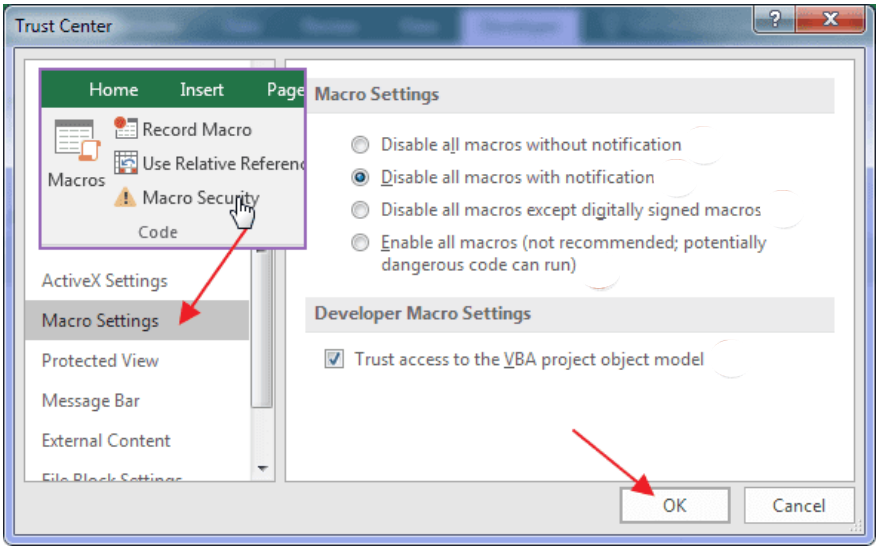
Dalam Ms. Excel terdapat fitur untuk menyimpan aktivitas yang kita lakukan yaitu tools -> macro -> record new macro. Record new macro ini berfungsi untuk merekam aktivitas yang kita lakukan dan mengubahnya menjadi baris-baris program. Macro sendiri merupakan deretan fungsi dan perintah program yang disimpan dalam menu Visual Basic.

Macro ini dapat dipakai oleh pengguna untuk membuat perintah menggunakan bahasa Visual Basic for Application (VBA). Fungsi macro ini dalam Excel dikenal sebagai User Defined Function (UDF).

Fungsi Macro adalah untuk menuliskan pekerjaan yang banyak, berulang, dan monoton dengan sekali klik. Jika Kamu melakukannya secara manual, tentu akan memakan banyak waktu.

Jadi, kita dapat merekamnya menggunakan Macro ini, harus mengubah setting pada macro security level di Ms. Excel. Atur security di level medium untuk menghindari file excel yang mengandung virus Macro. Tinggal aktifkan menu enable atau disable macro di file excel tersebut
Langkah pertama yang perlu dilakukan adalah membuat sebuah folder khusus untuk menyimpan hasil belajar VBA serta file-file hasil praktek. Terserah dimana kita akan meletakkan folder tersebut. Misalnya boleh menyimpan folder ini di Drive D dan berikan nama untuk folder ini "Stekom VBA".

2.2 Pengaturan untuk Check Security



Gambar 2.1

Untuk memastikan kode-kode VBA yang akan kita tulis berjalan normal pastikan bahwa excel yang Kamu gunakan sudah mengijinkan penggunaan macro. Caranya sebagai berikut:

Pada bagian TAB Developer, klik "Macro Security" yang ada di group "Code". Setelah muncul window Trust center atur saja seperti pada gambar 2.1

Keterangan:

- ***Disable all macros without notifications***: Macro disable tanpa notifikasi. Macro tidak boleh dijalankan sama sekali
- ***Disable all macros with notifications***: Macro disable dengan notifikasi, macro bisa berjalan jika diijinkan oleh User.
- ***Disable all macros except digitally signed macro***s: Opsi ini hanya akan mengijinkan Macro (VBA) yang telah disertai kartu digital. Penjelasan lebih lanjut bisa Kamu pelajari pada halaman ini.

- ***Enable all macros (not recommended, potentially dangerouscode can run***): Mengijinkan semua macro untuk dapat dijalankan pada excel
- ***Trust Acces to the VBA project object model***: Item Selectan ini merupakan ijin khusus untuk dapat mengakses komponen VBProject. Jika dicentang, maka akan diberikan ijin untuk mengakses komponen VBProject.

2.3 Pengaturan untuk Trusted Locations

Selain menggunakan digital signature, untuk file excel yang mengandung VBA terpercaya bisa melewati bagian security macro dengan meletakkan file tersebut pada sebuah lokasi terpercaya atau trusted locations.

Dengan menyimpan file pada folder terpercaya ini maka file excel yang berisi kode-kode macro VBA dapat berjalan tanpa pemeriksaan sehingga memunculkan peringatan atau notifikasi seperti biasanya. Lokasi ini biasa diatur pada Trust Center pada bagian Trusted Locations.

Berikut langkah-langkah pengaturan yang perlu Kamu lakukan untuk memasukkan folder "KelasVBA" yang telah kita buat sebelumnya sebagai salah satu lokasi yang dipercaya (trusted locations) :

Langkah 1

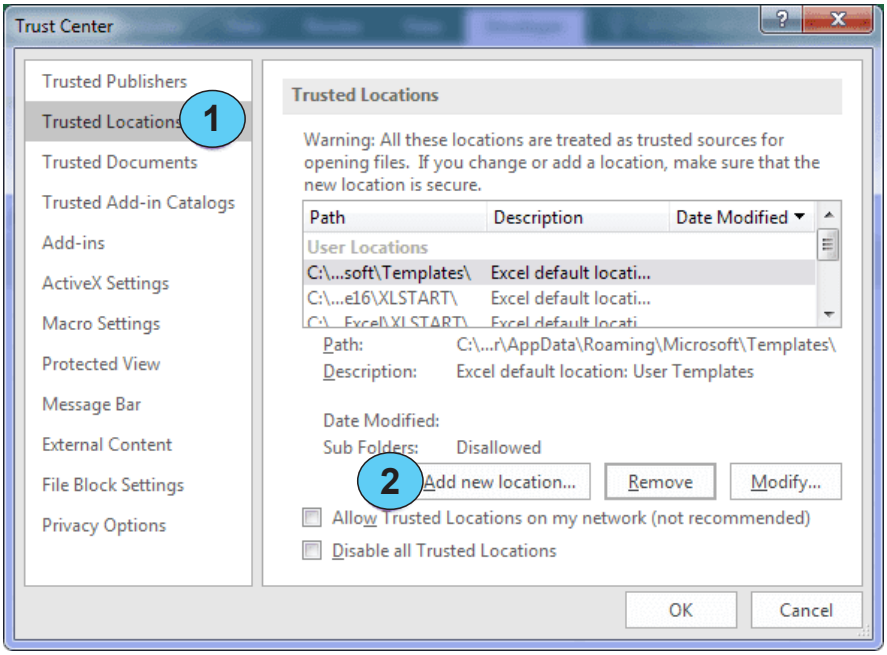
Pada bagian TAB Developer, klik "Macro Security" yang ada digroup "Code" seperti sebelumnya.

Langkah 2

Setelah muncul window Trust center Select bagian "Trusted Locations"

Langkah 3

Berikutnya Select Add new location. (Gambar 2.2)



Gambar 2.2

Langkah 4

Select tombol menu Browse dan cari lokasi folder yang akan dipercaya tadi.

Langkah 5

Jika sub folder atau folder lain yang ada di dalam folder yang kita Select juga akan dimasukkan ke dalam lokasi terpercaya, centang bagian Subfolders of this location are also trusted

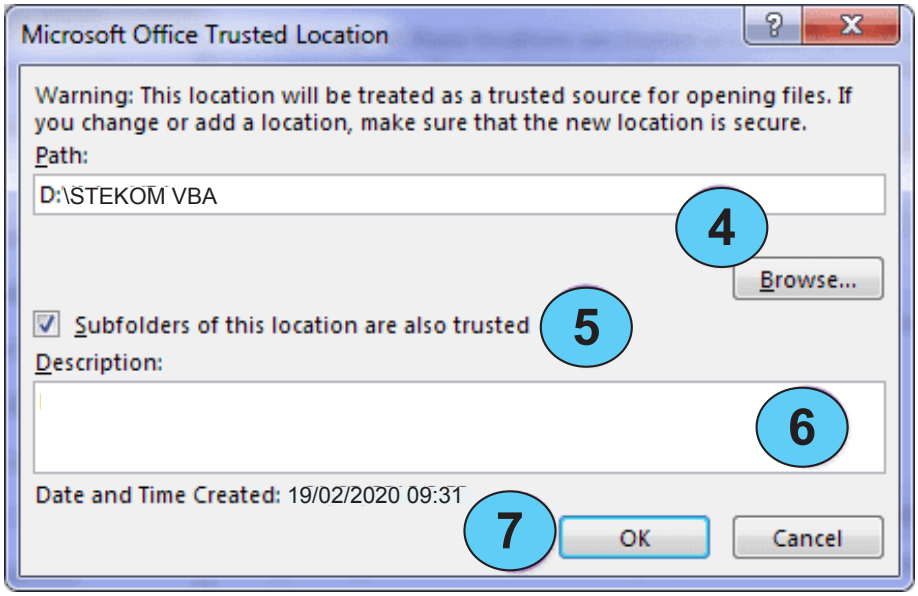
Langkah 6

Selanjutnya berikan deskripsi untuk lokasi terSelect (opsional)

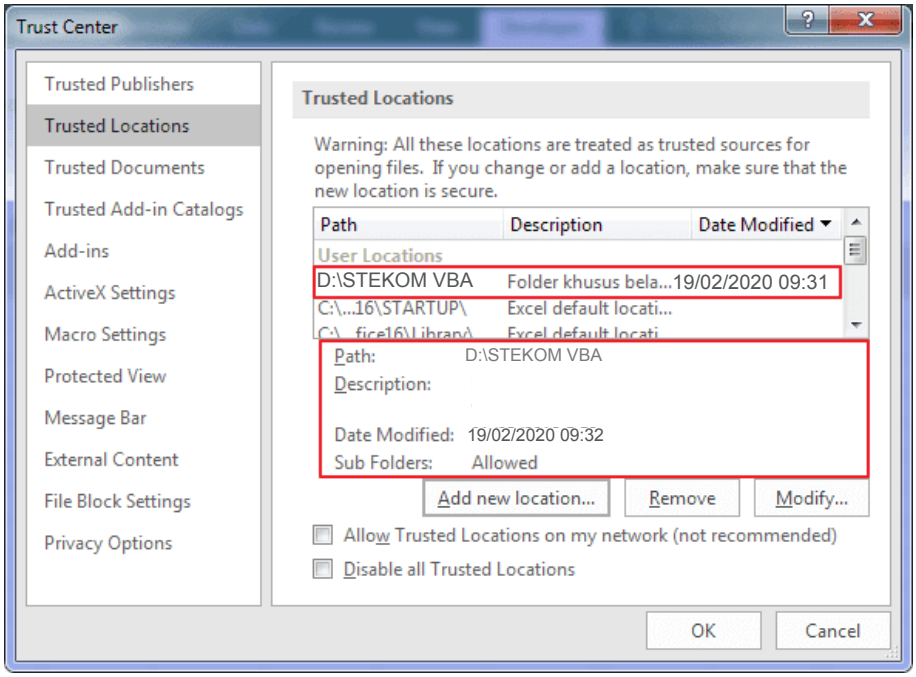
Langkah 7

Klik OK dan OK lagi untuk keluar pengaturan. Gambar 2.3

Apabila langkah-langkah yang Kamu lakukan benar maka hasilnya akan seperti yang terlihat pada Gambar 2.4



Gambar 2.3



Gambar 2.4

2.4 Membuat Macro

Kegunaan macro adalah apabila kita melakukan pekerjaan yang banyak dan dilakukan secara manual. Dari pada kita membuang-buang waktu dengan mengerjakan pekerjaan yang banyak secara manual, kita bisa merekamnya dengan fungsi macro kemudian melakukan sedikit modifikasi.

Coba bayangkan Kamu menerima file Microsoft Excel dari atasan dengan beberapa data dan memrintahkan untuk menulis tanggal menggunakan No, Tahun, Bulan dan Hari di kolom yang berbeda. Kamu melakukan pekerjaan menyusun data seperti ini secara berulang-ulang dengan data yang berganti-ganti. Dalam hal ini, proses semi-otomatis akan sangat membantu. Excel memberikan opsi itu kepada kita semua dengan Macro. Macro adalah proses semi-otomatis yang memungkinkan kita menjalankan tugas tertentu menggunakan jalur pintasan.

2.5 Langkah-Langkah mengaktfikan Macro

Langkah pertama, buka dahulu Microsoft Excel. Jangan lupa untuk mengaktifkan menu Developers.

Langkah - langkah selanjutnya adalah sebagai berikut:

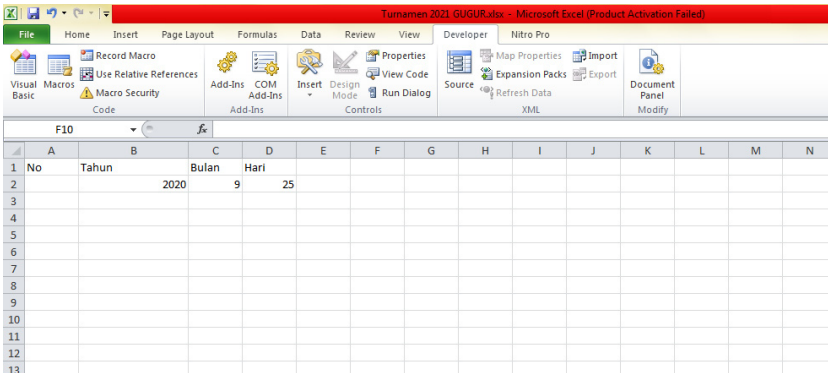
Langkah 1

Pembuatan tabel. Buatlah sebuah tabel terlebih dahulu, terserah diisi apa, disini saya membuat tabel.

Langkah 2

Starting. Arahkan dan aktifkan cell pada kolom A (terserah pada baris berapa). Kemudian tekan CTRL + Arah Bawah, maka cell akan berpindah ke cell yang paling bawah yaitu cell A1048576. Hal ini tidaklah terlalu penting, hanya saja kita akan memulai perekaman maka kita harus menentukan titik aktif cell dimana kita mulai merekam, karena segala aktifitas setelah kita mengeklik

Macro Recorder maka semua aktifitas yang akan kita lakukan akan terekam menjadi prosedur, maka dari itu kita harus menentukkan titik tersebut terlebih dahulu. Lihat Gambar 2.5



Gambar 2.5

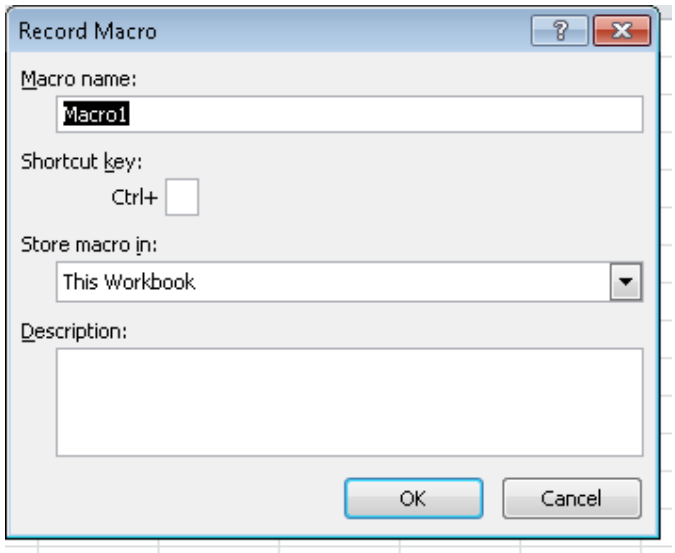
Langkah 3

Memulai Macro Recorder. Sekarang kita akan mulai merekam, navigasikan kursor dan klik icon Macro Recorder yang ada pada menu Developer.

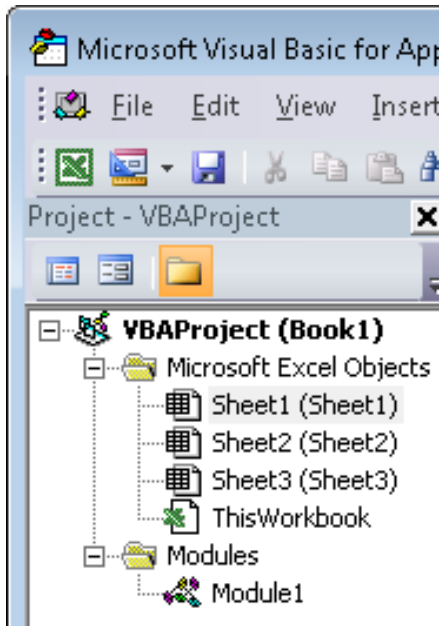
Kemudian akan muncul sebuah dialog box untuk pemberian nama Macro, diisi terserah atau dibiarkan saja kemudian klik OK maka Macro akan selesai dibuat. Gambar 2.6

Langkah 4

Perekaman. Pastikan cell masih aktif dan jangan dipindah - pindah terlebih dahulu. Kemudian, tekan CTRL + Arah Atas untuk mengalokasikan cell ke paling atas. Maka cell akan berhenti dan aktif pada cell A1, hal ini dikarenakan ketika menekan tombol tersebut kursor akan menuju ke cell atasnya yang memiliki konten, karena ada konten dengan tulisan Kode, maka cell akan berhenti di cell tersebut. Setelah itu, tekan Stop Recording pada menu Developer untuk mengakhiri perekaman. Sekarang Macro sudah dibuat dengan nama Macro1, klik icon View Code pada menu Developer. Pada Window VBA Project, klik pada sub-folder Module1. Gambar 2.7

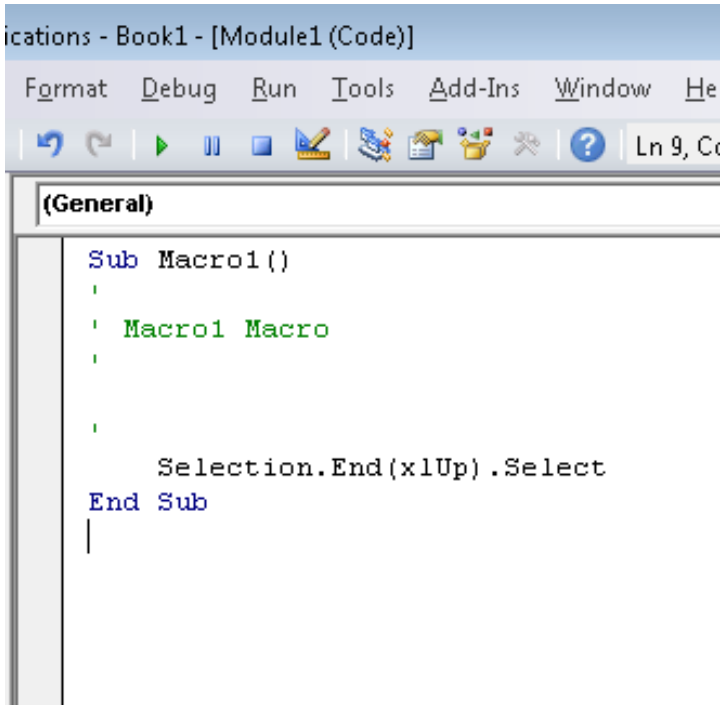


Gambar 2.6



Gambar 2.7

Disini, Macro yang kita rekam akan muncul dengan berbentuk barisan-barisan kode seperti gambar 2.8 dibawah ini:



Gambar 2.7

Sub Macro1() marupakan sebuah sintaks untuk memulai prosedur tersebut, dengan nama Macro1, kita dapat mengganti nama tersebut sesuai dengan keinginan kita. Baris yang diawal dengan tKamu petik satu/apostrophe (') itu hanyalah komentar dan tidak memengaruhi pada eksekusi kode. Selection.End(xlUp). Select merupakan sintaks gabungan yang memberikan perintah seleksi cell untuk mengaktifkan yang terdekat cell diatasnya. End Sub berfungsi mengakhiri sebuah Sub/Prosedur Macro tersebut.

Sekarang tambahkan diatas kode End Sub dengan kode berikut:

```
ActiveCell.Offset(1,0).Select
ActiveCell.Value = "=ROW()-1"
```

Maksud dari argumen Offset(1,0) merepresentasikan angka pertama sebagai Row/Baris dan angka kedua sebagai Column/Kolom. Fungsi ini untuk memindahkan cell yang aktif dengan settingan baris bergeser sebanyak 1 cell dan kolom berpindah 0 cell alias tetap, maka cell A2 akan aktif.

Value berfungsi untuk memberikan nilai atau value "=ROW()-1" yang mana konten itu saya tulis berupa rumus Formula Excel untuk memberikan penomoran kedalam suatu cell dan cell yang diSelect adalah yang sedang aktif.

Referensi Relatif

Tombol Referensi Relatif tepat di bawah tombol Rekam Macro. Setelah mengkliknya, itu tetap aktif sampai kita mengkliknya lagi. Komponen ini digunakan untuk merekam Macro yang prosesnya harus diterapkan ke Range yang berbeda, bukan yang sudah ditetapkan. Fungsinya sangat berguna. Macro yang direkam tanpa referensi relatif akan selalu mengulangi proses pada sel yang sama yang digunakan saat direkam. Tetapi jika menggunakan referensi relatif, Macro akan dijalankan dari sel aktif.

Menggunakan contoh di atas, bagaimana jika Kamu memerlukan tanggal yang ditulis pada sel F4: H4, bukan B2: D2? Satu-satunya hal yang harus dilakukan adalah memilih F4 dan menjalankan Macro. Atau Select sel yang dibutuhkan dan jalankan. Tetapi disini perlu merekam Macro menggunakan Relatif.

2.6 Cara Menyimpan File Macro

Setelah kita membuat macro menggunakan Record macro atau menuliskan baris perintah langsung dalam Visual Basic Editor, kita dapat simpan macro tersebut agar dapat kita jalankan kembali. bagaimana cara menyimpan file macro ??? Menyimpan file excel macro berbeda dengan file excel biasa. Pada versi Microsoft excel 2003 Kamu dapat menyimpanya bersamaan dalam ekstensi .xls akan tetapi pada versi excel 2007 sampai dengan excel 2016 terjadi penambahan fitur XML sehingga menyimpan file macro sedikit berbeda. Berikut ini adalah tipe file excel Xml untuk penyimpanan macro. Lihat Tabel 2.1

- Excel Workbook, Menyimpan dengan tipe ini seperti menyimpan file excel atau workbook pada umumnya. Pada file ini tidak terdapat Macro.
- Excel Macro-Enable Workbook, Menyimpan dengan tipe ini digunakan untuk menyimpan file yang terdapat macro didalamnya. Macro yang sudah dibuat akan tersimpan di dalam file.
- Excel Binary Workbook, Menyimpan dengan tipe ini digunakan untuk menyimpan file yang terdapat macro didalamnya akan tetapi macro akan tersimpan pada system Microsoft Office Excel.
- Excel Add-In, Menyimpan dengan tipe ini digunakan untuk menyimpan file macro yang dapat Kamu gunakan sebagai Add-in atau program tambahan dalam Program Excel.

Tabel 2.1

Tipe File Excel XML	Extension
Excel Workbook	.XLSX
Excel Macro-Enable Workbook	.XLSM
Excel Template	.XLTX
Excel Macro-Enable Template	.XLTM
Excel Add-In	.XLAM
Excel Binary workbook	.XLSB

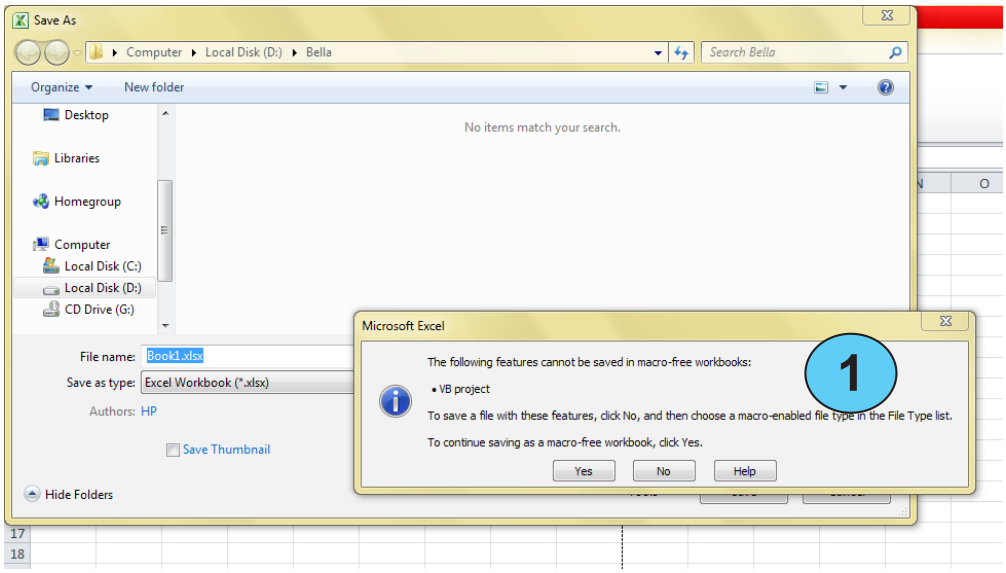
Langkah-langka Cara menyimpan file Macro :

Menyimpan file excel macro dari workbook Excel biasa (,xlsx)

Langkah 1

Lakukan Penyimpanan dengan tombol SAVE

Akan tampil peringatan “the following features cannot be saved in Macro-free workboks” peringatan ini menginformasikan bahwa file ini terdapat VB Project (macro). Jika Kamu Select Yes maka Macro yang sudah dibuat akan dihapus, dikarenakan workbook biasa tidak diperkenenkan untuk menyimpan macro. Gambar 2.8. Ini mungkin sedikit rumit, karena kebanyakan orang akan mencoba menyimpan file tanpa membaca pemberitahuan ini:



Gambar 2.8

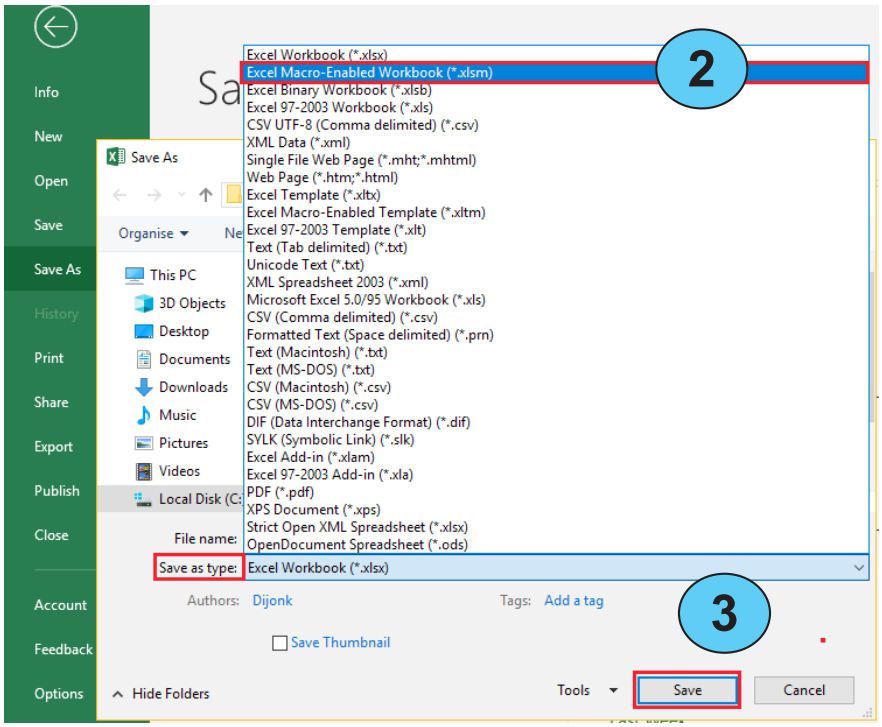
Langkah 2

Select No Untuk menyimpan macro.

Lakukan penyimpanan didalam folder tujuan. Ubahlah Save as Type Menjadi Excel macro-enable Workbook (xlsm). Gambar 2.9

Langkah 3

Klik Save



Gambar 2.9

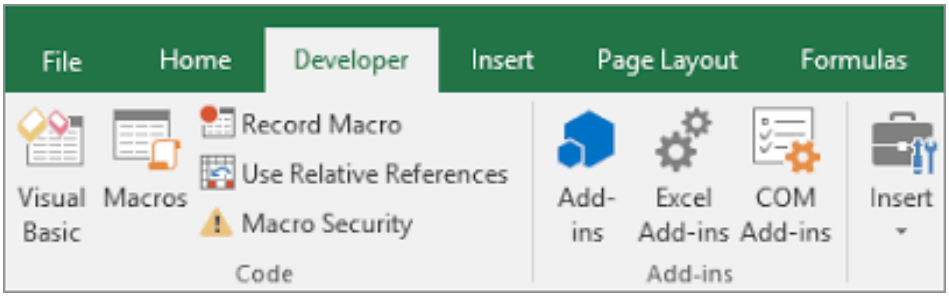
Ketika ingin membuka lagi, akan ada pesan yang mengatakan Macro telah dinonaktifkan (Macros have been disableb), dan Tombol bertuliskan Aktifkan Macro (Enable Macro) Klik di atasnya dan tidak akan ada masalah lebih lanjut. Jika tidak mengkliknya, maka tidak akan bisa bekerja dengan VBA, setidaknya Kamu mengaktifkannya di Macro Security di TAB Developer.

Untuk menyimpan file macro dari workbook biasa disarankan menggunakan perintah Save As untuk menghindari kesalahan pada saat menyimpan file macro.

Bab 3
Berlatih Menggunakan VBA Macro

Macro sangat penting untuk diperkenalkan ke VBA. Kiata dapat mengetahuinya alasannya dengan mengikuti prosesnya sebagai berikut :

1. Buatlah Macro baru tanpa referensi relatif.
2. Dalam prosesnya Select sel A1, tulis angka 1, dan tekan enter.
3. Berhenti Merekam. (Stop Recording)



Lanjut ke Tombol Record Macro, atau juga disebut Visual Basic.
Klik di atasnya dan akan terlihat kode seperti ini. Gambar 3.1

```
Range("A1"). Select
    ActiveCell.FormulaR1C1 = "1"
Range("A2"). Select
```

Selamat, sekarang kita memiliki beberapa Kode Visual Basic. Kemudian lakukan perintah berikut ini:

- Select Sel A1
- Tuliskan nomornya 1
- Select Cell A2

Persis seperti itulah yang Kita lakukan. Tapi sekarang, Kita akan mengedit kodenya sehingga Macro melakukannya sesuatu yang lain. Jika kita lihat, ternyata ada pola yaitu: *Select, Write, Select*.

Jadi, kita bisa melanjutkan polanya dengan menambahkan beberapa hal secara langsung, seperti ini:

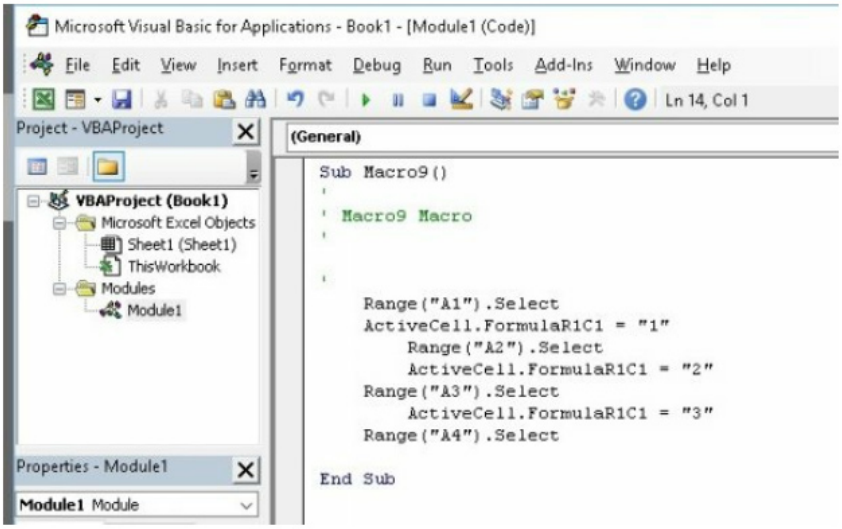
```
    Range ("A1"). Select
        ActiveCell.FormulaR1C1 = "1"
Range ("A2"). Select
        ActiveCell.FormulaR1C1 = "2"
    Range ("A3"). Select
        ActiveCell.FormulaR1C1 = "3"
    Range ("A4"). Select
```

Sekarang setelah menambahkan ini, jalankan Macro, tetapi sekarang kita akan melihat cara kedua untuk menjalankannya. Tekan tombol hijau di atas kode. Setelah mengkliknya, buka Excel Spreadsheet dengan mengklik simbol Excel. akan terlihat bahwa sel A1: A3 diisi dengan angka yang kita tulis dan Sel A4 di Select.

Ini mungkin terlihat rumit karena manusia biasa melakukan hal-hal yang tidak dapat dilakukan mesin. Coba memilih sel A1, A2, A3 dan seterusnys. Tetapi apakah Excel benar-benar perlu memilihnya untuk menulis angka sederhana?

Sekarang coba dengan menambahkan kode berikut: Gambar 3.1

```
        Range ("A1") = 1
        Range ("A2") = 2
Range ("A3") = 3
```



Gambar 3.1

Itu dia! Excel bisa melewati langkah-langkah yang tidak bisa dilakukan manusia. Jadi, dengan rumus ini pekerjaan akan lebih mudah dan cepat.

3.1 Mengisi Sel Hingga 100

Secara manual jika mengikuti pola ini hingga A100 akan sulit. Apa yang harus saya lakukan?
Mari tambahkan ini:

Range ("A1: A100") = 1

Secara otomatis akan menambahkan angka 1 ke semua sel dari A1 ke A100. Mari coba hal lain:

Range ("A1: A100") = 1 + 1

Ia menambahkan angka dua sebagai gantinya. Jadi, Bagaimana cara membuat angka dan mengisi sel secara berurutan?
Ada beberapa cara. Salah satunya adalah merekam Macro dan selama proses menambahkan angka 1 ke sel A1, lalu tahan klik kanan pada kotak kecil dan gulir ke bawah hingga sampai pada kolom A100. Select seri isi. Buka Visual Basic dan akan nampak kode seperti ini:

```
Range ("A1"). Select
    ActiveCell.FormulaR1C1 = "1"
    Selection.AutoFill Destination: = Range ("A1: A100"),
Type: = xlFillSeries
 Range ("A1: A100").Select
```

Berhasil! Jadi cara untuk mendapatkan kode VBA yang diinginkan adalah dengan menggunakan *Record Macro.* Namun, ada banyak fungsi yang tidak dapat ditawarkan oleh Macro. Misalnya mengisi sel secara berurutan sesuai dengan angka yang tertulis di sel B2, dan setelah berubah isikan lagi urutannya sesuai dengan angka tersebut. Jadi kalau ada angka 1 diisi satu per satu, kalau saya rubah jadi lima jadi lima kali lima dan seterusnya. Bagaimana Kamu akan merekamnya di Macro? Bagaimana jika tidak hanya mengisi urutan, tetapi masalah keuangan, bekerja dengan angka-angka?

Catatan : Merekam Macro tidak selalu cukup, tetapi dalam banyak kasus, ini sangat membantu.

Aplikasi Visual Basic (VBA) adalah bahasa pemrograman, tetapi tidak perlu mengetahui kode VBA atau pemrograman komputer jika Macro Recorder melakukan apa yang kita inginkan.

Kita harus tahu bahwa ketika merekam Macro, ia merekam semuanya bahkan kesalahan yang kita buat, dan itu akan terulang saat menjalankannya lagi. Jika ingin menyelesaikan masalah seperti ini, ada dua pilihani:

1. Rekam Macro lagi.
2. Edit kode VBA.

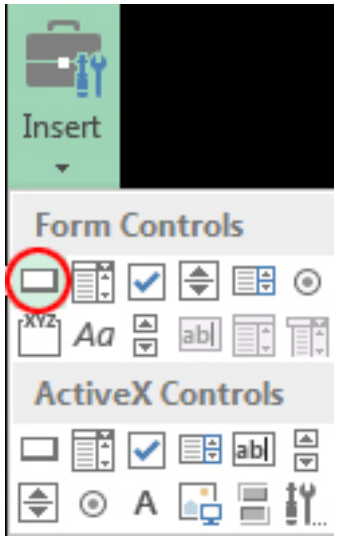
Ingatlah bahwa merekam Macro yang baik atau menulis kode VBA yang baik, akan membuat Excel berjalan dengan lancar. Jika tidak, akan muncul pesan bahwa program tidak merespons atau bahkan mungkin berhenti bekerja. Kita akan fokus pada hal-hal yang tidak dapat dilakukan dengan hanya menggunakan Macro yang direkam. Jadi, kita dapat mempelajari betapa hebatnya Visual Basic for Applications di Microsoft Excel.

3.2 Menyisipkan Data dengan Tombol Button

Kita memiliki beberapa Macro yang direkam sekarang. Macro VBA Excel menyimpan banyak kemudahan. Setelah Kamu mengetahui bagaimana membuat Sub dengan sederhana, kita akan mengaktifkan sebuah tombol pada spreasheet excel. Caranya pun tergolong mudah.

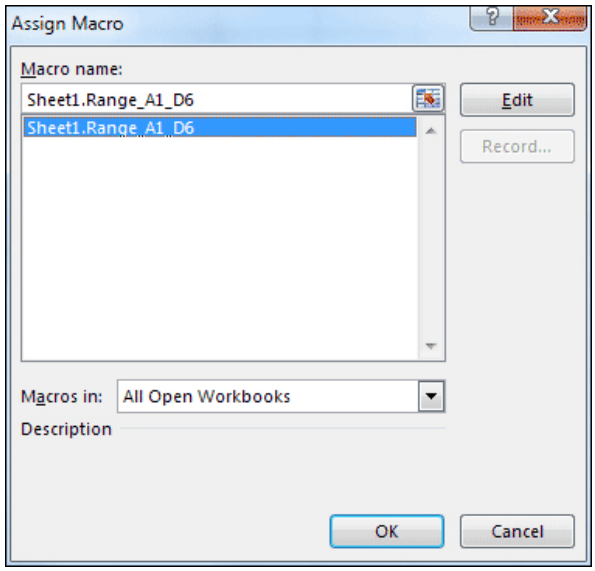
Untuk melakukan itu, klik tombol sisipkan di TAB Pengembang. Kamu akan melihatnya menampilkan dua kotak, yang disebut *form controls* dan Kontrol *ActiveX*, Gambar 3.2.

Sekarang pindahkan mouse Kamu ke spreadsheet Kamu. Tahan tombol kiri mouse Kamu di suatu tempat pada kolom F (biasanya akan jatuh pada F3). Tetap tahan dan tarik maka akan tercipta sebuah tombol persegi panjang. Lepaskan tombol kiri mouse saat kursor Kamu di H4.



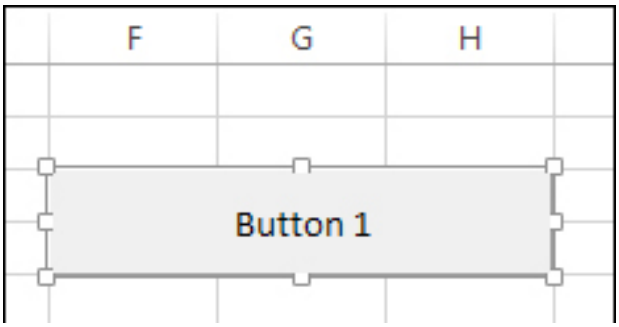
Gambar 3.2

Segera setelah Kamu melepaskan tombol kiri mouse Kamu akan melihat kotak dialog Assign Macro muncul pada Macro VBA Excel; Gambar 3.3

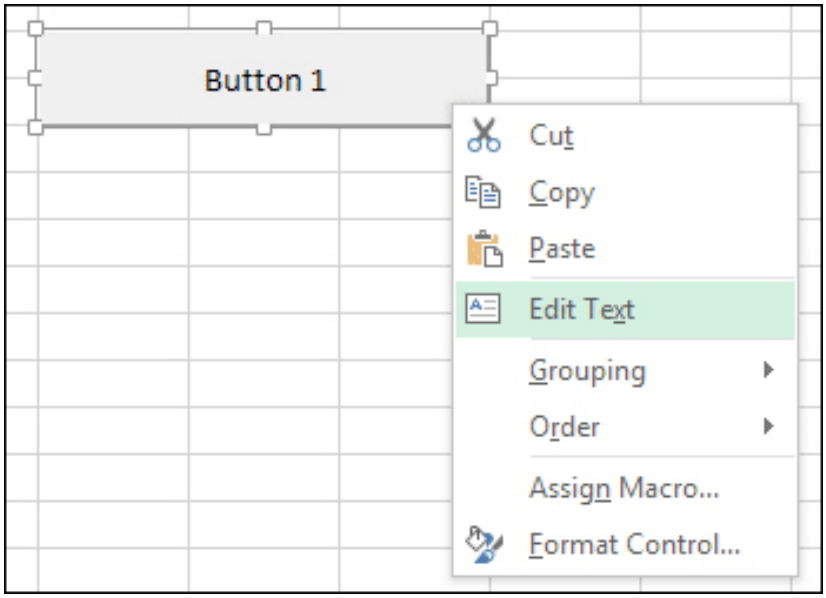


Gambar 3.3

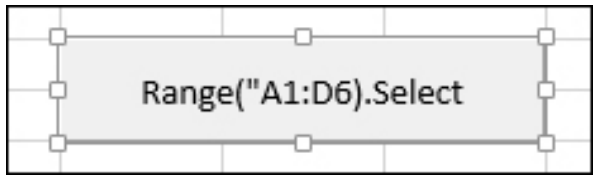
Pilih Macro Kamu dari daftar dan klik OK. Tombol pada spreadsheet Kamu akan terlihat seperti ini:



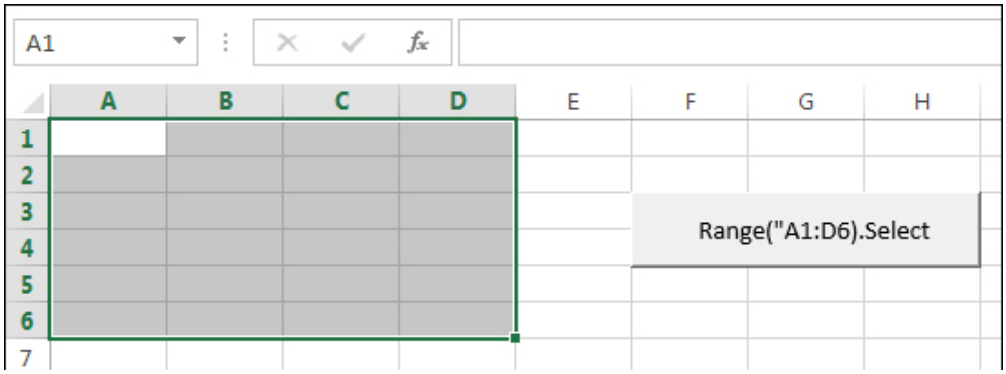
Kamu dapat mengedit teks pada tombol dengan cukup mudah. Klik tombol kanan untuk memunculkan sebuah menu. Dari menu, pilih Edit Text:



Bila Kamu pilih Edit Text, kursor akan muncul pada awal teks. Gunakan tombol panah pada keyboard untuk memindahkan kursor ke akhir baris. Hapus Button 1 dan ketik Range(“A1:D6”).



Pindahkan kursor keluar dari tombol tersebut, Kamu akan melihat garis pengaturan ukuran tombol akan menghilang. Sekarang Kamu dapat menguji tombol. Klik tombol tersebut, lalu Kamu akan melihat cell A1 ke D6 disorot:



Selamat! Sekarang Kamu telah menulis sebuah kode VBA Excel untuk memilih kisaran cell pada spreadsheet. Dan semua hanya dengan sebuah klik tombol. Sekarang kembali ke Visual Basic Editor (Dari toolbar Developer, klik Visual Basic pada panel Code.) Beri tKamu kutip pada awal kode Range Kamu, dan ini akan berubah menjadi hijau.

```
Sub Range_A1_D6()

    'Range("A1:D6").Select

End Sub
```

Mudah bukan membuat tombol pada Macro VBA Excel

3.3 Mengenal Variabel, Do dan Loop

Pembahasan mengenai variabel dan konstanta, namun karena pembahasan tentang Variable pada VBA tidak akan pernah terlepas dengan data type, akan lebih baik jika kita mengenal terlebih dahulu dengan tipe data (data type) pada VBA. Tanpa megenal data tipe akan sangat sulit sekali untuk untuk melakukan menejemen Variable dan konstanta secara efektif dan efisien.

Tujuan utama dari VBA adalah untuk memanipulasi data. Sehingga memiliki pemahaman yang baik tentang data type akan sangat membantu Kamu untuk menguasai VBA Excel.
Pada VBA, penyimpanan data di bagi dalam 2 jenis. Beberapa data di simpan dalam bentuk bentuk object dan beberapa data lainnya disimpan dalam bentuk variabel.

Objek adalah apa yang dimanipulasi oleh Visual Basic for Applications. Contoh objek adalah Workbook, Worksheet, rentang sel (range) dan sel pada excel. Sedangkan Variabel merupakan sebuah nama tertentu sebagai lokasi penyimpanan data. Variabel umumnya digunakan untuk mewakili nilai tertentu. Dengan kata lain, variabel merupakan wadah untuk sebuah nilai tertentu.

Variabel pada Visual Basic for Applications dibuat dengan melakukan sebuah deklarasi statement tertentu dengan menentukan nama dan karakteristik dari setiap variabel tersebut. Salah satu karakteristik yang bisa Kamu tentukan adalah tipe data-nya.

Tipe data merupakan sebuah cara yang digunakan untuk menentukan dan mengelompokkan jenis dari suatu data tersebut. Aktifitas ini biasa di sebut sebagai

"deklarasi variabel". Tentang Variabel akan kita bahas tersendiri pada panduan selanjutnya, kali ini kita akan fokus membahas tentang tipe data dulu.

Numeric Data Types (Tipe Data Numerik) Pada VBA Excel

Tipe data VBA yang termasuk ke dalam tipe data numerik ini adalah: Byte, Integer, Long, Single, Double, Currency, Decimal.

Berikut penjelasannya lebih lanjut.

Byte

Tipe data Byte di VBA merupakan tipe data yang hanya membutuhkan 1 byte kapasitas memori. Variabel dengan tipe data VBA-nya Byte dapat digunakan untuk menyimpan angka antara 0 dan 255.

Integer

Tipe data Integer dapat menyimpan bilangan bulat antara -32.768 sampai dengan 32.767. Variabel integer hanya membutuhkan kapasitas 2 byte memori komputer. Karena kebutuhan memori yang rendah, tipe data Integer di VBA bisa dijadikan pilihan yang paling efisien dan lebih baik untuk menyimpan bilangan bulat yang termasuk dalam jangkauannya.

Long

Tipe data Long biasa juga disebut sebagai "Long Integer". Seperti yang tersirat namanya, tipe ini dapat digunakan untuk menyimpan nilai integer yang berada dalam rentang yang lebih panjang daripada rentang tipe data Integer. Dengan menggunakan tipe data Long VBA, bisa menyimpan angka antara -2.147.483.648 dan 2.147.483.647. Jika angka ini belum cukup bisa menggunakan tipe data Double.

Single

Tipe data Single mengacu pada "single-precision floating-point", sebuah format angka yang menentukan bagaimana komputer menangani nomor tersebut.

Tipe data Single digunakan untuk menyimpan nomor dalam rentang berikut:

- Nilai negatif: -3,402823E38 sampai -1,401298E-45.
- Nilai positif: 1,401298E-45 sampai 3,402823E38.

Variabel yang menggunakan tipe data single membutuhkan 4 byte kapasitas memory komputer.

Double

Tipe data Double bisa digunakan untuk menampung bilangan bulat dan pecahan. Double berarti "Double-precision Floating-point".

Tipe Double untuk menyimpan nomor floating-point dalam rentang berikut:

- Bilangan negatif:
  -1,79769313486231E308 sampai -4,94065645841247E-324.
- Bilangan positif:
  4,94065645841247E-324 sampai 1,79769313486232E308.

Variabel yang menggunakan tipe data ini membutuhkan 8 byte memori, yang artinya 2 kali lipat kebutuhan tipe data Single dan Long Integer.

Currency

Seperti namanya tipe data Currency biasa digunakan untuk menyimpan data yang berhubungan dengan nominal uang. Meskipun pada dasarnya tidak harus nominal uang saja. Tipe data Mata Currency menghasilkan nilai skala dengan akurasi hingga 15 digit di sebelah kiri titik desimal dan 4 digit ke kanan. Sangat dianjurkan untuk menggunakan tipe data ini untuk menghindari kesalahan pembulatan saat presisi. Variabel mata uang dapat digunakan untuk menyimpan bilangan positif dan negatif. Tipe data ini disimpan sebagai angka dalam format integer yang diskalakan 10.000. Sebagai konsekuensinya, tipe data ini memungkinkan untuk rentang nilai antara -922.337.203.685.477,5808 dan 922.337.203.685.477,5807. Tipe data Currency membutuhkan 8 byte memori.

Decimal

Tipe data Decimal dapat digunakan untuk menyimpan bilangan bulat yang diukur dengan kekuatan 10. Faktor penskalaan ini bervariasi tergantung pada berapa digit yang ada di sisi kanan titik desimal. Jumlah maksimum digit yang dapat ditahan oleh variabel Desimal adalah 28.

Berapa nilai yang dapat di tampung oleh tipe data ini?

- Jika tidak mengandung nilai desimal Tipe ini dapat menampung - 79.228.162.514.264.337.593.543.950.335 sampai 79.228.162.514.264.337.593.543.950.335.
- Jika mengandung nilai desimal Tipe ini dapat digunakan untuk menampung angka -7,9228162514264337593543950335 sampai 7,9228162514264337593543950335.

Tipe data VBA Decimal memberikan jumlah digit terbesar untuk mewakili nomor tertentu. Oleh karena itu, tipe ini lebih sesuai untuk kasus di mana pengguna melakukan perhitungan dengan jumlah besar yang membutuhkan presisi dan tidak dapat menghindari kesalahan pembulatan. Tipe data VBA Desimal membutuhkan 12 byte, yang lebih besar dari tipe data numerik lainnya. Seperti yang dijelaskan oleh Microsoft, tidak dapat mendeklarasikan tipe data Decimal secara langsung. Sebenarnya, tipe Decimal merupakan sub tipe Variant. Karena itu, untuk menggunakan Desimal, harus mengaktifkan fungsi konversi CDec. Lebih singkatnya perhatikan tabel berikut ini.

Tabel Tipe Data Numerik

Tipe	Memory	Rentang Nilai
Byte	1 byte	0 sampai 255
Integer	2 bytes	-32.768 sampai 32.767
Long	4 bytes	-2.147.483.648 sampai 2.147.483.648
Single	4 bytes	• -3,402823E+38 sampai -1,401298E-45 untuk nilai negatif. • 1,401298E-45 sampai 3,402823E+38 untuk nilai positif.

Double	8 bytes	• -1,79769313486232E+308 sampai -4,94065645841247E-324 untuk nilai negatif. • 4.94065645841247E-324 sampai 1.79769313486232E+308 untuk nilai positif.
Currency	8 bytes	-922.337.203.685.477,5808 sampai 922.337.203.685.477,5807
Decimal	12 bytes	• +/- 79.228.162.514.264.337.593.543.950.335 (tanpa nilai desimal). • +/- 7,9228162514264337593543950335 (28 tempat desimal).

Tipe Data Non-Numerik (Non-Numeric Data Types)

Yang termasuk kedalam tipe data ini adalah tipe data string atau teks, tipe data Date, tipe data Boolean, tipe data Object dan tipe data Variant.

String

Pada VBA, tipe data String umumnya digunakan untuk menyimpan teks. Namun, ini tidak berarti bahwa hanya boleh menggunakan huruf dalam variabel String. Selain huruf, variabel String dapat berisi angka, spasi, tKamu baca dan karakter tertentu. Ada 2 jenis tipe data String yang bisa digunakan. Jumlah karakter dan memori yang dibutuhkan bervariasi tergantung jenisnya.

String-fixed length

Variabel yang menggunakan tipe Fixed-length String dapat berisi antara 1 dan sekitar 64.000 karakter. Variabel String ini membutuhkan jumlah memori sejumlah yang dibutuhkan oleh string atau teks itu sendiri.

String-variable length

Variabel yang menggunakan tipe Variable-length String dapat berisi apapun dari 0 sampai sekitar 2 miliar karakter. Tipe ini membutuhkan 10 byte memori ditambah memori yang dibutuhkan untuk string itu sendiri.

Date

Tipe data Date bisa digunakan untuk menyimpan nilai tanggal, waktu atau kedua-duanya (Tanggal dan waktu). Tipe ini dapat menyimpan nilai yang merepresentasikan tanggal antara 1 Januari 100 sampai 31 Desember 9999 dan atau waktu antara 00:00:00 (tengah malam) sampai 23:59:59. Kapasitas memory yang dibutuhkan oleh tipe Date ini adalah 8 Byte.

Boolean

Tipe data Boolean hanya digunakan untuk menyimpan salah satu dari 2 nilai logika TRUE atau FALSE. Data ini membuthkan 2 byte memori.

Secara umum data Boolean, TRUE dilambangkan dengan 1 dan FALSE dilambangkan dengan 0. Namun, dalam VBA, konversi antara Boolean dan tipe data VBA numerik bekerja sedikit berbeda:

- Saat mengubah variabel dengan tipe Boolean menjadi tipe data numerik, TRUE menjadi -1 dan FALSE menjadi 0.
- Saat mengubah tipe data VBA numerik menjadi Boolean, 0 menjadi FALSE dan semua nilai lainnya (terlepas dari apakah itu negatif atau positif) menjadi TRUE.

Object

Tipe data Object digunakan untuk menyimpan alamat yang mengacu pada object VBA tertentu. Tipe data ini membutuhkan 4 byte memori. Secara umum jika ingin membuat sebuah variable yang merujuk pada object VBA Excel tertentu, gunakanlah tipe data ini. Object pada VBA excel ini misalnya Workbook, worksheet, sel, Range, dan lain sebagainya.

Variant

Varian merupakan tipe data VBA default. Dengan kata lain, ini adalah jenis data yang digunakan oleh VBA saat tidak menentukan jenis datanya pada saat mendeklarasikan sebuah variabel. Dengan kata lain saat Kamu tidak menyebutkan tipe data saat melakukan deklarasi variabel maka Excel akan menggunakan tipe data yang membutuhkan 16 atau 22 kapasitas memori ini.

Tipe data ini memang membutuhkan kapasitas memori lebih besar dibanding lainnya, lebih tepatnya: Variant untuk data numeric membutuhkan 16 byte memori. Variant untuk data teks membutuhkan 22 byte memori ditambah memori yang dibutuhkan oleh string.

Tabel Tipe Data Non-Numerik

Tipe	Memory	Rentang Nilai
String (fixed length)	Sesuai panjang string	1 sampai 65.400 karakter
String (variable length)	Sesuai panjang string	+ 10 bytes0 to 2 Milyar karakter
Date	8 bytes	1 January 100 sampai 31 Desember 9999
Boolean	2 bytes	True atau False
Object	4 bytes	Untuk setiap objek yang disematkan
Variant (numeric)1	6 bytes	Nilai apapun sebesar
DoubleVariant (text)	Length + 22 bytes	Sama seperti variable-length string

Ada beberapa hal yang perlu diperhatikan sehubungan dengan pemilihan penggunaan tipe data, antara lain :

- Jika nilai data yang akan disimpan melebihi batas datatype yang ditentukan atau malah berbeda datatype-nya, maka akan menghasilkan error.
- Penulisan angka atau number pada VBA selalu menggunakan format English atau menggunakan karakter titik (.) sebagai tKamu desimal
- Penulisan date dalam VBA sebaiknya menggunakan format universal YYYY-MM-DDuntuk tanggal dan HH:mm:ss. untuk waktu
- VBA akan berusaha mengkonversi datatype yang diinputkan menjadi sesuai dengan yang dideklarasikan
- Untuk dapat menggunakan datatype Decimal, maka variabel atau prosedur jenis function harus menggunakan datatype Variant yang diisi dengan menggunakan fungsi CDec seperti yang sduah dijelaskan sebelumnya.

Mari kita lakukan ini untuk memulai:

1. Aktifkan Tombol Mode Desain, yang berada di sebelah Tombol Insert di TAB Pengembang.
2. Klik dua kali pada Tombol ActiveX.
3. Sekarang akan terlihat dan perintah untuk menulis kode untuk itu.

```
Private Sub CommandButton1_Click()

End Sub
```

Kemudian tulis beberapa kode di antara baris yang kita lihat:

```
Private Sub CommandButton1_Click()
        Dim X As Byte
        Dim Y As Byte

        X = 1
        Y = Range("B1")
  Do
                Range("A" & X) = Y

        X = X + 1
        Y = Y + Range("B1")
 Loop Until X = 101

End Sub
```

Inilah arti dari kode-kode ini.

Private Sub CommandButton1_Click()

Private berarti Kamu tidak akan menemukan kode di Tombol Macro. CommandButton1 adalah nama default yang diberikan ke Tombol ActiveX, dan Klik berarti kode di bawah ini hanya akan diterapkan ketika Kamu mengkliknya.

Dim X As Byte

Dim mendeklarasikan variabel, Jadi, dikatakan bahwa X adalah variabel yang akan digunakan dengan angka dari 0 hingga 255, karena " Byte "Hanya menerima rentang tersebut sesuai dengan tabel di atas.

X = 1

Penambahan nilai awal untuk variabel, yaitu 1.

Y = Range("B1")

Berarti nilai variabel Y nilainya sama dengan sel B1.

Do

Artinya Lakukan

Range ("A" & X) = Y

Kita sudah mendeklarasikan X sebagai variabel dengan nilai 1. Jadi, Range AX berarti Range ("A1") = Y, dan Y adalah Range ("B1"). Jika saya tambahkan angka 5 ke B1, maka Y = 5. Jadi akhirnya Range ("A1") = 5.

X = X +1

Artinya X = 1 + 1 karena pada saat itu X = 1, jadi akhirnya X = 2. Lain kali proses ini berulang, itu akan menjadi X = X + 1 lagi, tetapi kali ini X = 2, jadi akhirnya X = 3, karena X = 2 + 1, dan seterusnya.

Y = Y + Range("B1")

Ini akan mengikuti proses yang sama seperti langkah terakhir, tetapi dengan nilai yang ditambahkan dalam Range ("B1") yang akan kita tambahkan. Jika

B1 memiliki 5, maka ia akan menambahkan 5 lagi, mengikuti urutan sesuai dengan yang kita tulis di B1.

Loop Until X = 101

Artinya proses yang sama antara Do dan LOOP akan berulang hingga X = 101.

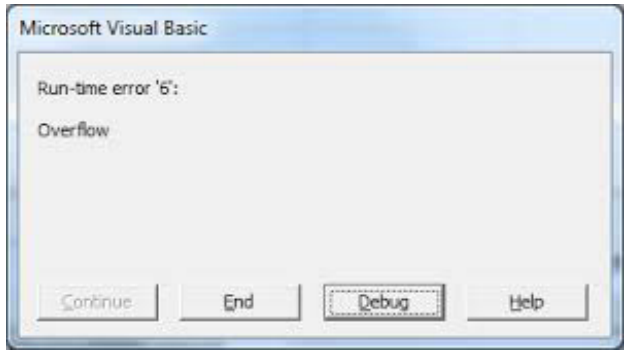
EndSub

Berarti akhir dari prosedur.

Sekarang mari kita berlatih.

Tambahkan angka 1 ke sel B1 dan tekan *Button*. Kamu akan melihat bahwa itu mengisi sel dari A1 hingga A100 satu per satu. Cobalah dengan angka 2, dan Kamu akan melihat bahwa itu menambahkan range A1: A100 angka 2 secara otomastis. Terakhir tambahkan angka 3 dan tekan tombol.

Selamat! Kamu telah menemukan *bug* (Error) pertamamu! Yaitu apa yang dinamakan *overflow.*



Bug adalah kesalahan ketika seseorang menulis kode yang salah. Dalam contoh diatas, kode berfungsi dengan baik hanya dengan angka dari 0 hingga 255, dan begitu kita meminta file untuk menuju angka 3 kali 3 seratus kali maka akan menjadi angka 300, yang lebih dari 255, hal inilah yang menyebabkan adanya *error overflow.*

Untuk memperbaikinya, Kita memiliki dua opsi, Kita meningkatkan nilai variabel atau Kita membiarkan Excel menambahkannya secara otomatis.

Pertimbangkan kemungkinan peningkatan kapasitas variabel, kita dapat mengubahnya dari byte (0 menjadi 255) menjadi integer (-32,768 menjadi 32,767). Akan lebih bagus jika kita cukup yakin bahwa tidak akan bekerja dengan jumlah yang lebih tinggi 32.767, atau bahkan mengubahnya menjadi "Long" yang dimulai dari -2.147.483.648 menjadi 2.147.483.647. Akan sangat bagus untuk bekerja bahkan dengan jutaan, tetapi mari kita bayangkan bahwa seseorang perlu menaruh miliaran di B1. Dalam hal ini bahkan variabel yang panjang tidak akan cukup. Jadi, cara terbaik untuk menyelesaikannya adalah jangan mendeklarasikan variabel dan meninggalkan kode dengan cara berikut:

```
Private Sub CommandButton1_Click()
    Dim X As Byte
      X = 1
       Y = Range("B1")
   Do
   Range("A" & X) = Y
          X = X + 1
     Y = Y + Range("B1")
   Loop Until X = 101

End Sub
```

Atau mendeklarasikan nilai sebagai Variant, yang membiarkan angka atau huruf apa saja.

```
Private Sub CommandButton1_Click()
    Dim X As Byte
    Dim Y As Variant
          X = 1
          Y = Range("B1")
   Do
          Range("A" & X) = Y
         X = X + 1
          Y = Y + Range("B1")
         Loop Until X = 101

End Sub
```

Keuntungan mendeklarasikan variabel sebagai byte, integer atau lainnya

Mungkin ada pertanyaan mengapa mendeklarasikan variabel sebagai byte atau integer, dll. Jika Excel dapat menambahkan satu secara otomatis seperti pada contoh yang telah kita lihat? Jawabannya sederhana dan penting untuk diketahui. Ingat bahwa variabel sebagai penyimpanan byte di RAM Memory hanya 1 byte, yang merupakan jumlah memori terkecil di Excel, tetapi varian yang dideklarasikan sebagai varian menyimpan setidaknya 22 byte. Jika kita tidak mendeklarasikan nilai yang kita yakin tidak akan menambahkan lebih dari nilai yang dibutuhkan, itu akan menghabiskan lebih banyak RAM, itu akan menyebabkan program berjalan lambat dalam tugas-tugas yang sangat sederhana.

Misalnya, kita perlu mendeklarasikan 1000 variabel sebagai byte. Ini hanya akan mengkonsumsi 1 Kilobyte RAM. Tetapi secara default, jika tidak memberikan tipe data, variabel tersebut dideklarasikan sebagai Varian tipe data, yang memakan 22 byte + panjang string, panjang string untuk huruf.

Jadi, 1000 variabel yang tidak dideklarasikan akan secara otomatis dideklarasikan sebagai varian, dan akan mengkonsumsi 22 kilobyte. Ini adalah 21 kilobyte kosong, yaitu 95% dari tidak tersedia ruang dan tidak digunakan sama sekali jika Kita memerlukannya sebagai byte.

Alasan utama mengapa memahami tipe data VBA itu penting adalah untuk menentukan karakteristik dari variabel yang akan dibuat di VBA. Tipe data VBA akan menentukan cara penyimpanan data di memori komputer. Masing-masing tipe data ini memiliki alokasi nominal yang berbeda pada memori yang artinya akan membutuhkan jumlah byte tertentu yang berbeda untuk setiap tipe dat

3.4 Pentingnya Require Variable Declaration

Mungkin, salah satu masalah terbesar tentang tidak mendeklarasikan Variabel adalah menciptakan bug, karena kita mengetik sesuatu yang salah. Mari buat contoh sederhananya.

1. Buka Visual Basic
2. Klik dua kali *ThisWorkbook*
3. Tambahkan kode berikut:

```
Public Sub Infinite()
myvariable = 200
    myrange = 1
Do Until myvariabe = 300
    Range("A" & myrange) = myvariable
myvariable = myvariable + 1
myrange = myrange + 1
Loop
End Sub
```

Seperti yang kita lihat, ini adalah kode yang sangat mirip dengan yang diterapkan sebelumnya. Ini harus berhenti setelah *myvariable* sama dengan 300, tetapi terus berlanjut. Lalu apa masalahnya?

Jika tidak, jalankan kodenya. Akan terlihat secara otomatis akan melampaui 300, dan tidak akan ada habisnya, atau setidaknya sampai tidak ada lagi file. Di Excel 365, total file yang tersedia adalah 1048576, jadi akan terus hingga A1048576 dan akan masuk saat ada kesalahan.

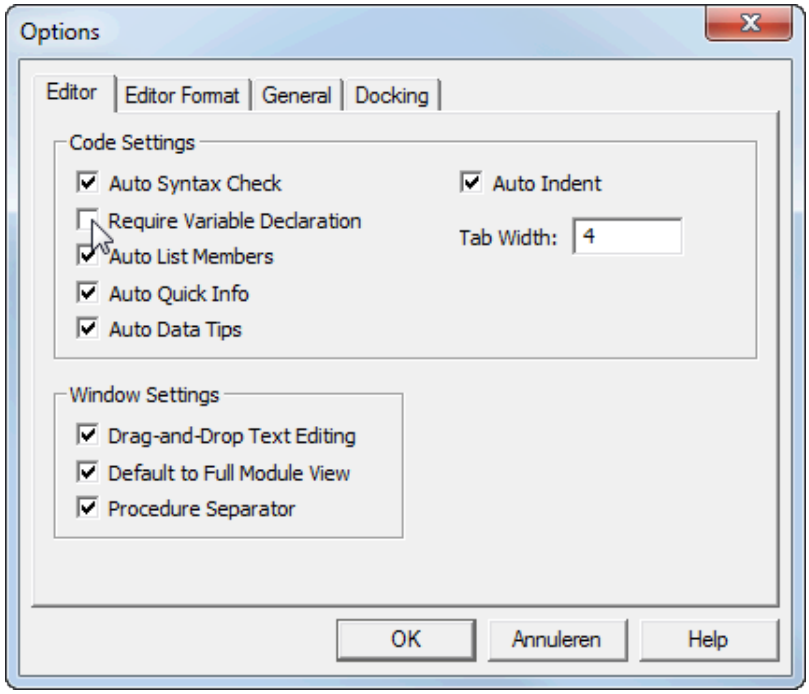
Jika tidak ingin menunggu sampai itu, Tekan ESC Key yang akan menghentikan eksekusi kode dan kemudian tekan tombol End.

Nah, masalahnya adalah kita hanya melewatkan satu huruf variabel. Kita menamakannya myvariable tetapi saat coding Kita melewatkan huruf L: Do Sampai myvariabe. Excel mengidentifikasi myvariable dan myvariabe sebagai dua variabel berbeda, dan secara otomatis menetapkannya sebagai tipe Varian. Apa yang akan terjadi jika memiliki banyak kode? Untuk menemukan masalah,ini harus melihat semua kode untuk menemukan huruf yang terlewatkan! Kita dapat mencegah kesalahan tersebut dengan menuliskan Option Explicit di baris pertama sebelum menuliskan semua macro VBA Kamu. Seperti yang kita lihat, ini akan menjadi masalah besar! Itu sebabnya Microsoft Excel memiliki opsi yang disebut *Require Variable Declaration*.

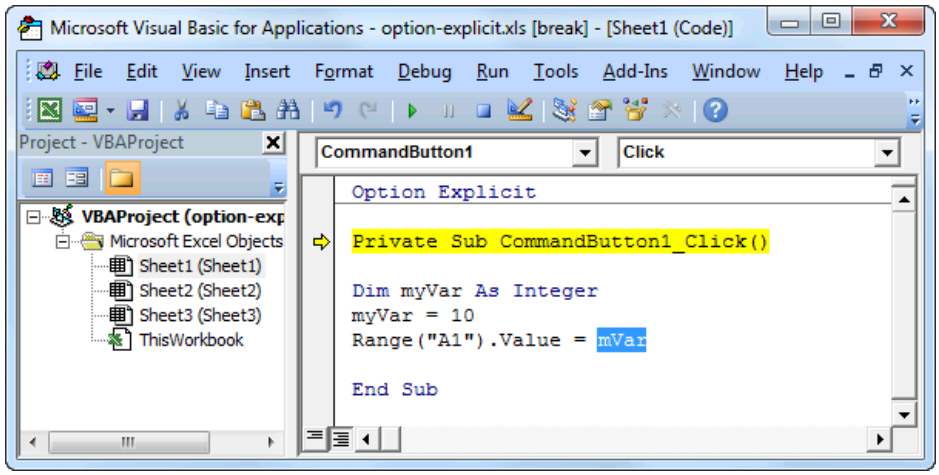
Dengan komponen ini tidak akan membiarkan menulis apa pun yang bukan kode atau variabel yang dideklarasikan.

Untuk mengaktifkan fitur ini lakukan hal berikut:

1. Dalam Visual Basic Editor, klik Tools dan kemudian klik Options.
2. Beri centang pada Require Variable Declaration.
3. Klik OK



Ini akan menambahkan sesuatu di atas kode, yang bertuliskan Option Explicit. Akan ada tampilan seperti berikut ini.



Tapi, tidak akan melihat perubahan itu sampai kita memulai proyek baru, memasukkan modul atau menulisnya sendiri. Itulah mengapa penting untuk memulai proyek baru selalu dengan opsi ini diaktifkan.

Dalam kasus ini, tambahkan Opsi Eksplisit secara manual di atas kode.

```
Option Explicit
    Public Sub Infinite()
        myvariable = 200
        myrange = 1
    Do Until myvariabe = 300
        Range("A" & myrange) = myvariable
        myvariable = myvariable + 1
        myrange = myrange + 1
    Loop
End Sub
```

Setelah menambahkan kode ini, klik Run!

Kamu akan melihat bahwa secara otomatis menampilkan pesan yang mengatakan Variable Not Defined dan bahkan menyoroti masalahnya. Jadi, kita dapat melihat bahwa ini akan menghindari masalah lebih lanjut untuk variabel yang salah eja.

3.5 Kinerja Aplikasi VBA

Sekarang, bayangkan bahwa kita benar-benar perlu mengisi satu per satu semua sel dari A1 hingga A1048576, dan itu adalah salah satu fungsi yang harus dilakukan oleh APP Kamu. Hampir selalu Developer dihubungkan tentang pembuatan Aplikasi yang berjalan cepat. Di Excel itu bukan pengecualian. Coba jalankan kode berikut dan lihat berapa lama waktu yang dibutuhkan untuk mengisi semua kolom A dengan angka:

```
Public Sub FillColumnA()
        Dim X As Long
        X = 1
Do Until Range("A1048576") <> Empty
        Range("A" & X) = X
        X = X + 1
Loop
        MsgBox "Finished"
End Sub
```

Jika sudah selesai maka akan muncul pesan bertuliskan “Finished”. Mungkin akan memakan waktu sekitar 5 menit, mungkin kurang atau lebih tergantung pada kinerja komputer yang dipakai. Kamu mungkin akan melihat Excel mengatakan bahwa itu tidak merespons, tetapi dalam banyak kasus masih berfungsi.

Kita mungkin tidak ingin menunggu terlalu lama! Cara yang baik bagi Developer VBA untuk membuat Aplikasi berjalan lebih cepat, adalah dengan menonaktifkan *ScreenUpdating*. Setiap perubahan yang dilakukan Excel seharusnya kita melihatnya. Tapi, jika kita tidak ingin melihatnya pasti akan meningkatkan kinerjanya.
Mari ubah kodenya menjadi ini:

```
Public Sub FillColumnA()
        Dim X As Long
        On Error GoTo A
        Application.ScreenUpdating = False
                X = 1
                Do Until Range("A1048576") <> Empty
                        Range("A" & X) = X
                X = X + 1
                Loop
        A:
                MsgBox "Finish"
End Sub
```

Jalankan dan berapa lama code ini bekerja, lalu bandingkan seberapa kecepatannnya. Jangan lupa untuk menghapus Kolom A sebelum memulai. Akan nampak adanya perbedaan besar! ScreenUpdating adalah alat yang hebat saat perlu meningkatkan kinerja Aplikasi. Namun, sekarang buka spreadsheet dan coba lakukan sesuatu. Kamu harus memperhatikan bahwa tidak mungkin untuk melakukan apa pun, atau setidaknya Kamu tidak akan melihatnya, karena ketika screenudating = false dalam posisi 'turned off' akan ada perubahan visibilitas. Jadi, ketika Kamu menggunakan Application.screenupdating = false, JANGAN PERNAH lupa menambahkan Application.ScreenUpdating = True di akhir kode,

dan bahkan lebih baik sebagai cadangan jika terjadi kesalahan lakukan seperti kode di bawah ini:

```
Public Sub FillColumnA()
    Dim X As Long
    On Error GoTo A
    Application.ScreenUpdating = False
        X = 1
        Do Until Range("A1048576") <> Empty
            Range("A" & X) = X
        X = X + 1
        Loop
    A:
    Application.ScreenUpdating = True
        MsgBox "Finished"
End Sub
```

3.6 MSGBOX

MsgBox di Excel merupakan salah satu fungsi yang paling sering digunakan dalam VBA Macro. fungsi MsgBox menampilkan pesan, ikon opsional, dan set yang dapat dipilih tombol perintah dalam kotak dialog. Memberikan pilihan kepada pengguna untuk mengklik sebuah tombol, dan mengembalikan sebuah Integer menunjukkan tombol yang diklik pengguna. Berikut adalah sintaks dan berbagai jenis MsgBox di VBA. Kita akan melihat pilihan yang berbeda dan penggunaan MsgBox.

MessageBox merupakan jendela yang secara umum menampilkan suatu pesan. Secara umum Visual Basic menyediakan kelas MessageBox dan MsgBox untuk membuat dialog. Sintak untuk membuat dialog adalah sebagi berikut :

MessageBox :

MessageBox.Show("Isi Pesan", "Judul Pesan", MessageBoxButtons.OK, MessageBoxIcon.Information)

Keterangan : Penulisan pada sintak diatas dimulai dari kelas Dialog (MessageBox), Isi dari dialog yang kan di tampilkan, Judul Dialog, Dialog Button (berupa OK, YesNo, Cancel dan lainnya), jenis pesan (berupa informasi, peringatan, warning dan lainnya).

MsgBox :

MsgBox("Isi Pesan", MsgBoxStyle.Information, "Judul Pesan")

Keterangan : Penulisan pada sintak diatas dimulai dari kelas Dialog (MsgBox), Isi dari dialog yang kan di tampilkan, Dialog Style (berupa pihan berupa dialog button atau jenis dialog), Judul Dialog

Tabel untuk tombol pengaturan argumen

Konstanta	Nilai	Deskripsi
vbOKOnly	0	Menampilkan tombol OK
vbOKCancel	1	Menampilkan tombol OK dan Batal
vbAbortRetryIgnore	2	Menampilkan tombol Batalkan, coba lagidan Abaikan
vbYesNoCancel	3	Menampilkan tombol ya, tidak adadan membatalkan
vbYesNo	4	Menampilkan tombol ya dan tidak ada
vbRetryCancel	5	Menampilkan tombol coba lagi dan membatalkan .

vbCritical	16	Menampilkan ikon Pesan penting
vbQuestion	32	Menampilkan ikon Peringatan kueri
vbExclamation	48	Menampilkan Pesan peringatan ikon
vbInformation	64	Menampilkan Pesan informasi ikon.
vbDefaultButton1	0	Tombol pertama adalah default.
vbDefaultButton2	256	Tombol kedua adalah default.
vbDefaultButton3	512	Tombol ketiga adalah default.
vbDefaultButton4	768	Tombol keempat adalah default.
vbApplicationModal	0	Aplikasi modal; pengguna harus merespons kotak pesan sebelum melanjutkan bekerja dalam aplikasi saat ini.
vbSystemModal	4096	Sistem modal; Semua aplikasi ditangguhkan hingga pengguna merespon kotak pesan.
vbMsgBoxHelpButton	16384	Menambahkan tombol bantuan ke kotak pesan
VbMsgBoxSetForeground	65536	Menentukan jendela kotak pesan sebagai jendela latar depan
vbMsgBoxRight	524288	Teks rata kanan
vbMsgBoxRtlReading	1048576	Menentukan teks akan muncul sebagai kanan-ke-kiri baca pada sistem Ibrani dan Arab

Catatan: Di Access 2010, Pembuat Ekspresi memiliki IntelliSense, sehingga dapat melihat argumen apa yang diperlukan oleh persamaan Kamu.

Keterangan:

Kelompok nilai pertama (0–5) menjelaskan jumlah dan jenis tombol yang ditampilkan di kotak dialog; kelompok kedua (16, 32, 48, 64) menggambarkan gaya ikon; grup ketiga (0, 256, 512) menentukan tombol mana yang merupakan default; dan kelompok keempat (0, 4096) menentukan modalitas kotak pesan. Saat menambahkan angka untuk membuat nilai akhir untuk argumen tombol , gunakan hanya satu angka dari setiap grup.

Tabel Nilai yang dikembalikan

Konstanta	Nilai	Deskripsi
vbOK	1	Oke
vbCancel	2	Batalkan
vbAbort	3	Batalkan
vbRetry	4	Coba lagi
vbIgnore	5	Abaikan
vbYes	6	Ya
vbNo	7	Tidak

Seperti yang terlihat pada kode di atas dimana menampilkan pesan yang mengatakan "Finished". Kita bisa melakukan lebih dari itu!

Itu adalah jenis pesan yang dapat ditampilkan, dan setelah mengkliknya, itu akan mengembalikan nilai sesuai dengan tabel di atas.
Jalankan kode berikut:

```
Public Sub fff()
Dim X As Byte
X = MsgBox("This is the body",
vbYesNoCancel, "This is the title")
MsgBox X
End Sub
```

Ini akan menampilkan sebuah msgbox (Kotak Pesan) dengan tombol Ya, Tidak dan Batal. Setelah Kamu mengklik tombol apa pun, X akan mengambil nilai itu dan akan menampilkannya di MsgBox berikutnya. Jadi, jika Kamu mengeklik Ya, akan terlihat angka 6, dan jika mengeklik Tidak, akan terlihat angka 7.
Ini memberikan manfaat yang besar, karena akan ada pemberitahuan apakah pengguna ingin melanjutkan atau tidak, dll.

3.7 If dan Select Case

If dan Select Case adalah sebuah pernyataan bersyarat. If dan Select Case adalah merupakan suatu perintah untuk mengambil suatu keputusan terhadap sebuah atau beberapa kondisi. Kondisi adalah ungkapan untuk mengevaluasi sebuah pernyataan apakah bernilai benar atau salah.

Select Case biasa dipakai untuk alternatif Bentuk If..Then..Else, karena kodenya lebih mudah dibaca. Perintah SELECT CASE adalah salah satu syntax yang digunakan dalam bahasa pemrograman Visual Basic for Applications (VBA). Perintah atau syntax ini sebenarnya memiliki peran yang hampir sama dengan perintah If-Then, namun dalam beberapa kasus sedikit berbeda atau kadang tergantung selera dari programmer lebih memilih menggunakan Select Case atau perintah If-Then. Mari kita gabungkan satu dalam kode di bawah ini, sehingga kita dapat mengetahui bagaimana melakukannya:

```
Sub Public myMessageBox ()
Dim X Sebagai Byte
        X = MsgBox ("Saya akan memberi tahu Kamu tombol yang
    Kamu tekan", vbYesNoCancel + vbExclamation,
          "Tombol apa yang akan Kamu tekan?")
        IfX = 2 Then
          MsgBox "Kamu Menekan Tombol Batal"
        ElseIf X= 6 Lalu
         MsgBox "Kamu menekan Tombol Ya" Lain
         MsgBox "Kamu menekan Tombol Tidak"

End If

End Sub
```

Ini akan memberi tahu tombol yang kamu klik. Artinya jika dapat mengetahui tombol apa yang ditekan maka dapat mengontrol tindakan yang sesuai dengannya. Sebagai Contoh, jalankan Macro.

Jika saja situasi seperti berikut ini

```
Public Sub Evaluation()
Dim X As Byte

X = Range("A1")
If X = 5 Then
MsgBox "There's a number " & X
   ElseIf X = 6 Then
      MsgBox "There's a number " & X
   ElseIf X = 7 Then
      MsgBox "There's a number " & X
   ElseIf X = 8 Then
      MsgBox "There's a number " & X
   Else
      MsgBox "There's another value"
   End If
End Sub
```

Jadi "if" membuat kondisional, "elseif" dicentang jika kondisi pertama tidak memenuhi, dan akan memeriksa sebanyak "elseif" yang ditambahkan hingga satu kondisi memenuhi. "Else" akan dijalankan jika ada kondisi yang tidak terpenuhi. Jadi, jika kita tidak menambahkan angka dari 5 sampai 8, kita akan melihat pesan yang mengatakan "*There's another value*", tetapi jika kita memasukkan angka 8, maka kita akan melihat pesan yang mengatakan Ada angka 8.

Jika, elseif dan lain sangat mirip dengan jenis kode lain yang disebut *Select Case*, dan *Case Else*. Namun, yang ini lebih efisien daripada 'IF' dalam beberapa kasus.

Mari lakukan hal yang sama dari di atas tetapi gunakan case sebagai gantinya:

```
Public Sub Cases()
Dim X As Byte
X = Range("A1")
  Select Case Range("A1")
   Case 5 To 8
        MsgBox "There's a number " & X
   Case Else
        MsgBox "There's another value"
   End Select
End Sub
```

Yang terakhir ini membutuhkan lebih sedikit kode. Lebih mudah dan lebih baik. Kita dapat melakukan hal yang sama tetapi terserah mana yang ingin digunakan.

BAB 4
Membuat Penghitungan dengan ActiveX

Kita akan membuat kalkulator sederhana terlebih dahulu. Setelah proyek ini selesai, Kita akan membuat yang lebih canggih. Untuk melakukan itu, kita perlu mengetahui hal-hal berikut:

4.1 Pengertian Modul

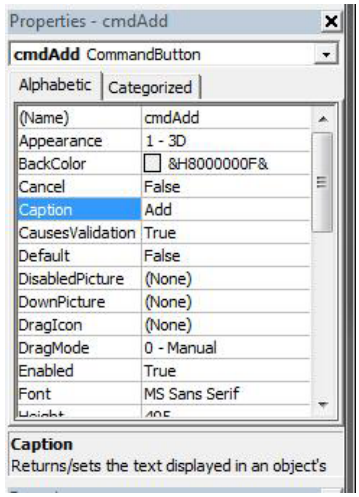
Dalam pengertian yang sederhana modul dapat diartikan sebagai tempat untuk menyimpan kode-kode VBA. Jika VBE merupakan sebuah rumah Kamu dapat membayangkan kamar-kamar atau ruangan pada rumah tersebut sebagai sebuah modul, dimana pada masing-masing kamar atau ruangan tersebut tersimpan berbagai macam barang lain dengan fungsi-fungsi tertentu.

Saat memulai menulis kode VBA, biasanya akan mulai menulis di Sheet1. Tapi, untuk memahami Modul, Prosedur Private dan Umum, kita akan membuat kalkulator visual. Pertama, kita akan melakukannya di spreadsheet, lalu akan melakukannya sebagai program nyata.

4.2 Cara membuat Modul

Pertama, tambahkan tombol ActiveX berikut pada spreadsheet di atas tabel seperti yang ditunjukkan pada gambar. Untuk menambahkan simbol yang sesuai ke setiap tombol, tidak seperti yang Kita lakukan pada Tombol Formulir. Dalam hal ini kita harus mengklik Design More Button, yang ada di TAB Pengembang, lalu klik kanan pada setiap tombol dan Kamu akan melihat sesuatu seperti gambar di bawah ini, di mana kita melihat sesuatu yang disebut "Nama". Kebanyakan orang akan berpikir bahwa ada tempat di mana kita dapat menambahkan simbol yang

benar untuk dilihat, tetapi itu salah. Excel mengidentifikasi Tombol dengan nama itu. Bayangkan Kamu memiliki beberapa tombol dengan + sebagai namanya. Bagaimana cara Excel atau Kamu mengidentifikasi mana yang benar? Jadi, setiap tombol akan memiliki nama yang berbeda, mari tambahkan yang bagus yang memberi tahu kita seperti apa. Dalam contoh saya menulis cmdAddition. Ini memberi tahu saya bahwa itu berlaku.

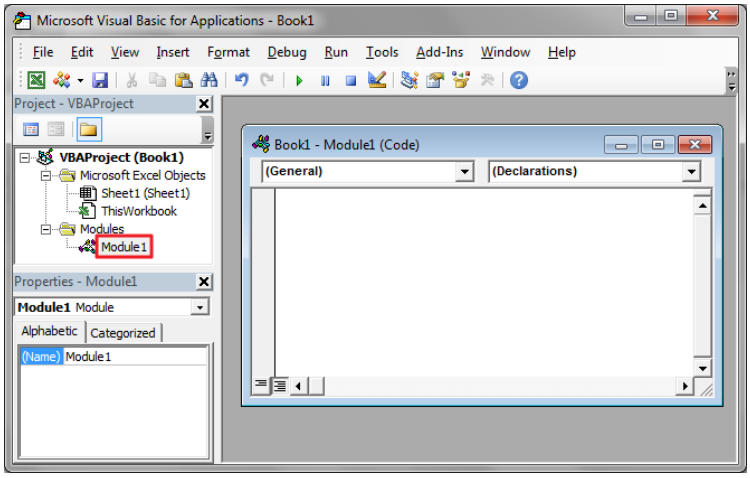


ke Tombol “+”. Namun, bagaimana cara menambahkan tKamu +?

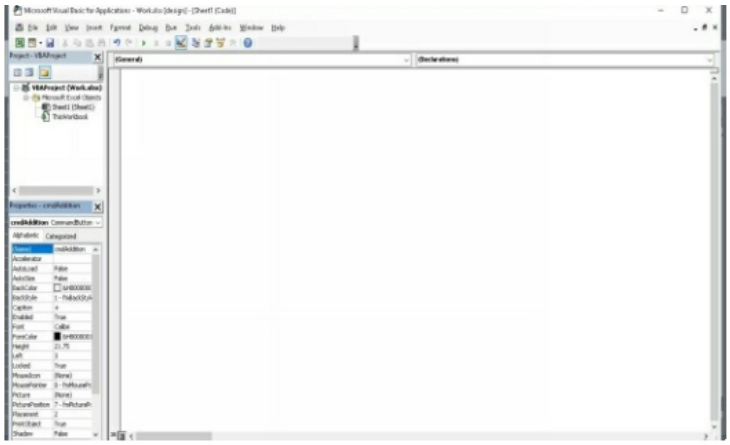
Pada gambar di atas, kamu akan melihat bahwa ada properti bernama Caption. Di situlah kita akan menambahkan tKamu "+".
Dapat juga mencoba berkreasi sedikit dengan opsi lain, seperti Backcolor, Font, Height dan bahkan menambahkan gambar.
Kita dapat meninggalkan semua opsi sebagaimana adanya dan biarkan secara default.

Setelah mengulangi proses untuk semua tombol, mari kita mulai dengan langkah selanjutnya. Mulai membuat buat Modul!
Ikuti langkah ini:

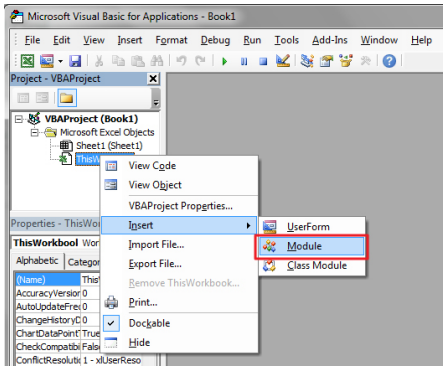
1. Buka Visual Basic melalui TAB Pengembang. Secara default Kamu akan melihat sesuatu seperti ini:



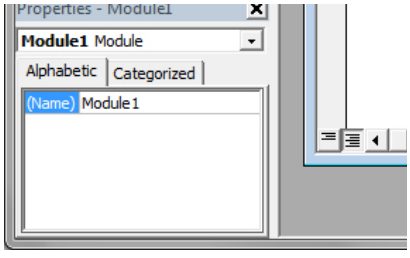
2. Klik "View TAB" dan Select "Properties Window". Kita akan selalu sering menggunakan jendela ini. Sekarang Kamu akan melihat sesuatu seperti yang ditampilkan di dalam kotak merah:



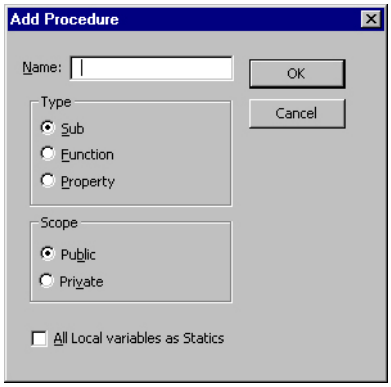
3. Mari tambahkan Modul pertama kita. Klik Sisipkan TAB, lalu Modul.



4. Kamu akan melihat Folder Baru yang Disebut "Modul" dengan file di sana Disebut "Modul 1":



5. Select dan di Properties Window Ubah namanya menjadi Addition, kemudian ulangi prosesnya sampai Kamu telah membuat Module untuk Minus, Division dan Times.
6. Klik dua kali pada Module “Addition” untuk membukanya.
7. Kamu akan melihat bahwa ini menampilkan seluruh lembar kosong. Kita perlu menambahkan Prosedur! Untuk melakukan itu, klik Sisipkan TAB, lalu Prosedur. Itu akan tampilkan jendela seperti ini, tambahkan nama dan biarkan opsi default:



8. Klik OK dan Kamu akan melihat ini:

9. Di antara baris-baris itu tulis kode berikut:

   Range("A2") + Range"B2") = Range("C2")

10. Sekarang jalankan dengan mengklik segitiga hijau di atas.
11. Ini menampilkan kesalahan yang mengatakan: Invalid Use atau Property. Mungkin karena belum menambahkan angka apa pun di sel A2 dan B2. Tambahkan dan jalankan lagi.

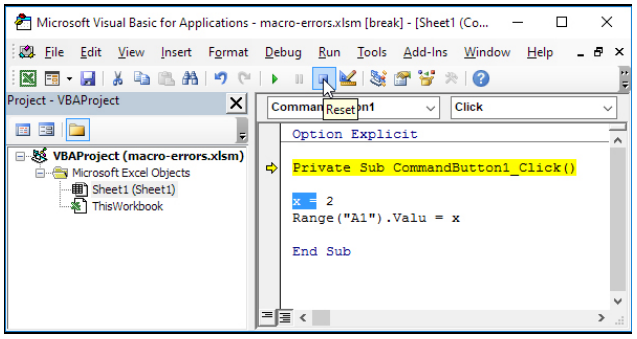
Masih ada masalah! Bisakah kamu melihat apa yang salah?
Selamat datang di bug kedua Kamu!

Masalahnya di sini sangat biasa. Kita meminta Excel bahwa sel A2 + B2 sama dengan C2, bukan C2 sama dengan A2 + B2. Masalah ini hanyalah masalah perintah. Sebaiknya jangan lupakan aturan ini !! Selalu tambahkan terlebih dahulu sel yang ingin Kamu ubah, lalu tambahkan nilai yang Kamu perlukan. Seperti ini:

Rentang ("C2") = Rentang ("A2") + Rentang ("B2")

Jalankan lagi dengan segitiga hijau yang sama. Dan ... Berhasil!

Jika melihat garis kuning yang tidak memungkinkan untuk menjalankannya, cukup klik stop, perbaiki kodenya, dan jalankan lagi.



Sekarang Kamu akan melihat bahwa di sel C2 kita menemukan hasil A2 + B2. Mari selesaikan modul lain dengan mengulangi proses yang benar di atas. Tambahkan kode-kode ini untuk melakukannya:

Modul Minus:

```
Public Sub Minus()
   Range("C2") = Range("A2")  - Range("B2")
End Sub
```

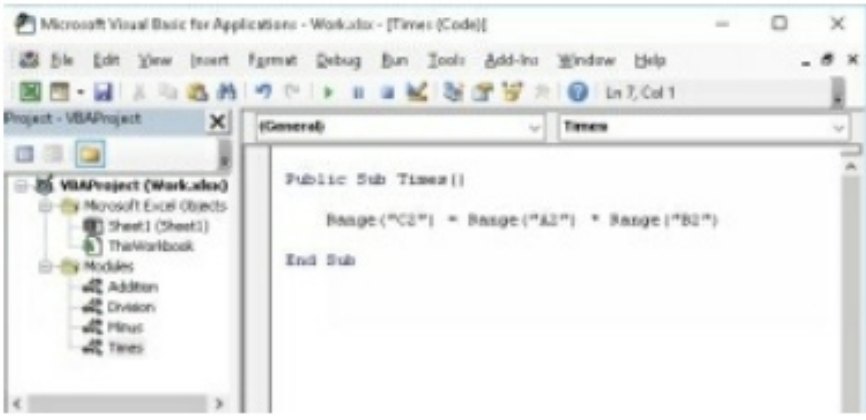
Modul Divisi:

```
Public Sub Division()
Range("C2") = Range("A2") / Range("B2")
End Sub
```

Modul Waktu:

```
Public Sub Times()
   Range("C2") = Range("A2") * Range("B2")
End Sub
```

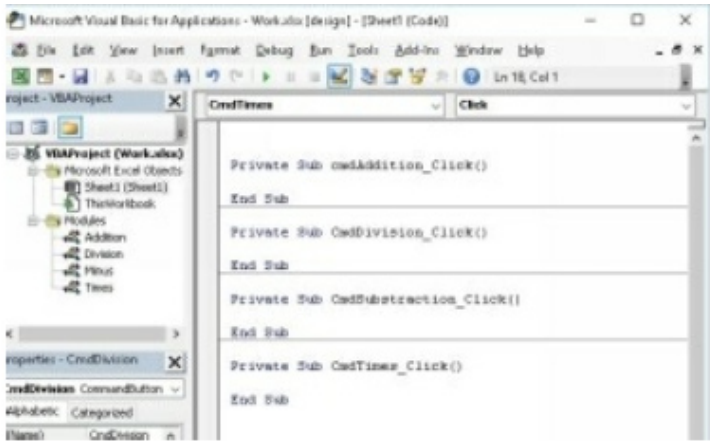
Kamu seharusnya melihat file seperti ini:



Semua modul harus berfungsi jika Kamu menjalankannya.
Sekarang, mari tautkan Modul ke tombol yang sesuai.

Untuk melakukan itu, lakukan hal berikut:

1. Buka spreadsheet
2. Klik Mode Desain
3. Klik dua kali setiap tombol. Kamu akan melihat bahwa setiap kali Kamu melakukannya, itu menambahkan beberapa kode ke Sheet1 (Sheet1). Akhirnya, akan terlihat seperti ini:

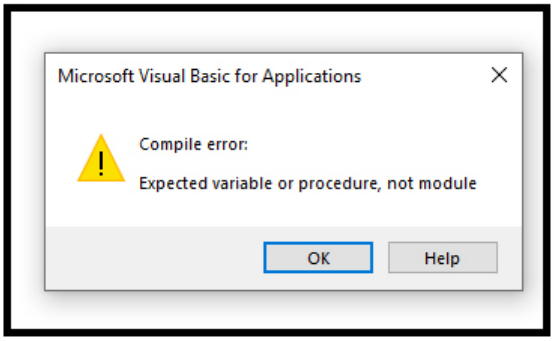


Sekarang, mari lakukan proses yang disebut "Calling". Untuk melakukan itu hanya akan menulis setiap Modul di antara baris yang sesuai:

```
Private Sub cmdAddition_Click()
Addition
End Sub

Private Sub CmdDivision_Click()
Division
End Sub
```

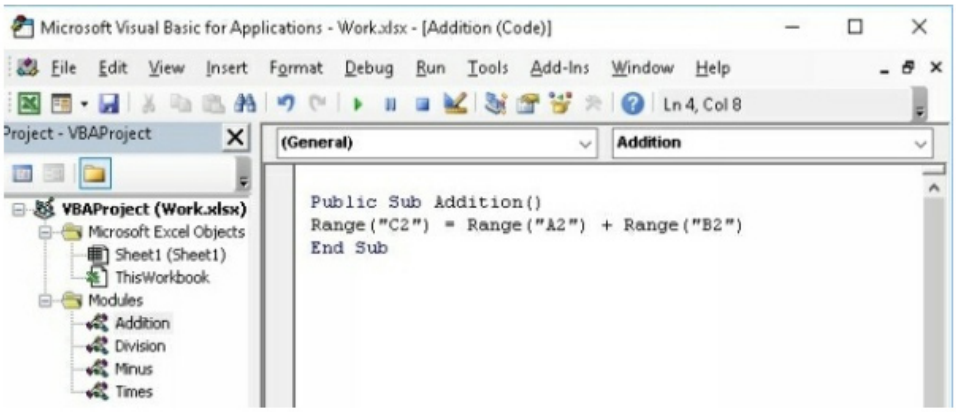
```
Private Sub CmdSubstraction_Click()
Minus
End Sub

Private Sub CmdTimes_Click()
Times
End Sub
```

Setelah Kamu menulis kode di atas, coba tekan sebuah tombol:



Selamat datang di bug ketiga Kamu !!
Mengapa ini terjadi? Ayo baca pesannya.

Ia mengatakan bahwa itu tidak mengharapkan *Variabel atau Prosedur. bukan Modul*, Masalah dalam bug ini adalah Modul dan Prosedur dalam contoh ini disebut sama dan itu adalah kesalahan besar. Tidak ada yang harus memiliki nama yang sama saat memprogram! Gambar 4.1



Gambar 4.1

Mari kita perbaiki dengan cepat. Tambahkan saja secara manual kali ini nomor 1 setelah setiap sub prosedur di setiap modul.

```
Public Sub Addition1()
Range("C2") = Range("A2") + Range("B2")

End Sub
```

Kamu juga harus mengubah kode tertulis pada Sheet1. Cukup tambahkan yang baru saja Kita tambahkan ke setiap sub prosedur di modul:

```
Private Sub cmdAddition_Click()
Addition1
End Sub

Private Sub CmdDivision_Click()
Division1
End Sub
```

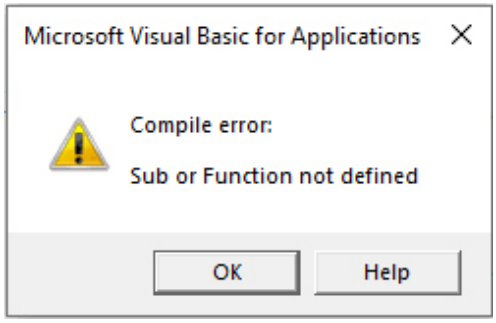
```
Private Sub CmdSubstraction_Click()
Minus1
End Sub

Private Sub CmdTimes_Click()
Times1
End Sub
```

Coba sekarang dengan menekan setiap tombol dan lihat apakah berhasil! Bagus!!

Kita diharapkan memiliki gagasan yang jauh lebih baik tentang Modul dan Prosedur. Kita akan selalu menggunakannya. Namun, mungkin Kamu memperhatikan bahwa beberapa di antaranya disebut Public dan yang lainnya disebut Private. Apa artinya?

PUBLIC berarti bahwa prosedur dapat dipanggil dari mana saja. Kamu bahkan akan melihat bahwa semuanya bahkan ada di daftar Macro, tetapi yang tidak ada yang mengatakan PRIVATÉ.

Coba artinya dengan mengubah hanya satu atau dua Modul, ubah kata Publlic menjadi Private, dan coba jalankan.



Karena itu adalah Private, tidak dapat menemukannya. Mari kita ubah kata Private kembali menjadi Public, Kamu akan melihat bahwa itu akan berfungsi.

Jadi, Private tidak dapat dipanggil, yang Public dapat dipanggil, bahkan dari daftar Macro. Dengan kata lain, Prajurit tidak dapat ditautkan, tetapi Public dapat.
Kita telah melihat fungsi paling dasar dari VBA. Kamu seharusnya sudah memiliki ide bagus tentang VBA sekarang. Namun, mari kita lakukan sesuatu yang jauh lebih profesional: Kalkulator sungguhan.

4.3 Menambahkan Huruf

Sebelum kita memulai kalkulator lain, lakukan percobaan ini yang akan menarik untuk Kamu ketahui, dan pada saat yang sama, sangat berguna. Di kalkulator terakhir yang Kita buat, coba tambahkan dua huruf, bukan angka, dan Tambahkan.

Kita akan melihat bahwa kita dapat menambahkan huruf !! A + A = AA !! ?? Hal ini tidak akan berhasil jika melakukan pengurangan, pembagian, atau perkalian.

Itu terjadi karena menggunakan "+" di VBA tidak sama dengan rumus = SUM (). Mari kita ingat sekarang, karena ini akan sangat berguna untuk bahasan selanjutnya.

BAB 5
Penghitungan Menggunakan Forms

Ulasan

Kali ini kita akan menggunakan lebih banyak lagi fungsi Visual Basic for Applications.
Sekarang kita mengerti mengapa ini disebut Visual Basic.

Pertama-tama mari kita tinjau beberapa hal penting untuk dipertimbangkan:

1. Piulihkah Variabel yang benar untuk menghindari bug dan membuat program berjalan lancar.
2. Tulis kode dalam urutan yang benar, jika tidak maka akan menghasilkan bug.
3. Jangan pernah mengulang nama apapun dalam sebuah program. Semuanya harus diidentifikasi dengan nama yang unik.
4. Perhatikan kata-kata yang bertnada warna biru. Mereka adalah kode bawaan yang digunakan oleh Excel VBA. Jangan menghapusnya atau mengubahnya tanpa sepengetahuan, atau itu akan menghasilkan kesalahan.
5. TIPS: Selalu tambahkan nama menggunakan huruf kapital, dan ketika memanggil mereka tulis nama mereka menggunakan huruf kecil. Kamu akan melihat bahwa dengan menggunakan cara ini akan menghindari kesalahan pengetikan, karena jika menulisnya dengan benar maka huruf kapital otomatis akan ditambahkan, jika tidak maka akan tetap sama.
6. Coba terus-menerus kode baru yang kamu tulis, jadi akan yakin bahwa kode tersebut berfungsi dengan baik.
7. Tambahkan komentar yang cukup agar tahu apa yang dilakukan kode jika diperlukan untuk meninjau sesuatu di masa mendatang. Kita akan belajar bagaimana melakukan itu.

5.1 FORMS

Visual Basic for Applications di Excel memiliki fitur yang sangat menarik bernama User*Forms.*

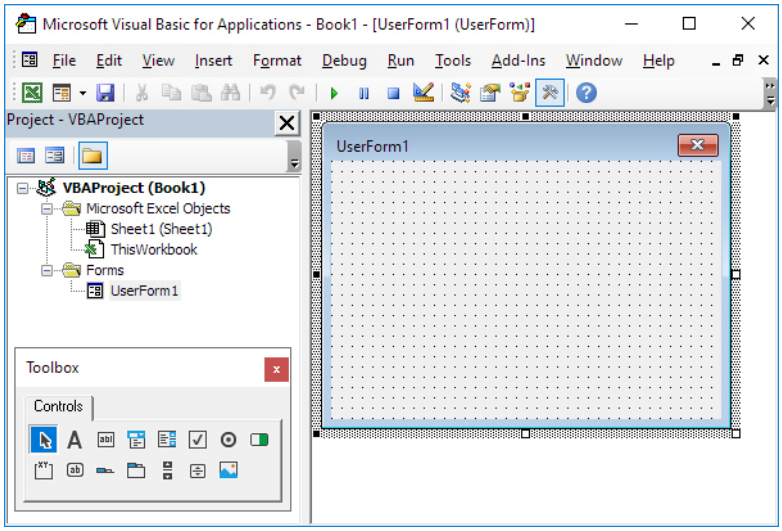
Bentuk-bentuk aplikasi visual yang bisa kita buat seperti yang kita lihat di awal buku ini.

Begitulah cara Kita membuat kalkulator.

Ikuti langkah ini:

1. Buka Dokumen Excel Kosong Baru.
2. Buka Visual Basic.
3. Klik Sisipkan TAB.
4. Bentuk pengguna.

Kamu akan melihat sesuatu seperti ini:



Ini yang dinamakan *userform*. Kali ini akan digunakan untuk membuat Kalkulator. Pertama, Kamu melihat Kotak yang disebut "Controls" dengan beberapa opsi di dalamnya. Kita akan menggunakannya hampir sepanjang waktu, jadi jangan

tutup. Jika terlanjur melakukannya, dapat membukanya lagi dengan mengklik Lihat TAB, lalu pada Toolbox.

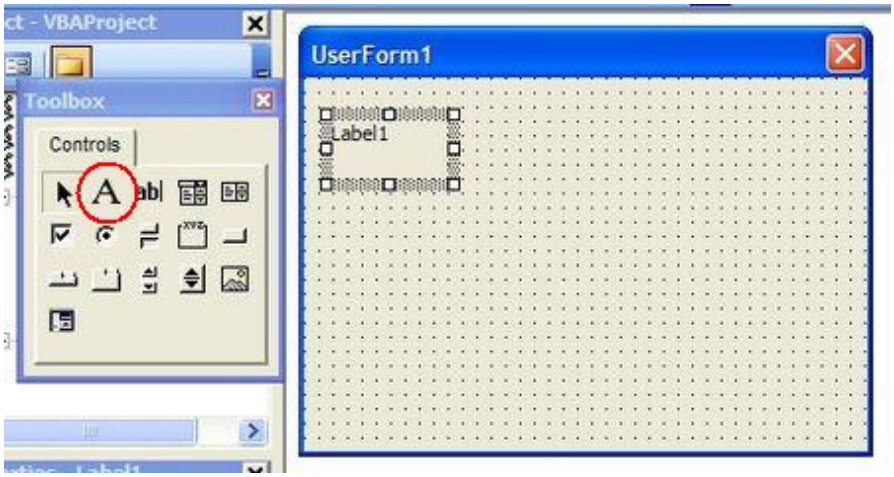
Jangan tutup juga properti *Userform.* Ini akan menjadi alat utama yang akan digunakan untuk proyek ini, dan Dalam kasus ini, bahkan tidak akan menyentuh Spreadsheed pada Excel.
Di jendela properti, ubah nilai-nilai ini:

Name: CalculatorProject
Caption: Calculator

Height: 260
Width: 200

Ingat bahwa Caption adalah Judul yang Ditampilkan.
Di toolbox pilih label, atau icon A di sebelah Panah.

Setelah dipilih, klik di sisi atas formulir, tahan dan seret untuk membuat persegi panjang yang akan menjadi layar untuk kalkulator.



Sekarang, Select label, dan ubah namanya di properti. Beri nama Display. Kemudian, hapus nilai teks.

Di ToolBox, Select CommandButton, lalu tambahkan rangkaian seperti berikut:

Setiap kali menambahkan tombol baru, pastikan untuk menambahkannya dengan ukuran berikut

Tinggi: 30

Lebar: 36

TIPS: Buat satu saja, lalu salin dan tempel semua tombol yang Kamu butuhkan. Tempatkan mereka dalam urutan yang benar, lalu pastikan untuk menambahkan setiap tombol dengan info berikut:

Nama	Caption
CmdDel	Del
CmdCE	CE
CmdPercent	%
CmdTimes	*
CmdDivision	/
CmdAdd	+
CmdMinus	=
CmdEquals	1
Cmd1	2
Cmd2	3
Cmd3	4
Cmd4	5
Cmd5	6
Cmd6	7
Cmd7	8
Cmd8	9
Cmd9	.
mdDot	0
Cmd0	00
Cmd00	

5.2 Kode Commmand Buttons

Setelah memiliki desain kalkulator dan namanya dengan benar, Kita dapat menjalankannya dan mengklik setiap tombol. Kamu akan melihat bahwa itu tidak melakukan apa-apa, karena belum memberi tahu Excel apa yang harus dilakukan dengannya. Excel tidak tahu apakah saya menginginkan formulir itu untuk lelucon atau hal lainnya. Jadi, mari tambahkan kode ke setiap tombol.

1. Klik dua kali ke tombol dengan angka 1.
2. Ini menambahkan secara otomatis kode ini:
   Private Sub cmd1_Click ()
   Akhiri Sub
3. Tambahkan kode ini di antara baris di atas:
   Display.Caption = cmd1.Caption
4. Jalankan *form*.
5. Kamu akan melihat nomor satu tertulis di Layar !! Tapi, itu tidak menambahkan lebih dari satu nomor *1 !!* Jika itu akan menjadi kalkulator, kita perlu menambahkan beberapa jika perlu! Apa yang salah??

Masalahnya di sini adalah bahwa kode tersebut mengatakan *Caption of the Display* (yang Kita biarkan kosong), sama dengan caption dari cmd1, yang merupakan angka 1 dan persis seperti yang dilakukannya.

Jadi, untuk mengatasi masalah ini kita harus memberitahunya untuk menambahkan angka 1 setelah angka 1, dan seterusnya.

Display.Caption = Display.Caption + cmd1.Caption

Mungkin Kamu mengerti sekarang. Persis seperti itulah yang Kita lakukan hampir di awal buku ini. Apakah Kamu ingat ketika kita membuat sel meningkatkan nilainya sesuai dengan angka yang tertulis di sel B1?

Jika ingin memeriksanya untuk meninjau periksa tentang Variabel, Do dan Loop. Di halaman sebelumnya.

Artinya X = X + 1 atau dengan kata lain Display.caption akan menambah nilainya sendiri ditambah caption dari cmd1. Jadi, 1 = 1 + 1 = 11 Ini bukan matematika, namun, apakah Kamu ingat ketika menambahkan huruf ke proyek kalkulator terakhir? Menurut kalkulator itu A + A = AA, jadi di sini 1 + 1 = 11. Untuk mencoba membuatnya lebih jelas, jika Kamu mengetik tombol satu itu akan menjadi 1 = 1 = 1 maka anggaplah Kamu mengetikkan angka dua. Ini akan menjadi 1 = 1+2 = 12, sekarang Kamu mengetik angka 7:12 = 12 + 7 = 127 dan terakhir Kamu mengetik angka 9: 127 = 127 +9 = 1279 dan seterusnya.

Jalankan dengan kode baru. Ini seharusnya bekerja sekarang! Kamu harus memiliki semua kode ini:

```
Private Sub cmd1_Click()
Display.Caption = Display.Caption + cmd1.Caption
End Sub
```

Mari ulangi proses ke semua angka dan simbol, dengan mengklik dua kali pada setiap tombol dan setelah menambahkan beberapa kode secara otomatis, tulis baris kode yang benar di antara mereka, lihat kode berikut sehingga mungkin memiliki ide yang lebih baik:

```
Sub Private Cmd0_Click ()
Display.Caption = Display.Caption + Cmd0.Caption End Sub

Private Sub Cmd0_Click()
Display.Caption = Display.Caption + Cmd0.Caption
End Sub
```

```
Private Sub Cmd00_Click()
Display.Caption = Display.Caption + Cmd00.Caption
End Sub

Private Sub cmd1_Click()
Display.Caption = Display.Caption + cmd1.Caption
End Sub

Private Sub Cmd2_Click()
Display.Caption = Display.Caption + Cmd2.Caption
End Sub

Private Sub Cmd3_Click()
Display.Caption = Display.Caption + Cmd3.Caption
End Sub

Private Sub Cmd4_Click()
Display.Caption = Display.Caption + Cmd4.Caption
End Sub

Private Sub Cmd5_Click()
Display.Caption = Display.Caption + Cmd5.Caption
End Sub

Private Sub Cmd6_Click()
Display.Caption = Display.Caption + Cmd6.Caption
End Sub

Private Sub Cmd7_Click()
Display.Caption = Display.Caption + Cmd7.Caption
End Sub
```

```
Private Sub Cmd8_Click()
Display.Caption = Display.Caption + Cmd8.Caption
End Sub

Private Sub Cmd9_Click()
Display.Caption = Display.Caption + Cmd9.Caption
End Sub

Private Sub CmdAdd_Click()
Display.Caption = Display.Caption + CmdAdd.Caption
End Sub

Private Sub CmdDivision_Click()
Display.Caption = Display.Caption + CmdDivision.Caption
End Sub

Private Sub CmdDot_Click()
Display.Caption = Display.Caption + CmdDot.Caption
End Sub

Private Sub CmdMinus_Click()
Display.Caption = Display.Caption + CmdMinus.Caption
End Sub

Private Sub CmdPercent_Click()
Display.Caption = Display.Caption + CmdPercent.Caption
End Sub

Private Sub CmdTimes_Click()
Display.Caption = Display.Caption + CmdTimes.Caption
End Sub
```

Semua kode ini harus memungkinkan kita melihat setiap angka dan simbol di layar. Tombol 'sama dengan', Del dan CE adalah yang tidak kita lihat tertulis pada tampilan kita, dan akan memberi perlakuan berbeda.

Tombol Pertama CE:

Ini harus membersihkan atau menghapus semua layar. Yang ini sangat mudah, ini adalah
kode:

```
Private, Sub CmdCE_Click()
Display.Caption = Empty
End Sub
```

Artinya persis seperti yang dijelaskan pada kode yang sama. Tampilan akan dibiarkan kosong.

Kedua, Tombol Del:

Ini harus menghapus satu per satu, dari nomor terakhir yang telah Kita tambahkan. Bagaimana Kamu melakukannya? Mungkin, jika Kamu sedang belajar VBA Kamu sudah tahu Rumus Excel. Jadi, cobalah mencari cara untuk melakukan itu.
Jika Kamu kesulitan dengan itu, gunakan jalan pintas berikut ini:

Value (Cells as A1 Below)	Formula applied	Result
12345678	=left(A1,len(A1))	12345678
12345678	=left(A1,len(A1)-1)	1234567
12345678	=left(A1,len(A1)-2)	123456

=left () Menampilkan nilai dari kiri, sesuai dengan jumlah huruf yang Kita perintahkan.

=len () menghitung jumlah huruf yang ada di sel.

- 1 Ini akan mengurangi satu dari total huruf atau angka len ().

Hasilnya ditampilkan di atas. Jelas, jadi apa yang terjadi jika kita mengulangi proses tersebut beberapa kali diterapkan pada sel yang sama?

Repetitions	Value (Cells as A1 Below)	Formula applied	Result
1	12345678	=left(A1,len(A1))	12345678
2	1234567	=left(A1,len(A1))	123456
3	123456	=left(A1,len(A1))	12345

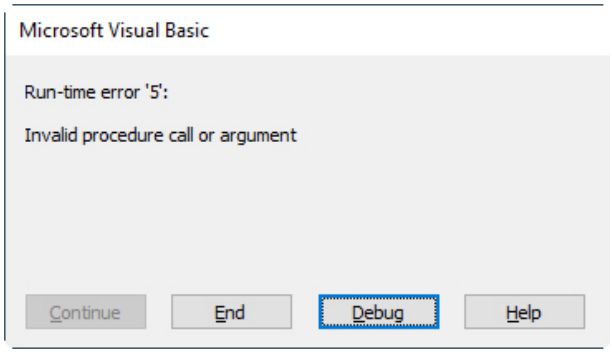
Jadi, Itu melakukan apa yang harus dilakukan tombol DEL. Kita membutuhkan bahwa setiap kali Kita menekan tombol itu, itu hanya menghapus huruf atau angka terakhir yang ditambahkan, seperti yang dilakukan rumus ini. Kemudian, kita membutuhkan tombol untuk menerapkan rumus ini. Bagaimana cara melakukannya di VBA?

Pertama, klik dua kali pada tombol dan tambahkan kode berikut:

```
X = Len(Display.Caption) - 1
Y = Display.Caption
Display.Caption = Left(Y, X
```

Itu dia!! Kita baru saja menambahkan dua variabel dan menggunakannya dalam urutan yang sangat sederhana! Cobalah. Tambahkan beberapa angka lalu tekan

tombol ini. Ini harus berhasil. Namun, apa yang terjadi jika tidak ada lagi nomor dan Kamu menekan tombol?



Itu jelas akan menghasilkan kesalahan. Bagaimana mengatasi ini? Caranya mudah, tambahkan beberapa kode termasuk IF:

```
Private Sub CmdDel_Click()
    If Display.Caption <> Empty Then
        X = Len(Display.Caption) - 1
        Y = Display.Caption
        Display.Caption = Left(Y, X)
    End If
End Sub
```

Kita telah menambahkan dua baris kode ini yang artinya ***"jika display.caption "tidak" kosong maka terapkan kodenya***. Jadi, jika kosong tidak akan berbuat apa-apa.

5.3 Rumus Excel di VBA

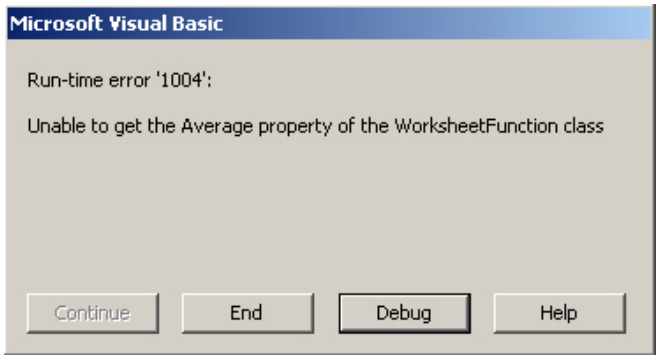
Ketiga, sekarang mari bekerja dengan tombol sama dengan:
Kita membutuhkannya untuk membuat semua operasi yang telah Kita tambahkan di display.caption. Bagaimana Kamu melakukannya?

Yang ini adalah yang tersulit dari kalkulator ini, dan tombol terpenting karena akan memberikan hasil dari semua yang telah kita lakukan. Ada beberapa cara untuk melakukan ini, tetapi kali ini mari kita gunakan: *Application.WorksheetFunction*. Setelah Kamu mengetik titik, Kamu akan melihat daftar yang panjang. Semua itu adalah daftar lengkap Rumus yang biasa Kita gunakan di Excel.

Mari tambahkan kode berikut dan lihat apa yang terjadi kali ini:

```
Private Sub CmdEquals_Click()
        Dim X As Variant
                X = Application.WorksheetFunction.Sum(Display.Caption)
                Display.Caption = Empty
                Display.Caption = X
End Sub
```

Seharusnya berhasil, bukan? Tapi itu tidak akan terjadi.



Baris pertama mendeklarasikan X sebagai varian yang artinya bisa menyimpan apa saja nilai.

X sama dengan SUM (display.caption)
Yang seharusnya seperti X = sum (2 + 2 * 4/7)

5.4 Menggabungkan VBA dan Spreadsheet

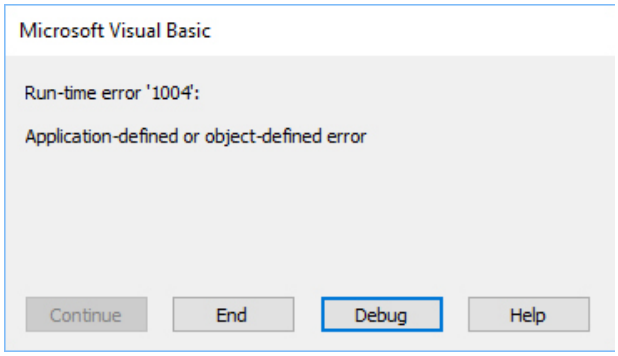
Jika Kamu mencobanya pada sel Spreadsheet, itu akan berhasil. Tetapi itu tidak berfungsi menggunakan variabel karena beberapa alasan, yang utama adalah karena seseorang tidak dapat benar-benar menambahkan angka 12 ditambah tanda, dll. Kita perlu membuat konversi dan beberapa prosedur lagi, tetapi dalam hal ini kita bisa mulai menggunakan spreadsheet. Kita akan menggunakan sel spreadsheet, sehingga dapat membiasakan diri menggabungkan VBA dengan Excel Sheets. Kita akan memerintahkan Excel untuk mengubah Display.Caption menjadi Formula di sel A1.

Jika Kamu merekam Macro dan menambahkan Formula, kodenya menjadi: Activecell.FormulaR1C1 = “=”… Tapi Kita tidak memerlukan R1C1 untuk kode itu di sini, itulah mengapa Kita menghapusnya.

```
Private Sub CmdEquals_Click()
        Range("A1").Formula = "=" & Display.Caption
        Display.Caption = Range("A1").Value

End Sub
```

Jalankan programnya dan coba sekarang! Berhasil!! Sekarang, tulis kode yang buruk, seperti: 5 *

Dan tekan =.



Bug lain muncul!

Itulah mengapa programmer sering mencoba aplikasi mereka. Kamu harus selalu melakukan hal yang sama. Seperti yang terlihat, Kita telah memperbaiki beberapa bug yang hanya berfungsi pada Kalkulator, tetapi setelah terbiasa dengan kode akan mengidentifikasi dan bahkan mencegahnya lebih mudah.

.

Untuk memperbaiki bug ini, Kita akan menerapkan Control Errors. Jadi, setiap kali terjadi kesalahan itu akan diterapkan sebagai gantinya jenis jendela ini. Tambahkan kode berikut:

```
Private Sub CmdEquals_Click()
        On Error GoTo A
                Range("A1").Formula = "=" & Display.Caption
        Display.Caption = Range("A1").Value
Exit Sub
A:
                Range("A1").Clear
        Display.Caption = "Error"
End Sub
```

Mungkin mudah dimengerti. Pada Error Goto A, dan Kita melihat A di tengah. Jadi, jika terjadi kesalahan, kode akan diterapkan dari A ke Akhir, yaitu Exit Sub.

Dikatakan Hapus sel A1 dan di Tampilan. Teks menampilkan pesan Kesalahan sebagai gantinya. Kita juga melihat di atas A: sesuatu yang mengatakan Exit Sub. Artinya, kode akan dibaca hingga hanya mencapai kode tersebut, jika tidak, kalkulator akan membaca sepanjang waktu apa untuk kesalahan juga. Kita tidak menginginkan itu.

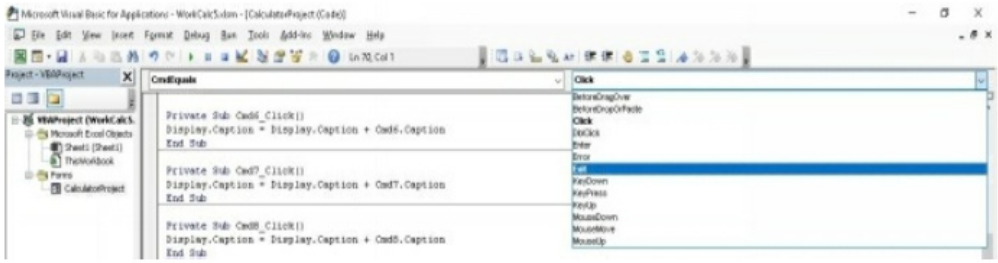
Coba lagi, tulis sesuatu yang salah seperti 5+ saja. Lihat?
Berhasil !!
Kalkulator akan terlihat lebih baik setiap saat.

5.5 Mulai Deklarasi:

Sekarang, saya pikir tidak baik melihat pesan Kesalahan bahkan setelah saya mengetik tombol lain:

Akan berguna jika menghapusnya secara otomatis setelah saya mengetik tombol lain. Untuk melakukan itu, Kita akan menambahkan kode baris lain ke tombol sama dengan. Kali ini tidak akan diterapkan ketika kita mengkliknya, tetapi ketika kita keluar dari tombol yang akan terjadi setelah kita mengetik tombol lain. Untuk melakukan itu, lakukan hal berikut:

1. Klik dua kali ke tombol Sama dengan.
2. Klik untuk melihat semua opsi seperti pada gambar dan Select Keluar



Sekarang Kamu telah berinteraksi dengan atribut berbeda yang dimiliki setiap tombol! Seperti yang Kamu lihat, kode dapat diterapkan jika mengklik tombol, setelah menahannya, keluar dan bahkan jika Kamu meletakkan kursor di atas tombol! Sungguh menakjubkan bukan ??

Setelah memilih Keluar, Kamu akan melihat kode baru di sana. Tambahkan berikut ini:

```
        Private Sub CmdEquals_Exit(ByVal Cancel As
MSForms.ReturnBoolean)
        Display.Caption = Empty
End Sub
```

Ini berfungsi seperti yang Kita inginkan sekarang. Saya pikir Kamu telah melihat bahwa pengkodean hanyalah logika, setelah mempelajari beberapa perintah lagi, Kamu akan melakukan hal-hal luar biasa.

Kamu telah belajar cara menghubungkan sedikit spreadsheet dan VBA, karena Kita menggunakan rentang A1. Bagaimana jika kita bahkan tidak ingin menggunakan sel spreadsheet apa pun ??

Seperti yang kita lihat, kita tidak bisa menggunakan variabel. Baik? Jelas, kita bisa !! Kita hanya perlu tahu caranya.
Sekarang, Kamu dapat menghapus semuanya dari tombol sama dengan, ubah kode dari:

```
On Error GoTo A
                Range("A1").Formula = "=" & Display.Caption
        Display.Caption = Range("A1").Value
Exit Sub
A:
                Range("A1").Clear
        Display.Caption = "Error"
End Sub

TO:
Private Sub CmdEquals_Click()
        On Error GoTo A
                Display.Caption = Application.Evaluate(Display.Caption)
Exit Sub
A:
                Display.Caption = "Error"
End Sub
```

Kita tidak langsung menuju solusi ini karena dengan melakukan kesalahan Kita mendapatkan pengalaman yang berharga. Mungkin, suatu saat tidak akan tahu fungsi apa yang akan digunakan dan mengetahui cara menautkan VBA dengan Spreadsheet akan sangat berguna untuk menyelesaikan masalah.
Kode diatas memiliki arti sebagai berikut:
Aplikasi sangat berarti dalam Excel.

Evaluasi adalah metode aplikasi atau Excel. Ini mengubah nama Microsoft Excel menjadi objek atau nilai. Dengan kata lain, dalam nilai-nilai yang dapat dioperasikan yang dapat dijumlahkan, dibagi, dikalikan, dll.

Sekarang, Kita tidak menggunakan spreadsheet sama sekali. Jadi, Mengapa kita harus membiarkannya terbuka?

Kita hanya tertarik untuk melihat Aplikasi yang Kita buat! Apa yang bisa kita lakukan?

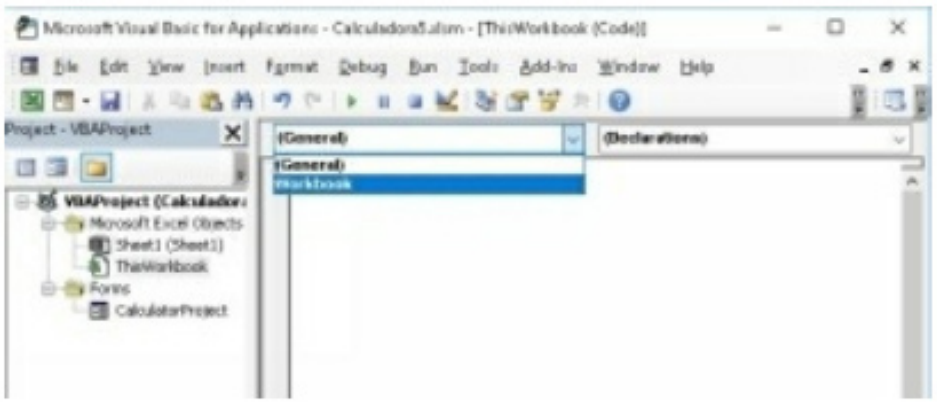
5.6 Deklarasi Buka dan Tutup: Menampilkan Formulir Tanpa Melihat Spreadsheet Apapun

Kalkulator ini berfungsi dengan baik. Tapi Kita tidak suka melihat keseluruhan spreadsheet di belakang. Kita bahkan tidak ingin melihat Excel dibuka sama sekali, karena Ini bukan hal yang umum untuk dilihat saat Kita menggunakan aplikasi, jadi mari kita keluarkan spreadsheet!

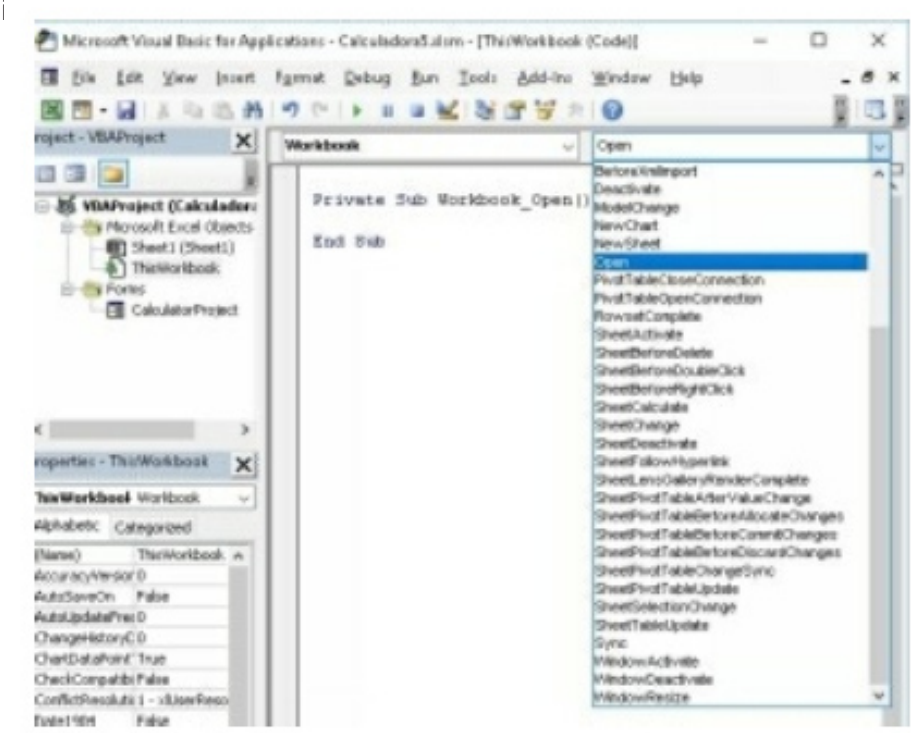
Ini akan terlihat seperti ini:

Kita hanya melihat Desktop di belakang, dan bukan spreadsheet apa pun.
Mari lakukan langkah demi langkah. Kita akan mempelajari lebih lanjut tentang fungsi VBA, dan bahkan beberapa sekuritas Macro.
Ikuti langkah ini:

1. Pertama-tama, Kita ingin kalkulator terbuka secara otomatis setelah file dibuka. Jadi, di Proyek VBA Folder (Calculadora), Select Buku Kerja Ini. Kamu harus melihat dua opsi, yang satu mengatakan Umum dan yang lainnya mengatakan Deklarasi.
2. Buka General dan Select Workbook.



3. Sekarang, Deklarasi Seharusnya diubah menjadi terbuka. Jika tidak, Select

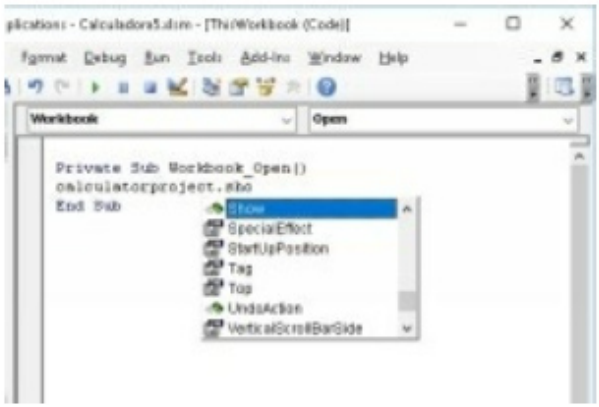


Artinya, setiap kali Buku Kerja Ini Terbuka, jalankan kode antara:

```
Private Sub Workbook_Open()

End Sub
```

4. Sekarang, di antara baris kode tersebut, masukkan yang berikut ini:

```
Private Sub Workbook_Open()
CalculatorProject.Show
End Sub
```



5. Simpan file Kamu.
6. Tutup dan buka untuk mencoba! Ini harus membuka proyek Kalkulator secara otomatis. Itulah Open Event dan sangat berguna dari proyek kecil hingga besar.

Kita masih tidak melihat aplikasinya saja, tetapi Excel masih muncul di sana. Jadi, apakah Kamu ingat bahwa dalam beberapa kode di atas “Aplikasi” berarti Excel? Artinya kita harus menulis beberapa kode yang mengatakan bahwa Aplikasi harus tidak terlihat. Tambahkan berikut ini ke kode yang baru Kamu tambahkan:

```
Private Sub Workbook_Open()
    Application.Visible = False
    CalculatorProject.Show

End Sub
```

Simpan file, tutup dan buka untuk mencobanya!

Sekarang Kamu hanya melihat Aplikasi !! Setelah Kamu menutup Kalkulator, semuanya tampak berfungsi dengan baik. Namun, masih ada yang salah.

Setelah Kamu menutup kalkulator, Excel tampaknya juga menutup, namun tetap terbuka tetapi tidak terlihat. Kita hanya menutup aplikasi !! Coba buka file yang sama dengan tempat Kamu menyimpan kalkulator. Kamu akan melihat bahwa itu terbuka tanpa memuat apa pun dan tidak menampilkan kalkulator, alasannya adalah terbuka!

Kita ingin bahwa sekali menutup kalkulator, itu juga menutup Excel. Untuk melakukan ini ikuti langkah-langkah berikut:

1. Buka VBA.
2. Proyek Kalkulator
3. Klik dua kali di bagian mana pun dari Formulir Kamu untuk melihat kode. Bukan judulnya juga bukan tombolnya, tapi bagian lain darinya.
4. Ini akan membuka sesuatu seperti ini:

```
Private Sub UserForm_Initialize()

End Sub
```

5. Ini sangat mirip dengan Deklarasi Terbuka yang kita pelajari, tetapi bekerja hanya dengan proyek. Namun, Kita tidak membutuhkannya sekarang. Kita membutuhkan kode penutup untuk Formulir Kita. Cara memilihnya sama seperti yang Kita lakukan Buku Kerja Ini. Pergi ke tempat dikatakan Inisialisasi, lalu Select yang paling logis, yang akan Mengakhiri.
6. Setelah Kamu memilihnya, Kamu akan melihat kode seperti ini:

```
Sub Private UserForm_Terminate ()
Akhiri Sub
```

7. Ingat kembali itu Aplikasi artinya Excel? Kamu akan perhatikan setelah Kamu mengetik Aplikasi. (Diikuti dengan titik) yang menampilkan daftar besar. Kamu bisa melihatnya, tapi hati-hati tentang bereksperimen. Terkadang, Kamu bisa membuat kesalahan besar dengan bermain-main dengan kode. Bahkan di luar Excel !! Memilih Application.Quit
   Kode Kamu akan terlihat seperti ini:

```
Sub Private UserForm_Terminate ()
Application.Quit
End Sub
```

8. Simpan file Kamu dan cobalah! Ini harus bekerja sekarang! Cepat dan mudah!

5.7 Macro Security

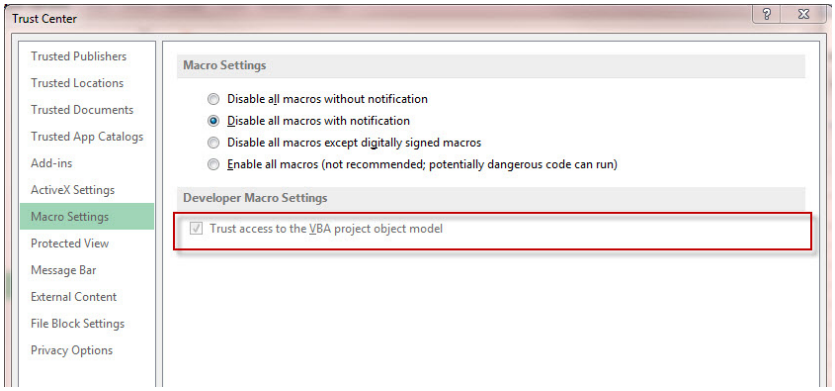
Kalkulator tampaknya telah berfungsi dengan baik sekarang. Namun, coba lakukan hal berikut:

1. 7 * 7
2. Tekan Tombol "Sama Dengan"
3. Kamu lihat 49
4. Sekarang, saya ingin 49 + 1. Jadi, ketik (+)…
5. Kamu akan melihat bahwa 49 terhapus secara otomatis. Ini tidak normal dalam kalkulator, tetapi saya harus menambahkan beberapa kode setelah hasilnya ditampilkan. Mari tambahkan kodenya.

Mungkin kita sedang dalam masalah. Bagaimana kita akan mengedit kode VBA if itu membuka dan menutup Excel secara otomatis? Bagaimana akan mengaksesnya ??

Ini adalah tentang bagaimana untuk memahami keamanan Macro. Kamu seharusnya sudah mengklik tombol Enable Macros untuk bekerja dengan VBA seperti yang kita pelajari di Bab 2. Setelah Kamu mengkliknya, Excel menyimpan jalur di mana file tertentu itu diizinkan untuk bekerja dengan Macro. Jadi, untuk menonaktifkan Macro yang diizinkan, yang harus Kamu lakukan adalah memindahkan file ke lokasi lain. Coba lakukan, lalu buka lagi. Kamu akan melihat pesan yang mengatakan bahwa Macro telah dinonaktifkan.

Cara lain untuk melakukannya tanpa mengubah lokasi adalah dengan mengubah nama file. Cobalah dan Kamu akan melihat Kamu menerima pesan yang sama tentang Macro. Jika Kamu melewatkannya dan tidak mengklik Aktifkan Konten, atau hanya ingin mengubah Opsi Keamanan Macro, Buka TAB Pengembang dan Klik Keamanan Macro. Kamu akan melihat semua opsi yang tersedia untuk Macro di semua buku kerja dan untuk file tertentu itu.



Sekarang Kamu dapat membuka VBA dan mengubah kode yang Kita butuhkan.

5.8 Comments

Sekarang, Kita tidak ingin kalkulator Kita menghapus apa yang tertulis di tampilan Kita, kecuali jika itu adalah pesan kesalahan. Ini akan memungkinkan kita untuk bekerja lebih efisien, jadi kita perlu kembali ke kode yang menghapusnya. Apakah Kamu ingat yang mana yang melakukan itu? Mungkin Kamu melakukannya karena hanya beberapa tombol. Namun, apa yang akan terjadi jika ada ratusan dan ratusan perintah?

Selalu mudah untuk menulis beberapa komentar yang memungkinkan Kita mengidentifikasi dengan mudah hal-hal yang dilakukan beberapa kode. Sangat umum untuk memperbaiki beberapa bug, menambah, menghapus, atau meningkatkan beberapa fungsionalitas saat pemrograman. Jadi, jangan lupa untuk menambahkan Komentar agar Kamu dapat lebih cepat membuka kode yang perlu diubah. Dalam hal ini kita perlu mengubah Tombol "Sama Dengan". Namun, ini memiliki dua kode yang ditambahkan. Satu berjalan saat kita mengklik tombol dan yang lainnya setelah kita berhenti memilihnya.

```
Private Sub CmdEquals_Click()
        On Error GoTo A
                Display.Caption = Application.Evaluate(Display.Caption)
        Exit Sub
A:
        Display.Caption = "Error"

End Sub
Private Sub CmdEquals_Exit(ByVal Cancel As MSForms.ReturnBoolean)
        Display.Caption = Empty

End Sub
```

Kode yang perlu kita ubah adalah yang kedua dari yang di atas. Tapi pertama-tama, mari tambahkan beberapa komentar untuk mengingat apa yang mereka lakukan untuk referensi di masa mendatang. Untuk menambahkan komentar, yang perlu Kamu lakukan adalah menulis apa yang Kamu inginkan dengan menambahkan 'di awal setiap kalimat. Dalam Contoh di bawah ini saya menambahkan dua kalimat, jadi saya perlu meletakkan 'di awal setiap kalimat.

Sub Private CmdEquals_Click ()

'Macro ini menampilkan pesan kesalahan jika kita mengetik matematika yang salah dan klik sama dengan 'jika tidak maka akan menjalankan bug dan program akan berhenti bekerja.

```
        On Error GoTo A
                Display.Caption = Application.Evaluate(Display.Caption)
        Exit Sub
A:
        Display.Caption = "Error"
End Sub
```

Private Sub CmdEquals_Exit (ByVal Cancel As MSForms.ReturnBoolean)

'Ini menghapus tampilan setelah kita memilih berhenti memilih tombol yang sama.

```
        Display.Caption = Empty
    End Sub
```

Baik. Mari perbaiki kodenya sekarang.

Kita ingin memberi tahu Excel: Jika layar menampilkan Pesan Kesalahan, hapus, jika tidak simpan.

Sekarang, mari terjemahkan pesan ke beberapa kode, sehingga Excel bisa mengerti.

```
    Private Sub CmdEquals_Exit(ByVal Cancel As
MSForms.ReturnBoolean)
```

Ini menghapus tampilan setelah Kita memilih berhenti memilih tombol yang sama.

```
    If Display.Caption = "Error" Then
        Display.Caption = Empty
Else
        Display.Caption = Display.Caption
    End If

End Sub
```

Persis seperti yang kita inginkan. Cobalah! Ketik sesuatu seperti 3 * lalu Sama. Ini menampilkan pesan kesalahan dan menghapusnya setelah Kamu mengetik sesuatu yang lain. Sekarang, ketik lagi 7 * 7 lalu sama dengan. Seharusnya Kamu melihat 49, sekarang ketik +1 Berhasil!

Itu hanya menghapus pesan kesalahan tetapi menyimpan nomornya. Sekarang, meskipun berfungsi dengan baik, Kita telah menambahkan beberapa kode yang tidak perlu: *Display.Caption = Display.Caption*

Mari kita hapus, termasuk "else" yang artinya "sebaliknya". Mungkin kita tidak akan menyadarinya, tetapi itu tidak profesional dan dalam tingkat yang sangat kecil itu menghabiskan sumber daya. Jangan pernah menambahkan kode yang tidak perlu ke program apa pun yang Kamu buat. Ini akan bagus untuk Kamu dan mereka yang menggunakan program Kamu. Jika Kamu membuat kebiasaan pengkodean yang buruk, Kamu mungkin akan mengalami masalah di masa depan ketika perlu menambahkan halaman dan halaman kode.
Setelah Kamu menghapus kode yang tidak perlu, akan terlihat seperti ini:

```
Private Sub CmdEquals_Exit(ByVal Cancel As
MSForms.ReturnBoolean)
   If Display.Caption = "Error" Then
      Display.Caption = Empty
   End If
End Sub
```

Ini bekerja juga! Kamu melihat? Kode yang Kita hapus tidak diperlukan!

Untuk selalu bekerja sebagai Pro, jangan pernah melupakan tip berikut:

- Selalu tambahkan komentar.
- Jangan pernah menambahkan kode yang tidak perlu.
- Selalu Select variabel yang benar

Mungkin Kamu bertanya-tanya mengapa tidak memilih variabel apa pun untuk Proyek Kalkulator. Faktanya, Kita melakukannya meskipun tidak mengumumkannya. Semua variabel Kita dideklarasikan secara otomatis sebagai Variant. Kita membutuhkannya karena Kalkulator ini dapat bekerja dengan miliaran dan miliaran angka. Sebuah byte, Integer, Long, dll. Tidak akan berfungsi jika

pengguna perlu menambahkan jumlah yang lebih besar dari yang diizinkan oleh mereka.

Ingat, jika Kamu tidak mendeklarasikan variabel, mereka ditambahkan secara otomatis sebagai varian yang membiarkan nilai apa pun, tidak peduli seberapa besar atau kecilnya.

5.9 Seluruh kode dalam Proyek Kalkulator

Untuk proyek ini, seluruh kode di CalculatorProject akan terlihat seperti ini:

```
Sub Private Cmd0_Click ()
    Display.Caption = Display.Caption + Cmd0.Caption
End Sub

Sub Private Cmd00_Click ()
    Display.Caption = Display.Caption + Cmd00.Caption
End Sub

Sub Private cmd1_Click ()
    Display.Caption = Display.Caption + cmd1. Caption
End Sub

Sub Private Cmd2_Click ()
    Display.Caption = Display.Caption + Cmd2.Caption
End Sub

Sub Private Cmd3_Click ()
    Display.Caption = Display.Caption + Cmd3.Caption
End Sub

Sub Private Cmd4_Click ()
    Display.Caption = Display.Caption + Cmd4.Caption
```

```
End Sub

Sub Private Cmd5_Click ()
    Display.Caption = Display.Caption + Cmd5.Caption
End Sub


Sub Private Cmd6_Click ()
    Display.Caption = Display.Caption + Cmd6.Caption
End Sub

Sub Private Cmd7_Click ()
    Display.Caption = Display.Caption + Cmd7.Caption
End Sub

Sub Private Cmd8_Click ()
    Display.Caption = Display.Caption + Cmd8.Caption
End Sub

Sub Private Cmd9_Click ()
    Display.Caption = Display.Caption + Cmd9.Caption
End Sub

Sub Private CmdAdd_Click ()
    Display.Caption = Display.Caption + CmdAdd.Caption
End Sub

Private Sub CmdCE_Click()
   Display.Caption = Empty
End Sub

Sub Private CmdDel_Click ()
 If Display.Caption = "Error" Then
```

```
        Display.Caption = Empty
    End If
End Sub

Private Sub CmdMinus_Click()
    Display.Caption = Display.Caption + CmdMinus.Caption
End Sub
Private Sub CmdPercent_Click()
    Display.Caption = Display.Caption + CmdPercent.Caption
End Sub

Private Sub CmdTimes_Click()
    Display.Caption = Display.Caption + CmdTimes.Caption
End Sub

Private Sub UserForm_Terminate()
    Application.Quit
End Sub


Sub Private CmdMinus_Click ()
Display.Caption = Display.Caption + CmdMinus.Caption End Sub
Sub Private CmdPercent_Click ()
Display.Caption = Display.Caption + CmdPercent.Caption End Sub

Sub CmdTimes_Click Private ()
Display.Caption = Display.Caption + CmdTimes.Caption End Sub
Sub Private UserForm_Terminate ()
Application.Quit
Akhiri Sub
```

Di Buku Kerja Ini kodenya akan terlihat seperti ini:

```
Private Sub Workbook_Open()
    Application.Visible = False
    CalculatorProject.Show
End Sub
```

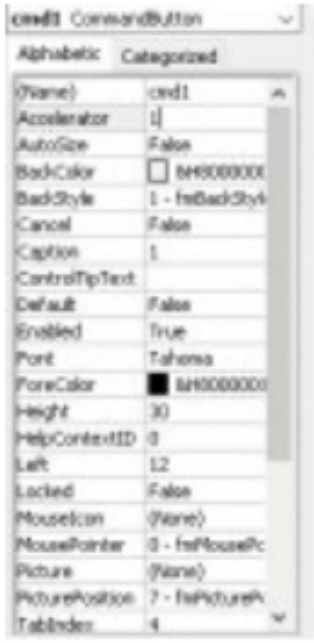
Cobalah! Ini bekerja dengan baik!

5.10 Menambahkan kombinasi Keyboard ke Tombol Perintah

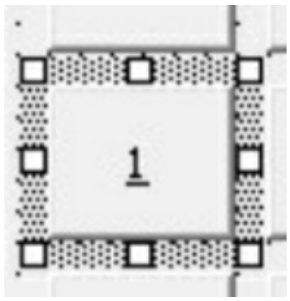
Ada hal lain yang bagus untuk ditambahkan. Pernahkah Kamu memperhatikan bahwa ini hanya berfungsi dengan mengklik tombol?
Saya pikir akan lebih baik lagi jika kita bisa menggunakan keyboard daripada menggunakan semua klik saja.
Mari kita lihat bagaimana melakukannya:

1. Buka Formulir Proyek VBA Kamu.

2. Sekarang, misalkan kita ingin menambahkan beberapa kombinasi tombol pintas ke Cmd1, yang merupakan tombol dengan nomor satu. Ini sangat sederhana.
   Klik di atasnya.
3. Buka propertinya.
4. Di Accelerator ketik tombol apa saja. Dalam hal ini paling banyak yang sesuai akan menjadi nomor 1, seperti pada contoh di bawah ini:



5. Setelah Kamu melakukannya, Kamu akan melihat nomor satu yang digarisbawahi.



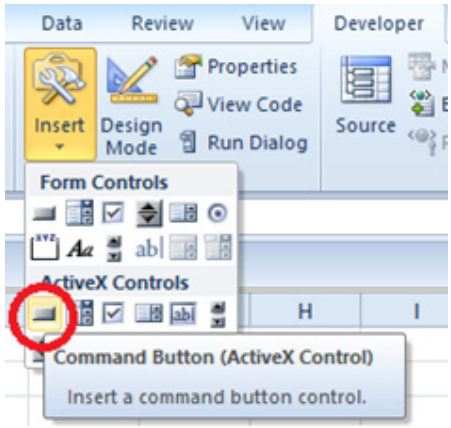
6. Ketika Kamu melihat sesuatu yang digarisbawahi seperti itu, tidak hanya di Excel tapi di semua Sistem Operasi Microsoft, itu berarti jika Kamu menahan Alt + "Huruf atau angka yang digarisbawahi" itu akan menjalankannya.
7. Coba jalankan Kalkulator dan Tekan Alt + 1 untuk melihat fungsinya. Ini harus mengetik nomor satu.
8. Lakukan hal yang sama untuk semua angka dan simbol lainnya, kecuali untuk DEL, CE, 00 dan =.

   Jika Kamu mencoba menambahkan tombol cepat Backspace ke DEL atau Supr ke CE

   Kamu akan melihat bahwa itu tidak mungkin. Setidaknya, tidak dengan cara ini.

5.11 Tombol Command

Saat Kamu membuka Kalkulator, Kamu akan melihat bahwa jika Kamu mengetik TAB, tombol akan diSelect, tetapi mungkin dengan cara yang tidak berurutan. Kita ingin memberi mereka perintah. Lakukan hal berikut untuk memperbaikinya:

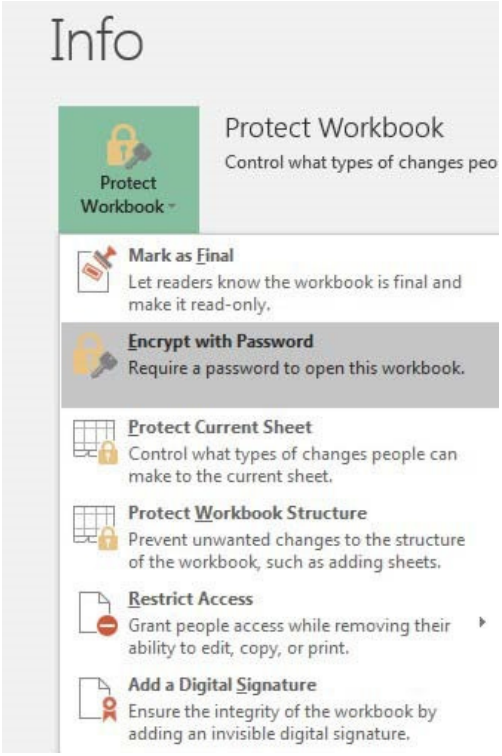


1. Buka VBA dan buka CalculatorProject Kamu.
2. Klik Lihat TAB.
3. Klik TAB Order.

4. Atur ulang daftar sesuai urutan yang Kamu inginkan dengan menggunakan tombol Pindah ke Atas dan Pindah ke Bawah.
5. Klik Ok setelah selesai.

5.12 Menambahkan Kata Sandi ke Kode VBA

Mungkin Kamu tahu bahwa Kamu dapat menambahkan Kata Sandi ke File Excel apa pun dengan mengikuti cara ini:



1. File TAB
2. Info
3. Protect Workbook
4. Encrypt with Password

Ini adalah opsi yang bagus, namun ini bukan Kata Sandi VBA. Kita tidak ingin memasukkan kata sandi setiap kali Kita membuka kalkulator. Satu-satunya hal yang Kita inginkan adalah melindungi kode Kita dari orang yang tidak berwenang yang ingin mengubahnya. Untuk melakukan itu hanya lakukan yang berikut:

Dengan cara itu, Kamu melindungi kode kalkulator !!

Soal Latihan

1. Bagaimana kamu Mengakses TAB Pengembang?
   a. Ini tersedia secara default di Excel.
   b. Klik kanan pada Ribbon, Customize the Ribbon, aktifkan Kotak Centang untuk Pengembang dan Terima.
   c. Buka file, Opsi, Lanjutan dan Aktifkan TAB Pengembang
2. Apa itu Makro)
   a. Ini adalah Formula Excel
   b. Ini adalah jalan pintas yang menjalankan proses yang direkam.
   c. Ini adalah proses bawaan yang disertakan dalam Excel.
3. Bagaimana cara membuat Makro?
   a. Mengklik Tombol Visual Basic
   b. Mengklik Tombol Macro
   c. Mengklik Tombol Rekam Makro
4. Untuk apa Referensi Relatif itu?
   a. Ini untuk Merekam Makro tanpa mengatur sel tertentu.
   b. Ini untuk Merekam Makro dengan sel tertentu.
   c. Tanpa itu, Makro tidak dapat diedit.
5. Bagaimana Cara Menjalankan Makro?
   a. Mengklik Tombol Makro
   b. Mengklik Tombol Rekam Makro
   c. Mengklik Tombol Referensi Relatif
6. Bagaimana cara menyimpan buku kerja dengan makro?
   a. Simpan saja file secara normal, Macro akan disimpan.
   b. Anda akan mendapatkan pemberitahuan, di mana kami harus ditolak dan kemudian pilih simpan sebagai Buku Kerja Macro-Enabled.
   c. Anda akan mendapatkan pemberitahuan, di mana kami akan diberi tahu bahwa kami menyimpan buku kerja berkemampuan Makro, lalu terima untuk menyimpan.

7. Apa itu Variabel?
   a. Ini adalah nilai yang tidak pernah berubah.
   b. Ini adalah nilai yang berubah.
   c. Ini adalah nomor khusus.

8. Berapa ukuran penyimpanan variabel byte dan jangkauannya?
   a. Ini menyimpan 2 byte dan beralih dari -32567 ke 32567.
   b. Ini menyimpan 1 byte dan beralih dari -256 ke -256.
   c. Ini menyimpan 1 byte dan mulai dari 0 hingga 256.

9. Apa keuntungan mendeklarasikan variabel?
   a. Menggunakan lebih sedikit RAM dan membuatnya berjalan lancar.
   b. Menggunakan lebih banyak RAM dan membuatnya berjalan lebih lambat.
   c. Ini tidak akan berfungsi jika variabel tidak dideklarasikan.

10. Apakah variabel wajib dideklarasikan?
    a. Hanya jika Memerlukan Deklarasi Variabel diaktifkan
    b. Itu wajib dalam segala keadaan
    c. Tidak, Excel mendeklarasikannya secara otomatis sesuai dengan nilai tambah.

11. Msgbox mengembalikan nilai tergantung pada tombol yang ditekan:
    a. Kami menetapkan nilai untuk setiap tombol
    b. Ini mengembalikan nilai tergantung pada tombol yang ditekan
    c. Nilai tidak dikembalikan

12. Apa itu Prosedur Pribadi?
    a. Artinya tidak ada yang dapat melihat Kode karena tersembunyi.
    b. Artinya tidak dapat disalin dan ditempel
    c. Itu tidak dapat dipanggil dari bagian lain dari kode

13. Apa itu Prosedur Umum?
    a. Ini berarti dapat dipanggil dan dijalankan dari bagian manapun dari Buku Kerja
    b. Bisa dilihat secara online
    c. Artinya Anda tidak dapat melindunginya dengan kata sandi

14. Apa itu Userform?
    a. Ini adalah Antarmuka Visual dari Aplikasi.
    b. Mereka adalah Tombol perintah, label dan Kotak Centang.
    c. Ini adalah aplikasi pramuat yang perlu kita edit untuk penggunaan Private.

15. Bagaimana Kamu membuat Userform?
    a. TAB Pengembang, Sisipkan Formulir
    b. Visual Basic, Masukkan TAB, Userform
    c. Visual Basic, Modul, Formulir

16. Apa cara memasukkan Rumus Excel di VBA?
    a. Application.Formulas
    b. Application.FormulaBarHeight
    c. Application.WorksheetFungsi

17. Formula apa yang dibuat untuk menghentikan Excel terlihat?
    a. Aplikasi. Berhenti
    b. Application.Visible = False
    c. Application.Visible = True

18. Apa cara untuk menonaktifkan Macro atau VBA setelah diaktifkan?
    a. Ubah nama file, pindahkan dari lokasinya dan di TAB Pengembang dan Keamanan Macro
    b. TAB Pengembang, hanya Keamanan Macro
    c. Tidak ada cara untuk melakukannya setelah diaktifkan.

19. Bagaimana cara mengatur kata sandi ke kode VBA Kamu?
    a. Visual Basic, Alat, Opsi, Keamanan, Kata Sandi.
    b. Visual Basic, Tools, VBAProject Properties, Protection, Passwords.
    c Visual Basic, File, Save as, Enkripsi kode VBA, Kata Sandi.

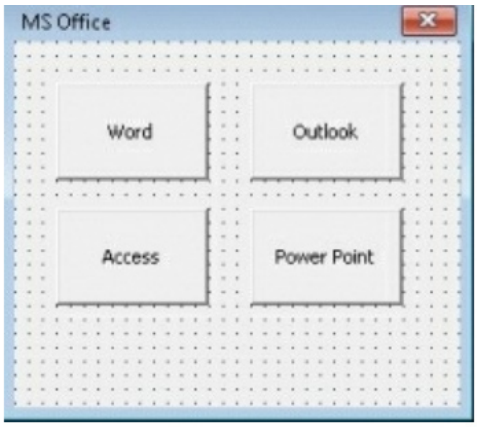
20. Berapa jumlah baris maksimum yang didukung worksheet Excel versi 2007?
    a. 78.487
    b. 1.028.176
    c. 1.176.048

VBA Terkait dengan Aplikasi lain

Membuka aplikasi lain dari Excel

Microsoft Excel VBA jauh lebih kuat dari yang pernah kita lihat. Ia bahkan dapat berinteraksi dengan aplikasi lain, seperti Microsoft Word, Access, Powerpoint, dll.

Kita hanya akan melihat tentang berinteraksi dengan aplikasi lain dari VBA. Untuk latihan ini, buat Formulir berikut:



Tambahkan kode berikut:

```
Private Sub cmdAccess_Click()
  Application.ActivateMicrosoftApp xlMicrosoftAccess
End Sub

Private Sub cmdOutlook_Click()
   Application.ActivateMicrosoftApp xlMicrosoftMail
End Sub
Private Sub cmdPowerPoint_Click()
```

```
Application.ActivateMicrosoftApp xlMicrosoftPowerPoint
End Sub

Private Sub cmdWord_Click()
   Application.ActivateMicrosoftApp xlMicrosoftWord
End Sub
```

Membuka Notepad

Ini akan memulai aplikasi setelah Kamu mengklik setiap tombol. Mungkin Kamu ingin memulai aplikasi lain. Mari coba buka Notepad. Tambahkan kode ini ke Tombol baru:

```
Sub Perintah PrivateButton1_Click ()
Redupkan Tugas Sebagai GKamu
    Task = Shell ("notepad.exe", 1)
 End Sub
```

Mengirim email Outlook dari Excel:

Tambahkan kode ini ke satu tombol, modul atau Macro:

```
Dim OutlookApp As Object
Dim Email As Object
Dim Subject As String
Dim EmailAddress As String
Dim Msg As String

'Create Outlook Object
Set OutlookApp = CreateObject("Outlook.Application")
```

```
    Subject = "This is my subject"
    EmailAddress = "email@gmail.com"
    Msg = "This is the body message"

'Create email and send it:

Set Email = OutlookApp.CreateItem(0)
With Email
  .to = EmailAddress
  .Subject = Subject
  .body = Msg
  .display
  End With
```

Seperti yang terlihat, secara praktis telah menulis email dari Excel! Jelas, Kamu dapat mengganti nilai untuk Sel Excel yang bisa sangat bermanfaat jika Kamu memiliki basis data dengan email dan pesan. Yang perlu Kamu lakukan adalah memikirkan sedikit tentang bagaimana melakukan apa yang Kamu butuhkan.

Jika ingin melihat berapa banyak aplikasi VBA yang tersedia untuk berinteraksi, Kamu hanya perlu mengklik Tool, Referensi.

Semua daftar ada aplikasi yang dapat berinteraksi dengan Excel, tetapi seperti yang Kamu lihat, kebanyakan dari mereka dinonaktifkan. Ini hanyalah fitur perlindungan. Jika mereka tidak cukup untuk Kamu, Kamu juga dapat Menelusuri lebih banyak.

Visual Basic for Applications adalah alat yang hebat dan merupakan salah satu bahasa pemrograman paling kuat yang digunakan di dunia untuk Analisis. Seperti kamu belajar VBA Kamu akan membuat proyek yang luar biasa.

Kamu telah mempelajari dasar-dasar menggunakan VBA dan bahkan membuat Aplikasi! Sekarang, terserah akan bagaimana menggunakan pengetahuan ini. Eksperimen pengkodean, buat Proyek baru!

Daftar Pustaka

Andi Sunyoto. 2007. Pemrograman Database dengan Visual Basic dan Microsoft SQL 2000. Yogyakarta: Andi Offset.

Anonim(2012).”VB 6 Mengatasi Error dengan Error Handler.” http://mydishaapp.blogspot.co.id/2012/11/vb-6-mengatasi-error-dengan-error.htm (diakses Januari 2021)

Ariansyah(2012).”Mengenal Apa Itu Macro dan VBA PadaMs.Excel.”http://blog.ariansyah.net/201 2/10/mengenal-apa-itu-macro-dan-vba-pada-excel.html (diakses Januari 2021)

Guantar Pangaribuan. 2008. Penggunaan Visual Basic Application Excel Untuk Program Perhitungan. PT Elex Media Komputindo. Jakarta

https://corporatefinanceinstitute.com/resources/excel/study/excel-vba# (diakses Januari 2021)

Karadelis, J. N. 1999. A Numerical Model for The Computation of Concrete Pavement Moduli: A Non-destructive Testing and Assessment Method. NDT & E International, 33(2), 77-84.

Madcoms. 2008. Microsoft Excel 2007 Pemograman VBA. Andi Publisher : Yogyakarta

Rusli, M. 2015. Aplikasi Perhitungan Indeks Prestasi Mahasiswa Menggunakan Excel Dan Macro Visual Basic For Application.I-Statement: Information And Technology Manegement (E-Journal).

Subari dan Yuswanto. 2008. Pemrograman Visual Basic 6.0. Cerdas Pustaka Publisher. Jakarta.

Tofik, Moch. 2009. Bekerja Secara Otomatis di Microsoft Office Excel 2007 Dengan Macro. PT Trans Media. Jakarta Selatan

Walkenbach, John. (2013). Excel VBA Programming For Dummies (3rd ed.). John Wiley & Sons.

Winpec Solution. 2010. Menguasai VBA Macro Microsoft Excel 2010. PT Elex Media Komputindo. Jakarta