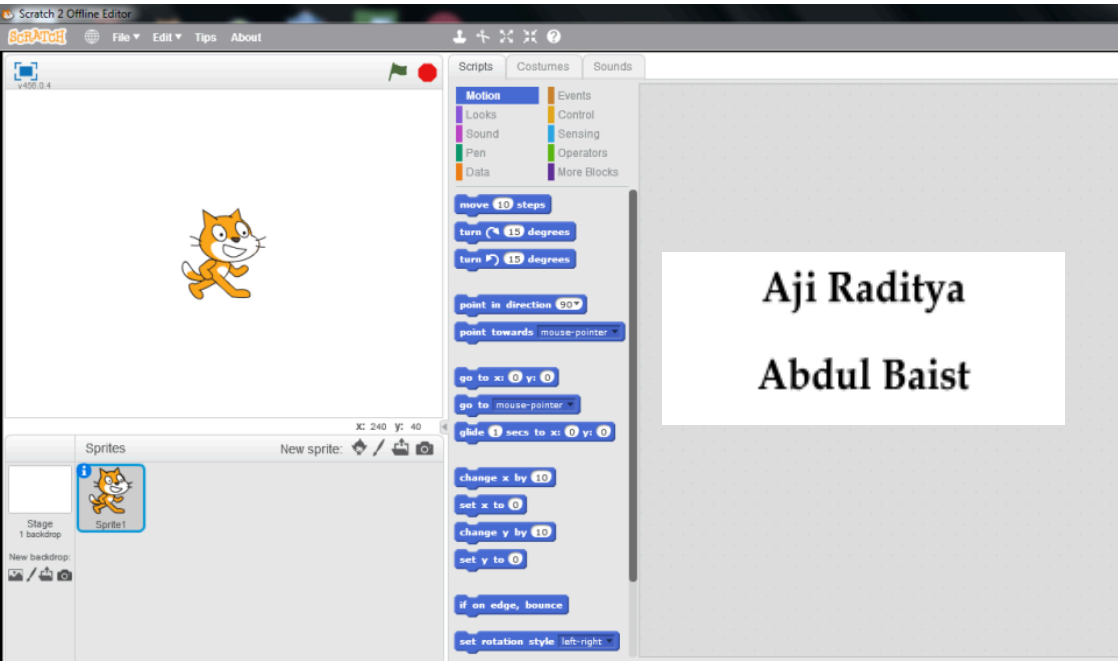
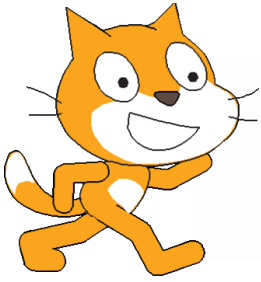
MODUL





SCRATCH 2.0

PEMROGRAMAN KOMPUTER

KATA PENGANTAR

Pertama-tama kami mengucapkan terima kasih kepada Tuhan Yang Maha Esa atas terwujudnya modul ini. Tujuan dari dibuatnya modul ini agar mahasiswa terbantu dapat mempelajari dasar-dasar pemrograman dalam mata kuliah Pemrograman Komputer. Dalam mata kuliah ini, perangkat lunak Scratch dipilih karena perangkat lunak tersebut dapat menampilkan kegiatan dan hasil pemrograman secara visual dan menarik. Modul ini dikembangkan dengan menggunakan pendekatan eksplorasi, dengan harapan pembaca dapat mencoba, memodifikasi dan selanjutnya dapat berkreasi dengan menggunakan perangkat lunak ini. Modul ini dimulai dengan pengetahuan awal tentang algoritma dan pemrograman. Berikutnya pengenalan terhadap perangkat lunak Scratch, dilanjutkan dengan instalasi Scratch dan beberapa contoh sederahana dalam menggunakan Scratch. Modul ini diakhiri denganpemrograman Scratch berupa pembuatan permainan sederhana dengan menggunakan perangkat lunak tersebut. Modul ini memberikan pengenalan

secara sederhana dan bertahap tentang Scratch dan diakhiri dengan contoh permainan sederhana yang menarik.

Harapan penulis, semoga modul ini dapat digunakan sebagai referensi bagi siapapun yang ingin mengenal dasar-dasar pemrograman khususnya pemrograman dengan menggunakan Scratch. Tetap Kreatif!!

Tangerang, 19 September 2017

Penulis

DAFTAR ISI

Pendahuluan

Deskripsi Mata Kuliah

Mata kuliah ini diperuntukan bagi mahasiswa yang mengambil mata kuliah pemrograman komputer. Kuliah ini mempelajari tentang dasar-dasar pemrograman, diawali dengan pengertian pemrograman, pembuatan program menggunakan flowchart dan pseudocode. Selanjutnya pengenalan salah satu perangkat lunak pemrograman, Scratch, diikuti dengan pemaparan dan penjelasan (serta contoh) pengembangan permainan menggunakan Scratch.

Rencana Pembelajaran

Rencana kegiatan pembelajaran yang dilakukan adalah sebagai berikut:

1. Menuliskan Flowchart/ pseudocode untuk beberapa kasus
2. Pengenalan fungsi-fungsi dasar pada program Scratch
3. Praktik pembuatan permainan sederhana menggunakan Scratch
4. Diskusi terkait Projek tugas akhir (pembuatan program menggunakan Scratch)

5. Membuat program menggunakan Scratch
6. Mendiskusikan (berkelompok) pembuatan program menggunakan Scratch
7. Mempresentasikan hasil (berupa program)

Petunjuk Penggunaan Modul

Modul ini tidak akan ada gunanya apabila hanya dibaca, mahasiswa sebaiknya secara aktif mengeksplorasi Scratch.

Capaian Pembelajaran Lulusan

Mahasiswa mampu mengembangkan kecakapan untuk memahami konsep dan prinsip pemrograman (menggunakan flowchart atau pseudocode) serta penerapannya dalam pembuatan program menggunakan Scratch.

Bentuk Evaluasi

Pada modul ini evaluasi berupa beberapa soal di setiap bab yang dikembangkan sedemikian rupa sehingga diharapkan pembaca dapat menguasai materi pada bab

tersebut. Selain itu, Pada akhir bab juga terdapat evaluasi berupa project.

PEMROGRAMAN KOMPUTER

Bagian ini memaparkan tentang dasar-dasar pemrograman. Materi yang dibahas pada bagian ini meliputi: **algoritma**, **flowchart** dan **pseudocode** untuk menggambarkan alur penyelesaian dari sebuah masalah.

Algoritma

Pendahuluan

Apakah Anda menyukai nasi goreng? Berikut ini 'algoritma' untuk membuat nasi goreng: Nasi Goreng Spesial (diambil dari: https://situsguru.files.wordpress.com/2010/12/pustaka-ebook-com_aneka-resep-nasi-goreng.pdf)

NASI GORENG SPESIAL

Bahan

- 250 gr nasi putih
- 100 gr udang sedang, kupas
- 5 buah bakso sapi, iris bulat
- 50 gram kacang polong
- 2 batang daun bawang iris halus
- 1 sdm saus cabai
- 1/4 buah tomat merah, iris
- 2 sdm minyak untuk menumis
- 1 sdm kecap manis
- 1 sdm kecap asin
- 1 sdt ang ciu
- Pelengkap: telur mata sapi dan kerupuk udang

Bumbu Halus

- 2 siung bawang putih
- 3 butir bawang merah
- 1 buah cabai merah
- 1/4 sdt merica
- garam secukupnya

Cara Membuat

1. Panaskan minyak, tumis bumbu halus sampai harum, masukkan daun bawang, dan tomat, aduk rata. Tambahkan udang dan bakso, aduk rata.
2. Masukkan nasi putih, tambahkan saus, kecap manis, kecap asin, dan ang ciu, aduk rata, angkat.
3. Sajikan hangat dengan taburan bawang goreng, kerupuk udang, telur mata sapi, dan acar ketimun.

Untuk: 3 porsi

Seperti Anda lihat bahwa **‘algoritma’** tersebut merupakan **sebuah resep**, yaitu sekumpulan petunjuk langkah demi langkah dalam mencampur bahan masakan dan menghasilkan **nasi goreng yang lezat**. Secara umum, algoritma dapat digambarkan sebagai **sebuah prosedur untuk menyelesaikan sebuah masalah**.

Target Transportasi : biar nyaman, murah, cepet

Problem : Orang ke kantor : macet, angkot ribet , mahal, telat ke kantor

Macet, Lalulintas, Waktu tugas ga cukup, Internet lemot / masalah,

Problem Solver :

Peluang : Rara = belum orang yang yang kasih solusi . Ternyata : ruang guru yang solusi

Peluang : Bisnis menjawab masalah yumna. Internet error saat hujan. Biznet, First Media-> mahal relatif, GSM

Nadim Makarim : algortima aplikasi Gojek (aplikasi) transportasi

Actor Ojek : saingan ojek, penghasilan ga tetap, bentrok opang

Dalam konteks pemrograman komputer, **algoritma** didefinisikan sebagai: **kumpulan operasi** yang tertata dengan baik, jelas dan dapat dihitung secara efektif, yang ketika

dieksekusi, menghasilkan suatu hasil dan penghentian dalam waktu yang terbatas (Schneider & Gersting, 2010).

Karakteristik suatu Algoritma

- Tertata dengan baik: **Langkah-langkahnya dalam urutan yang jelas**.
- Jelas: Operasi yang digambarkan dipahami oleh suatu komputasi tanpa penyederhanaan lebih lanjut.
- Dapat dihitung secara efektif: Komputasi tersebut sebenarnya dapat melakukan operasi.

Algoritma berasal dari nama terakhir **Muhammad ibnu Musa *Al-Khowarizmi***. Seorang ahli matematika terkenal dan penulis pada abad ke-8 dan 9 masehi yang berasal dari Persia. Al-Khowarizmi adalah seorang guru di Institut Matematika di Baghdad dan penulis buku Kitab **Al-Jabr wal Muqaabalah**, yang berarti "Aturan Menyatukan dan Memisahkan". Buku tersebut merupakan satu dari **buku teks matematika paling awal**, dan judul tersebut memberikan kita kata **Aljabar** (Schneider & Gersting, 2010).

Metode untuk Mengembangkan suatu Algoritma

1. Definisikan masalah: **Nyatakan masalah yang sedang Anda coba selesaikan dengan istilah yang singkat dan jelas**.
2. Buat daftar untuk input (**informasi yang diperlukan untuk menyelesaikan masalah**) : **mau kemana?Pilih kendaraan apa? Berangkat nya dari mana?** dan output (apa yang akan algoritma tersebut hasilkan). **GPS (titik koordinat); android ; e-money**
3. Gambarkan langkah-langkah yang diperlukan untuk mengubah atau memanipulasi input tersebut untuk menghasilkan output. Mulai pada tingkat tinggi terlebih dahulu, dan tetap memperbaiki langkah-langkah tersebut hingga operasi dapat dihitung secara efektif.
4. Uji algoritma: pilih sekumpulan data dan periksa bahwa algoritma Anda bekerja!

Dalam tahapan penyelesaian masalah pemrograman komputer, Anda akan mendesain **algoritma**. Ini berarti bahwa Anda harus menyadari strategi yang Anda gunakan untuk menyelesaikan masalah dapat diterapkan dalam masalah

pemrograman. Algoritma tersebut dapat didesain meskipun menggunakan **flowchart atau pseudocode**.

Flowchart

Flowchart merupakan **sebuah alat grafis yang menjelaskan cara penyelesaian masalah penanganan informasi (Chapin, 1970)**. Flowchart merupakan sebuah diagram yang terdiri dari **bentuk kotak, wajik, dan lainnya**, terhubung oleh panah, tiap bentuk mewakili sebuah langkah dalam proses, dan panah menunjukkan urutan. Flowchart mengkombinasikan simbol dan garis alir, untuk menunjukkan **seperti apa operasi dari suatu algoritma**.

Dalam komputasi, terdapat banyak simbol berbeda yang digunakan dalam flowchart. Dalam analisis proses bisnis, sepasang simbol cukup. Sebuah kotak dengan teks di dalamnya menandai sebuah langkah dalam proses, sementara sebuah wajik dengan teks mewakili sebuah keputusan.

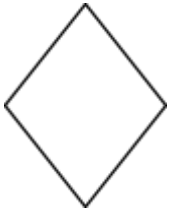
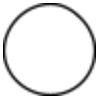
Jika flowchart terlalu berantakan untuk digambar, coba mulai lagi, tetapi biarkan semua poin keputusan dan berkonsentrasi pada cara yang paling sederhana mungkin. Kemudian sesi tersebut dapat kembali dilanjutkan dan

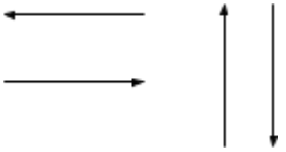
menambahkan poin keputusan setelahnya. Mungkin ini berguna untuk memulai dengan menggambar flowchart tingkat tinggi untuk keseluruhan bagian, dengan tiap kotak menjadi sebuah proses lengkap yang harus diisi nanti.

Dari pengertian umum ini bisa datang sejumlah hal, seperti ide perbaikan proses akan sering muncul secara spontan selama sesi pembuatan flowchart. Dan setelah sesi tersebut, Anda juga dapat menyusun prosedur tertulis, seperti sesi pembuatan flowchart yang merupakan cara yang baik untuk mendokumentasikan sebuah proses. Proses perbaikan diawali dengan memahami proses tersebut, dan membuat flowchart yang merupakan langkah awal menuju proses pemahaman.

Simbol Flowchart

Terdapat beberapa simbol dasar yang biasa digunakan dalam membuat flowchart program bahasa Assembly (bahasa pemrograman tingkat rendah), yaitu: *Terminal*, *Process*, *Input/Output*, *Decision*, *Connector*, dan *Predefined Process*. Berikut ini bukan merupakan daftar lengkap dari semua simbol flowchart yang mungkin, melainkan beberapa simbol yang sering digunakan dalam pemrograman bahasa Assembly.

Simbol	Nama	Fungsi
	Process	Menunjukkan jenis operasi internal di dalam prosesor atau memori.
	Input/ output	Digunakan untuk operasi Input/Output (I/O).
	Decision	Digunakan untuk memberikan pertanyaan yang dapat dijawab dalam format biner (Yes/No, True/False).
	Connector	Membuat flowchart dapat dibuat tanpa memotong garis atau alur yang berlawanan.

	Predefined Process	Digunakan untuk meminta subrutin atau program interupsi.
	Terminal	Menunjukkan awal atau akhir dari program, proses, atau program interupsi.
	Flow Lines	Menunjukkan arah aliran.

(Chapin, 1970)

Secara umum, terdapat banyak simbol flowchart yang baku.

Aturan Umum Membuat Flowchart

1. Semua kotak flowchart terhubung dengan panah (*Flow lines*).
2. Simbol flowchart memiliki titik masuk pada bagian atas simbol tanpa titik masuk lain. Titik keluar untuk semua simbol flowchart berada pada bagian bawah kecuali untuk simbol Decision.

3. Simbol Decision memiliki dua titik keluar; berada di samping atau bawah dan satu sisi samping lainnya.
4. Secara umum flowchart akan mengalir dari atas ke bawah. Bagaimanapun, aliran ke atas dapat ditampilkan selama tidak melebihi tiga simbol.
5. Connector digunakan untuk menghubungkan jeda dalam flowchart. Contoh:
    - dari satu halaman ke halaman lain.
    - dari bawah halaman ke atas halaman yang sama.
    - aliran ke atas lebih dari tiga simbol.
6. Subrutin dan program interupsi memiliki flowchart sendiri dan bebas (independent).
7. Semua flowchart dimulai dengan simbol Terminal atau Predefined Process (untuk program interupsi atau subrutin).
8. Semua flowchart berakhir dengan Terminal atau loop.

Tips membuat flowchart

- Buatlah diagram proses tersebut seperti benar-benar terjadi. Jangan mendokumentasikan seperti sebuah proses yang ditulis atau seorang manajer yang berpikir proses tersebut terjadi.

- Orang biasanya memodifikasi proses yang ada agar mendapatkan proses yang lebih efisien. Jika proses teoritis atau yang diinginkan tersebut digambarkan, masalah dengan proses yang ada tidak akan disadari dan tidak ada perbaikan yang dapat dibuat.

Catat semua keadaan yang benar-benar ditangani.

- Uji flowchart dengan mencoba mengikuti diagram tersebut untuk menampilkan proses yang dibuat. Jika terdapat masalah yang menampilkan operasi yang dibuat, catat setiap perbedaan dan modifikasi diagram tersebut agar tepat. Pendekatan yang lebih baik dengan cara meminta orang yang tidak akrab dengan proses tersebut untuk mencoba mengikuti flowchart dan catat pertanyaan atau masalah yang ditemukan.
- Sertakan tahapan mental dalam proses tersebut seperti keputusan. Tahapan tersebut terkadang hilang karena akrab dengan proses tersebut, bagaimanapun, representasikan sumber masalah yang disebabkan oleh kemungkinan kurang informasi yang biasanya dapat membuat keputusan tersebut tidak memadai

atau tidak tepat jika ditampilkan oleh orang yang berbeda.

Contoh Algoritma dan Flowchart

Contoh 1. Buatlah sebuah algoritma dan flowchart yang terkait untuk menambahkan skor yang diberikan: 26, 49, 98, 87, 62, 75.

Solusi:

a) Algoritma

[1] Start

[2] Jumlah = 0

[3] Dapatkan skor pertama

[4] Tambahkan skor pertama ke jumlah

[5] Dapatkan skor kedua

[6] Tambahkan ke jumlah

[7] Dapatkan skor ketiga

[8] Tambahkan ke jumlah

[9] Dapatkan skor keempat

[10] Tambahkan ke jumlah

[11] Dapatkan skor kelima

[12] Tambahkan ke jumlah

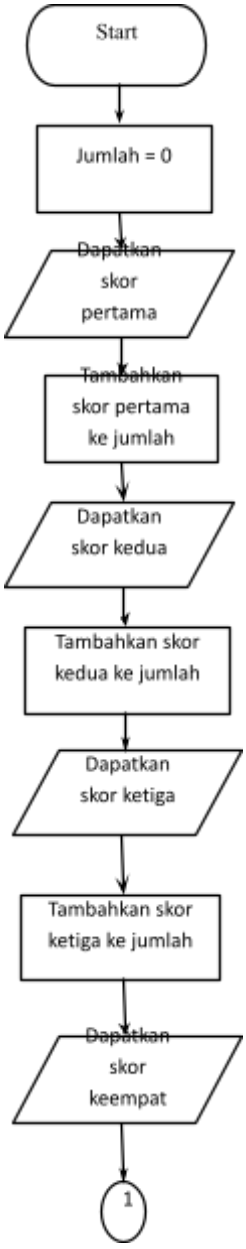
[13] Dapatkan skor keenam

[14] Tambahkan ke jumlah

[15] Output jumlah

[16] Stop

b) Flowchart yang terkait


Start
Jumlah = 0
Dapatkan skor pertama
Tambahkan skor pertama ke jumlah
Dapatkan skor kedua
Tambahkan skor kedua ke jumlah
Dapatkan skor ketiga
Tambahkan skor ketiga ke jumlah
Dapatkan skor keempat
1

Algoritma dan flowchart di atas menggambarkan langkah-langkah untuk menyelesaikan masalah menambahkan enam skor. Di mana satu skor ditambahkan ke jumlah pada suatu waktu. Algoritma dan flowchart keduanya harus selalu memiliki Start pada awal algoritma atau flowchart dan setidaknya satu Stop di akhir, atau di manapun pada algoritma atau flowchart. Karena kita ingin menjumlahkan enam skor, maka kita harus memiliki sebuah muatan yang merupakan hasil penjumlahan. Dalam contoh ini, muatan tersebut disebut jumlah dan kita memastikan bahwa jumlah harus dimulai dengan nilai nol pada langkah 2.

Contoh 2. Masalah dengan algoritma ini adalah bahwa, beberapa langkah muncul lebih dari sekali, maksudnya langkah 5 mengambil angka kedua, langkah 7, mengambil angka ketiga, dan seterusnya. Algoritma atau flowchart dapat diringkas sebagai berikut:

1. Start
2. jumlah = 0
3. Dapatkan sebuah nilai
4. jumlah = jumlah + nilai

5. Menuju langkah 3 untuk mendapatkan nilai berikutnya
6. Output jumlah
7. Stop

Flowchart yang terkait

Algoritma ini dan flowchart yang terkait lebih pendek dari sebelumnya. Dalam algoritma ini, langkah 3 hingga 5 akan diulang, di mana sebuah angka didapatkan dan

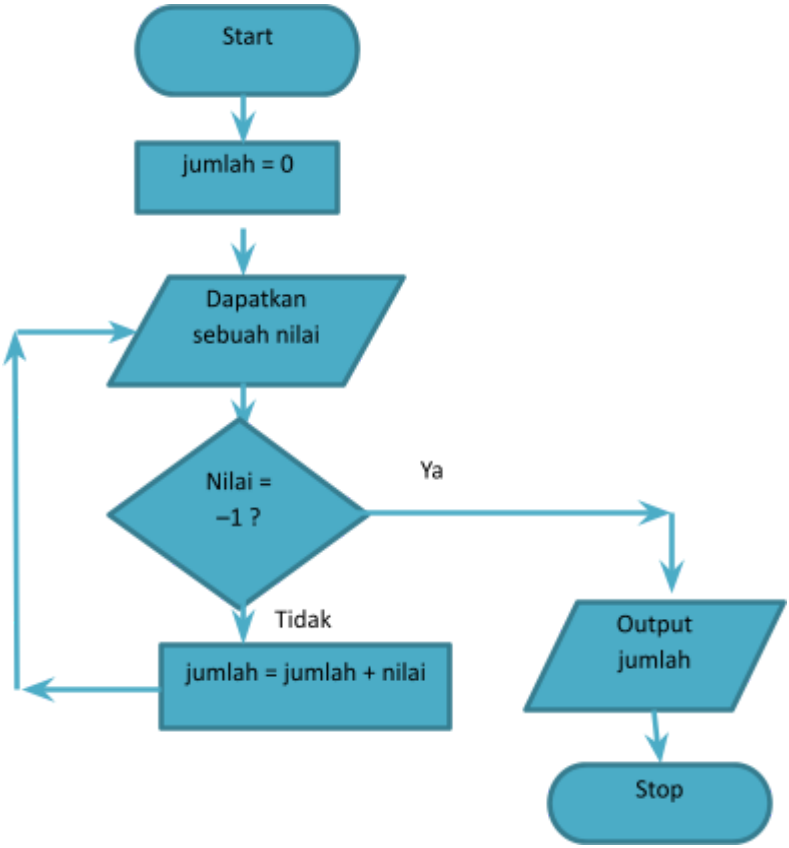
dijumlahkan ke jumlah. Sebuah garis alur yang dibuat berbalik ke langkah sebelumnya mengindikasikan bahwa porsi flowchart berulang. Sebuah masalah yang mengindikasikan bahwa langkah-langkah ini akan berulang tanpa akhir, menghasilkan sebuah algoritma atau flowchart tanpa akhir. Algoritma tersebut perlu diubah untuk mengatasi masalah ini. Dalam rangka menyelesaikan masalah ini, kita perlu menambahkan nilai terakhir untuk membuat daftar angka yang diberikan. Nilai ini harus unik sedemikian sehingga, tiap saat kita dapat sebuah nilai, kita uji nilai tersebut untuk melihat jika kita telah mencapai nilai terakhir. Dengan cara ini algoritma kita akan menjadi algoritma yang berhingga di mana berakhir dalam sejumlah langkah berhingga seperti ditunjukkan berikut. Terdapat banyak cara membuat algoritma berhingga.

List angka baru akan menjadi 26, 49, 98, 87, 62, 75, –1. Nilai –1 angka unik karena semua angka lain positif.

1. Start
2. jumlah = 0
3. Dapatkan sebuah nilai
4. Jika nilai sama dengan –1, menuju langkah 7
5. Tambahkan jumlah (jumlah = jumlah + nilai)

6. Menuju langkah 3 untuk mendapatkan nilai berikutnya
7. Output jumlah
8. Stop

Flowchart terkait



Pseudocode

Pseudocode merupakan salah satu alat yang dapat digunakan untuk menuliskan rencana awal yang dapat dikembangkan ke dalam sebuah program komputer. Pseudocode merupakan cara umum penggambaran sebuah algoritma tanpa menggunakan sintaks bahasa pemrograman tertentu (Levitin, 2012). Seperti namanya, *pseudo* code –tidak dapat dieksekusi pada komputer yang sebenarnya, tetapi memodelkan dan menyerupai kode pemrograman yang sebenarnya, dan ditulis secara kasar pada tingkatan detail yang sama.

Pseudocode, secara alami, hadir dalam berbagai macam bentuk, meskipun banyak meminjam sintaks dari bahasa pemrograman populer (seperti C, Lisp, atau FORTRAN). Bahasa dasar digunakan kapanpun rincian tidak penting atau mengganggu.

Buku sumber sains komputer sering menggunakan pseudocode dalam contoh mereka sehingga semua programmer dapat memahaminya, bahkan jika mereka tidak semua memahami bahasa pemrograman yang sama. Karena gaya pseudocode beragam dari penulis ke penulis, biasanya

terdapat pendahuluan yang menemani penjelasan sintaks yang digunakan.

Dalam desain algoritma, tahapan algoritma ditulis dalam teks bahasa Inggris bebas dan, meskipun keringkasan diinginkan, tahapan tersebut mungkin diperlukan selama untuk menjelaskan operasi tertentu. Tahapan tersebut dari sebuah algoritma dikatakan ditulis dalam pseudocode.

Banyak bahasa, seperti Pascal, memiliki sintaks yang hampir identik dengan pseudocode dan karenanya membuat transisi dari desain ke coding sangat mudah. Bagian berikut berkaitan dengan struktur kontrol (konstruk kontrol). Barisan, Seleksi dan Iterasi, atau Pengulangan.

Struktur Kontrol atau Struktur Logis

Kunci untuk desain algoritma yang lebih baik dan dengan demikian pemrograman berada dalam pembatasan struktur kontrol untuk hanya tiga konstruk. Diilustrasikan sebagai berikut:

Struktur barisan

Tipe pertama struktur kontrol disebut struktur barisan. Struktur ini merupakan struktur yang paling mendasar.

Struktur barisan merupakan sebuah kasus di mana tahapan dalam sebuah algoritma dibangun sedemikian rupa, tidak ada tahapan bersyarat yang diperlukan. Struktur barisan setara dengan garis lurus.

Sebagai contoh, misalkan Anda diminta untuk mendesain sebuah algoritma untuk menemukan rerata enam angka, dan jumlah angka tersebut diketahui. Pseudocodenya seperti berikut:

Start

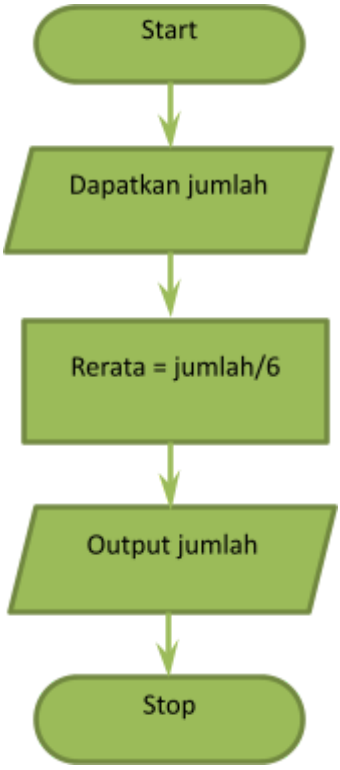
Dapatkan jumlah

Rerata = jumlah/6

Output rerata

Stop

Flowchart yang terkait akan muncul seperti ini:



Contoh 3. Ini merupakan pseudocode yang diperlukan untuk memasukkan tiga angka dari keyboard dan menampilkan hasilnya.

Gunakan variabel: jumlah, angka1, angka2, angka3 bilangan bulat

Terima angka1, angka2, angka3

jumlah = angka1 + angka2 + angka3

Print jumlah

End program

Contoh 4. Pseudocode berikut menggambarkan sebuah algoritma yang akan **menerima dua angka** dari **keyboard** dan **menghitung jumlah** dan **kali menampilkan jawaban pada layar monitor**.

Gunakan variabel: **jumlah**, **kali**, **angka1**, **angka2** bilangan real

Tampilkan "Masukkan dua angka"

Terima angka1, **angka2**

jumlah = angka1 + angka2

Print "**Jumlahnya adalah**", jumlah

kali = angka1*angka2

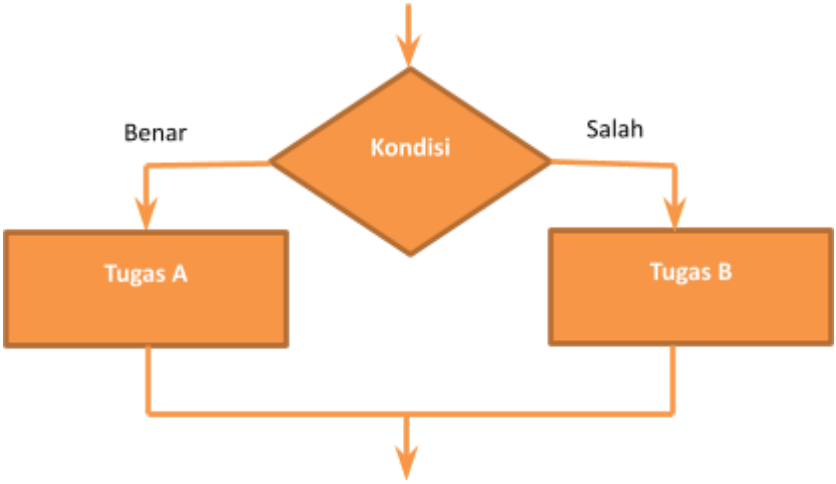
Print “Hasil kalinya adalah”, kali

End program

Struktur Keputusan atau Struktur Pilihan

Struktur keputusan atau biasa disebut struktur pilihan, merupakan kasus di mana dalam algoritma, harus memilih di antara **dua pilihan** dengan **membuat keputusan** bergantung pada **kondisi yang diberikan**. Struktur pilihan disebut juga struktur pemilihan kasus di mana terdapat dua atau lebih pilihan untuk dipilih.

Struktur ini dapat diilustrasikan dalam sebuah **flowchart** sebagai berikut:



Bentuk pseudocodenya sebagai berikut:

If kondisi **benar**

Then **lakukan Tugas A**

Else

Do Tugas B

If KKM Bhs Inggris =>70

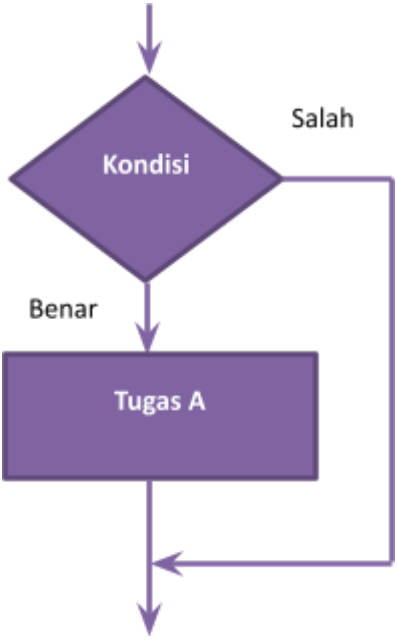
Then NIlai A

Else

Do Remedial(mengulang)

Dalam contoh ini, kondisi tersebut dievaluasi, jika kondisi benar Tugas A dieksekusi dan jika salah, maka Tugas B dieksekusi.

Variasi konstruksi gambar di atas ditunjukkan seperti berikut:



Dari struktur di atas, kita memiliki

If kondisi benar maka

Do Tugas A

Dalam kasus ini, jika kondisi salah, tidak terjadi apa-apa. Selain itu Tugas A dieksekusi.

Pilihan memerlukan hal berikut

- Pilih tindakan lain sebagai sebuah hasil pengujian kondisi logis
- Buatlah kode untuk menguji serangkaian tes logika

Membuat pilihan

Terdapat banyak hal di mana sebuah program diminta untuk mengambil tindakan lain. Sebagai contoh, hal di mana kita perlu mengambil tindakan berdasarkan pilihan pengguna. Semua bahasa komputer menyediakan sarana pilihan. Biasanya ini dalam bentuk pernyataan If dan pseudocode bukan pengeceualian untuk ini.

Kita akan menggunakan pernyataan if bersama dengan operator logika untuk menguji benar atau salah seperti pernyataan berikut.

If a = b

Print “a = b”

Tindakan diambil hanya ketika pengujian tersebut benar.

Operator logika yang digunakan dalam pseudocode adalah

= sama dengan

> lebih besar dari

< kurang dari

>= lebih besar atau sama dengan

<= lebih kecil atau sama dengan

<> tidak sama dengan

Contoh 5. Contoh berikut ini menunjukkan bagaimana struktur kontrol pilihan yang digunakan dalam sebuah program di mana pengguna memilih pilihan untuk mengalikan angka-angka atau menambahkannya atau mengurangkannya.

Gunakan variabel: **pilihan**, **bertipe karakter**

jawaban, **angka1**, **angka2**, **bertipe bilangan bulat**

tampilkan “pilih satu dari berikut ini”

tampilkan “k untuk perkalian”

tampilkan “j untuk penjumlahan”

tampilkan “p untuk pengurangan”

terima pilihan

tampilkan “masukkan dua angka yang Anda inginkan”

terima angka1, angka2

if pilihan = k then jawaban = angka1*angka2

if pilihan = j then jawaban = angka1 + angka2

if pilihan = p then jawaban = angka1 – angka2

tampilkan jawaban

Campuran operator logika

Terdapat banyak hal ketika kita perlu memperluas kondisi yang akan diuji. Sering terdapat kondisi yang perlu dihubungkan. Dalam bahasa sehari-hari kita mengatakan hal-hal seperti Jika saya memiliki waktu dan uang saya akan pergi berlibur. Kata dan berarti bahwa kedua kondisi mesti terpenuhi sebelum kita mengambil tindakan. Kita juga dapat berkata Saya bahagia pergi ke teater atau bioskop. Tautan logika kali ini adalah atau. Kondisi dalam pernyataan jika terhubung dengan cara yang sama. Kondisi yang dihubungkan dengan dan menghasilkan tindakan hanya ketika semua kondisi terpenuhi. Sebagai contoh, jika a > b dan a > c maka tampilkan “a palin besar”. Kondisi yang dihubungkan dengan

atau menghasilkan tindakan ketika salah satu kondisi terpenuhi.

Contoh 6. Program berikut ini memasukkan nilai ujian dan mengujinya untuk pemeringkatan. Nilai ujian tersebut adalah semua angka antara 1 dan 100. Peringkat berdasarkan kriteria berikut:

>= 80 sangat baik

>= 60 baik

>= 40 cukup

< 40 gagal

Pseudocodenya adalah

Gunakan variabel: nilai bertipe bilangan bulat

If nilai>=80 tampilkan “sangat baik”

If nilai >= 60 dan nilai < 80 tampilkan “baik”

If nilai >= 40 dan nilai < 60 tampilkan “cukup”

If nilai < 40 tampilkan “gagal”

Sebuah pernyataan if dengan sendirinya seringkali bukan cara terbaik dalam menyelesaikan masalah. Sekumpulan kondisi yang lebih elegan dapat dibuat dengan menambahkan pernyataan else ke pernyataan if. Pernyataan else digunakan untuk mengatasi situasi seperti ditunjukkan dalam contoh berikut.

Contoh 7. Seseorang digaji tinggi pada kategori pekerjaan 1 selain itu digaji normal.

If kategori pekerjaan 1

gaji tinggi

Else

gaji normal

Pernyataan else memberikan cara yang rapi dalam mengatasi kondisi lain. Dalam pseudocode kita tulis

If pekerjaan = kat1 then g-rate:= tinggi

Else g-rate = normal

Atau

If pekerjaan = kat1 maka

g-rate:= tinggi

else

g-rate = normal

Contoh berikut menggambarkan penggunaan pernyataan if … else dalam menampilkan kondisi lain berganda.

If gaji < 5000000 then

Pajak = 0

Else

If gaji > 5000000 AND gaji < 10000000 then

Pajak = 5000000 * 0,05

Else

Pajak = 10000000 * 0,30

Case statement

Pengulangan pernyataan if … else beberapa kali dapat sedikit membingungkan. Metode lain yang diberikan dalam

sejumlah bahasa adalah menggunakan selector yang ditentukan oleh kondisi lain yang diperlukan. Dalam pseudocode kita, ini disebut case statement.

Contoh 8. Program berikut menghasilkan sebuah pesan ke layar monitor yang menggambarkan asuransi yang tersedia berdasarkan kategori yang dimasukkan oleh pengguna.

Gunakan variabel: kategori bertipe karakter

Tampilkan “masukkan kategori”

Terima kategori

If kategori = T

Tampilkan “asuransi tidak tersedia”

Else

If kategori = A then

Tampilkan “asuransi ganda”

Else

If kategori = B then

Tampilkan “asuransi normal”

Else

If kategori = M then

Tampilkan “asuransi tergantung secara medis”

Else

Tampilkan “entry invalid”

Ini dapat diekspresikan dalam case statement sebagai berikut:

Gunakan variabel: kategori bertipe karakter

Tampilkan “masukkan kategori”

Terima kategori

DO kategori kasus

CASE kategori = T

Tampilkan “asuransi tidak tersedia”

CASE kategori = A

Tampilkan “asuransi ganda”

CASE kategori = B

Tampilkan “asuransi normal”

CASE kategori = M

Tampilkan “asuransi tergantung secara medis”

OTHERWISE

Tampilkan “entry invalid”

ENDCASE

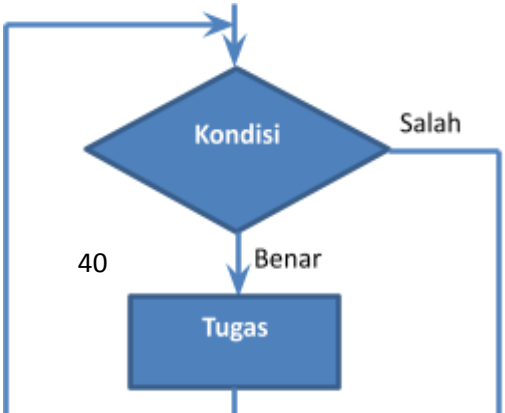
Selain menggunakan kata otherwise, dapat juga digunakan else.

Struktur Iterasi atau Pengulangan

Struktur ketiga menyebabkan tahapan tertentu diulang. Struktur pengulangan dapat diterapkan menggunakan

- Repeat Until Loop
- While Loop
- For Loop

Instruksi suatu program yang



mengulang suatu pernyataan atau barisan pernyataan beberapa kali disebut iterasi atau loop. Perintah yang digunakan untuk membuat pengulangan atau loop semuanya berdasarkan pada tes logika. Terdapat tiga konstruksi untuk iterasi atau loop dalam pseudocode kita.

Repeat Until Loop

Sintaksnya adalah

REPEAT

Sebuah pernyataan atau bagian dari pernyataan

UNTIL kondisi terpenuhi

Contoh 9. Sebuah segmen program meminta secara berulang entri sebuah angka dalam rentang 1 sampai 100 hingga angka yang valid dimasukkan.

REPEAT

DISPLAY “Masukkan sebuah angka antara 1 dan 100”

ACCEPT angka

UNTIL angka < 1 OR angka > 100

Contoh 10. Sebuah survei dikeluarkan untuk mengetahui olahraga yang paling populer. Hasilnya akan diketikkan ke dalam komputer untuk dianalisa. Tulislah sebuah program untuk menyelesaikan ini.

REPEAT

DISPLAY “Ketikkan dalam huruf yang dipilih atau Q untuk selesai”

DISPLAY “A: Atletik”

DISPLAY “R: Renang”

DISPLAY “S: Sepak bola”

DISPLAY “B: Bulu tangkis”

DISPLAY “Masukkan data”

ACCEPT huruf

If huruf = ‘A’ then

Atletik = Atletik + 1

If huruf = ‘R’ then

Renang = Renang + 1

If huruf = ‘S’ then

Sepakbola = Sepakbola + 1

If huruf = ‘B’ then

Bulutangkis = Bulutangkis + 1

UNTIL huruf = ‘Q’

DISPLAY “Skor atletik”, Atletik, “yang memilih”

DISPLAY “Skor renang”, Renang, “yang memilih”

DISPLAY “Skor sepakbola”, Sepakbola, “yang memilih”

DISPLAY “Skor bulutangkis”, Bulutangkis, “yang memilih”

While loop

Pengulangan jenis kedua yang akan kita lihat adalah iterasi while. Jenis loop bersyarat ini menguji untuk memutus kondisi pada permulaan loop. Dalam kasus ini tidak ada tindakan yang ditampilkan sama sekali jika uji pertama menyebabkan kondisi yang dievaluasi salah.

Sintaksnya adalah

WHILE (sebuah kondisi terpenuhi)

Sebuah pernyataan atau bagian dari pernyataan

ENDWHILE

Contoh 11. Sebuah segmen program untuk mencetak tiap karakter yang diketik di keyboard hingga karakter ‘q’ dimasukkan.

WHILE huruf < > ‘q’

ACCEPT huruf

DISPLAY “Karakter yang Anda ketik adalah”, huruf

ENDWHILE

Contoh 12. Tulislah sebuah program yang akan menampilkan pangkat dua dari sebarang input angka hingga input angka nol.

Dalam beberapa kasus, sebuah variabel harus diawali sebelum eksekusi loop seperti yang ditampilkan dalam contoh berikut.

Gunakan variabel: angka bertipe real

DISPLAY “Ketikkan sebuah angka atau nol untuk berhenti”

ACCEPT angka

WHILE angka < > 0

kuadrat = angka * angka

DISPLAY “Kuadrat dari angka tersebut adalah”, kuadrat

DISPLAY “Ketikkan sebuah angka atau nol untuk berhenti”

ACCEPT angka

ENDWHILE

For loop

Iterasi jenis ketiga, yang akan kita gunakan ketika banyaknya iterasi diketahui terlebih dahulu, adalah for loop. Ini, dalam bentuk paling sederhana, menggunakan sebuah inisialisasi variabel sebagai sebuah titik awal, kondisi berhenti bergantung pada nilai variabel. Variabel tersebut bertambah pada tiap iterasi hingga mencapai nilai yang diinginkan.

Sintaks pseudocodenya sebagai berikut:

FOR (keadaan awal, kondisi berhenti, kenaikan)

Pernyataan

ENDFOR

Contoh 13.

FOR (n = 1, n <= 4, n + 1)

DISPLAY “Loop”, n

ENDFOR

Penggalan kode tersebut akan menghasilkan output

Loop 1
Loop 2
Loop 3
Loop 4

Dalam contoh tersebut, n biasanya sebagai variabel loop, atau variabel hitungan, atau indeks loop. Variabel loop dapat digunakan dalam sebarang pernyataan loop. Variabel tersebut tidak boleh diberi sebuah nilai baru dalam loop, yang mana dapat mengubah perilaku loop.

Contoh 14. Tulislah sebuah program untuk menghitung jumlah dan rerata dari sederet angka. Solusi pseudocodenya adalah:

Gunakan variabel: n, hitungan bertipe bilangan bulat

jumlah, angka, rerata bertipe bilangan real

DISPLAY “Berapa banyak angka yang Anda ingin masukkan”

ACCEPT hitungan

jumlah = 0

FOR (n = 1, n <= hitungan, n + 1)

DISPLAY “Masukkan angka”

ACCEPT angka

jumlah = jumlah + angka

ENDFOR

rerata = jumlah / hitungan

DISPLAY “Jumlah angka tersebut adalah”, jumlah

DISPLAY “Rerata angka tersebut adalah”, rerata

Flowchart telah digunakan dalam bagian ini untuk mengilustrasikan bentuk ketiga struktur kontrol tersebut. Ini merupakan struktur kontrol dasar dari mana semua program dibangun. Di luar ini, flowchart melayani programmer dalam dua cara yang berebda: sebagai alat menyelesaikan masalah dan sebagai alat untuk mendokumentasikan sebuah program.

Contoh

Buatlah sebuah algoritma dan flowchart yang terkait untuk mendapatkan jumlah dari n angka.

Pseudocode Program

Start

jumlah = 0

Display “Masukkan nilai n”

Input n

For (I = 1, n, 5)

Input nilai

jumlah = jumlah + nilai

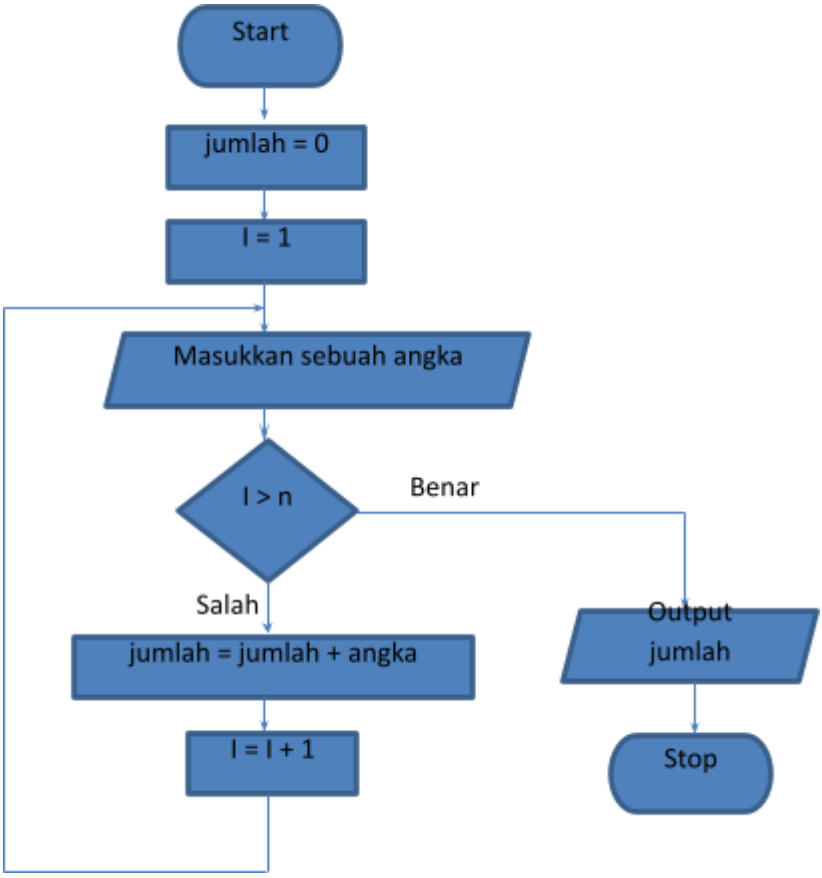
ENDFOR

Output jumlah

Stop

Dalam contoh ini, kita telah menggunakan I untuk membiarkan kita menghitung banyaknya pengulangan, yang kita dapat dari penambahan. Kita bandingkan I dengan n untuk memeriksa apakah kita telah kehabisan angka atau tidak dalam rangka menghentikan komputasi jumlah (atau untuk menghentikan struktur iterasi). Dalam kasus ini, I merupakan counter.

Flowchart yang terkait adalah sebagai berikut:



Latihan

1. Buatlah sebuah algoritma dan flowchart yang terkait untuk menemukan jumlah dari angka-angka 2, 4, 6, 8, …, n

2. Dengan menggunakan flowchart, tulislah sebuah algoritma untuk membaca 100 angka dan kemudian menampilkan jumlahnya.
3. Tulislah sebuah algoritma untuk membaca dua angka kemudian menampilkan yang paling besar.
4. Tulislah sebuah algoritma untuk membaca dua angka kemudian menampilkan yang paling kecil.
5. Tulislah sebuah algoritma untuk membaca tiga angka kemudian menampilkan yang paling besar.
6. Tulislah sebuah algoritma untuk membaca 100 angka kemudian menampilkan yang paling besar.

SCRATCH

Bagian ini akan dibahas sekilas tentang Scratch, dimulai dengan instalasi, fitur-fitur yang dimiliki serta contoh penggunaannya.

Pengenalan Scratch

Banyak orang menganggap pemrograman komputer sesuatu yang misterius dan proses rumit yang memerlukan pendidikan dan pelatihan teknik tingkat tinggi. Ini merupakan pendapat yang salah. Bahasa pemrograman BASIC telah ada sekitar beberapa dekade dan dikembangkan untuk mengajarkan programmer belajar untuk pertama kalinya bagaimana membuat program. Beberapa tahun ini, bahasa pemrograman jenis baru telah muncul, khususnya dalam membantu anak dan siswa belajar pemrograman. Salah satunya yang terbaik dan terbaru saat ini adalah Scratch. (Ford, 2009)

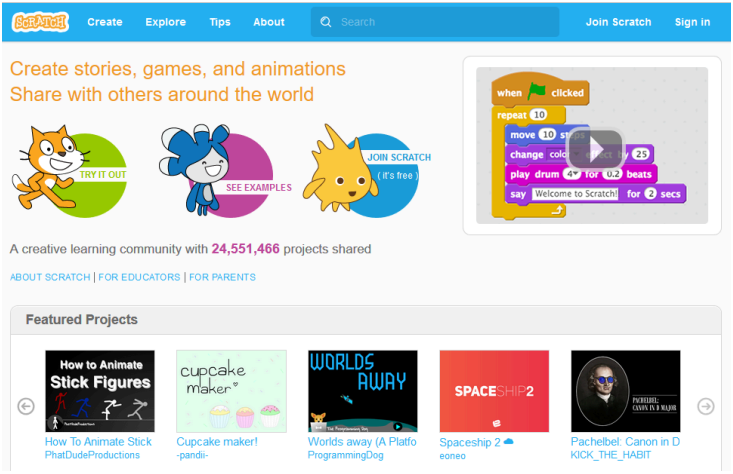
Scratch merupakan bahasa pemrograman yang memudahkan Anda untuk membuat cerita, animasi, games, musik, dan seni yang Anda miliki secara interaktif. Scratch dikembangkan oleh kelompok penelitian Lifelong Kindergarten di MIT Media Lab. (Kindergarten, 2013). Scratch dapat digunakan secara online atau offline. Untuk

penggunaan secara online disarankan untuk menggunakan Mozilla Firefox atau Google Chrome sebagai browser. Penggunaan Internet Explorer bermasalah ketika Sign In. (Lero, 2014)

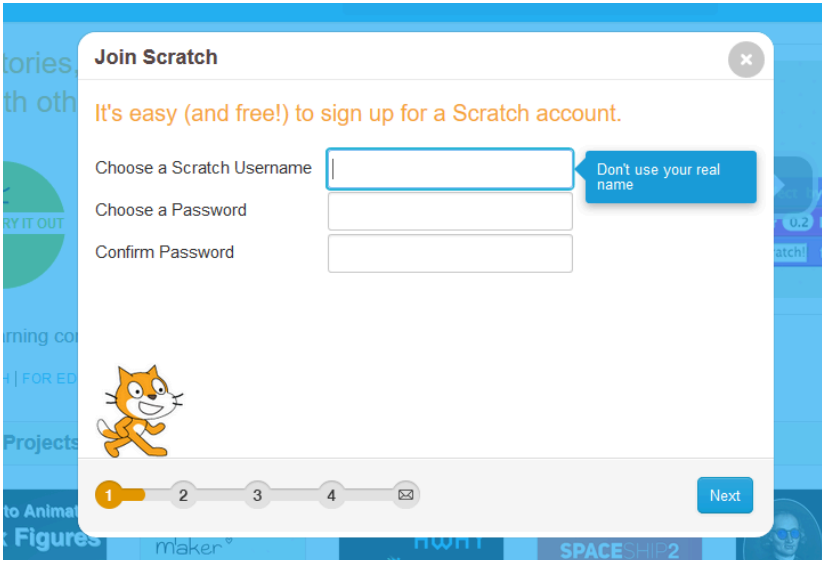
Scratch Online

Penggunaan Scratch secara online melalui https://scratch.mit.edu/ . Scratch memiliki komunitas di mana Anda dapat berbagi dan berkolaborasi dengan pengguna lain. Anda dapat bergabung ke dalam komunitas dengan membuat akun Scratch.

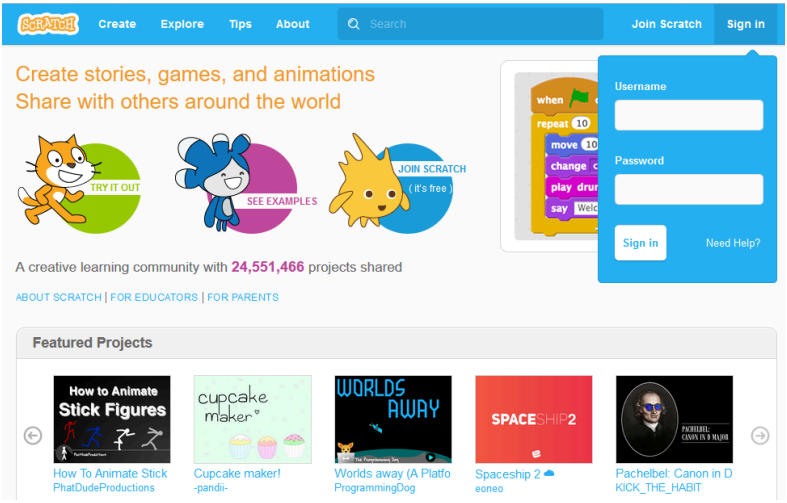
Berikut ini tampilan awal https://scratch.mit.edu/ .



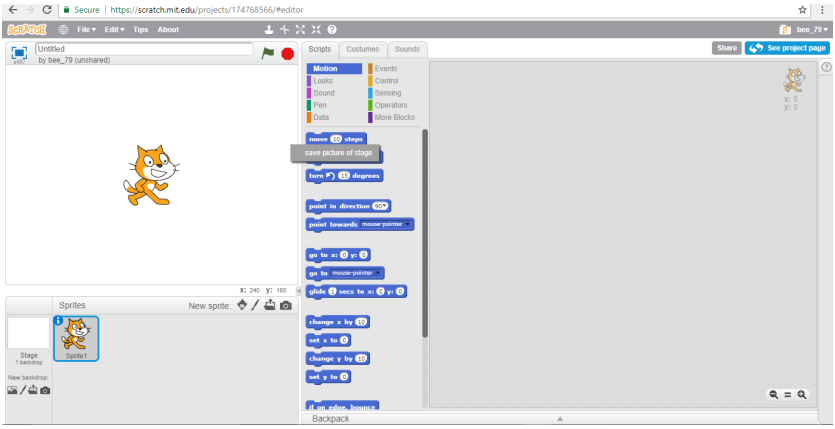
Klik “Join Scratch” untuk mendaftarkan akun Scratch jika Anda belum memilikinya.



Jika Anda telah memiliki akun klik “Sign In”.



Klik “Create” untuk menuju tampilan pemrograman Scratch secara online.



Fitur Scratch 2.0

Berikut ini fitur yang ada pada Scratch 2.0:

- Ketika Anda membuat sebuah project, hanya Anda yang dapat melihatnya. Setelah Anda membaginya (share), siapapun dapat melihat dan mengubahnya.
- Fitur Backpack memungkinkan Anda meng-copy dan memindahkan sprites, costumes, backdrops, dan scripts dari suatu project ke project lain. Anda dapat membuka Backpack di dalam setiap project (Backpack berada di bagian bawah layar. Backpack belum tersedia di offline editor).
- Anda dapat menggunakan webcam pada komputer Anda untuk berinteraksi dengan project dengan menggerakkan tangan atau tubuh Anda.
- Sekarang Anda dapat membuat blocks pemrograman Anda sendiri.
- Gunakan clone blocks dalam scripts Anda untuk membuat salinan sprites.

- Simpan angka dalam variabel cloud untuk membuat survey dan project lain.

Fitur Komunitas

1. Pada home page, Anda dapat melihat apa yang telah dibagi (share) oleh yang lain.
2. Ketika melihat sebuah project, klik 

untuk melihat bagaimana project tersebut bekerja dan bereksperimen dengan kode.
3. Di dalam setiap project, klik 

untuk menyimpan dan membuat perubahan versi Anda sendiri. Setelah Anda membaginya, halaman project tersebut akan menandai pembuat aslinya dan membuat link ke project mereka.
4. Klik username atau icon Anda untuk menuju halaman Profile, di mana Anda dapat menampilkan salah satu project Anda dan memberitahu apa yang sedang Anda kerjakan.
5. Orang-orang dapat menyampaikan komentar pada halaman Profile Anda dan 

akun Scratch Anda untuk melihat update.

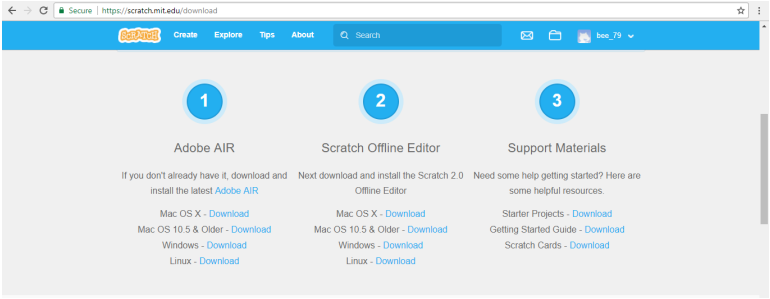
6. Studios dapat dikurasi oleh Anda dan lainnya yang Anda undang.
7. Fitur Search memungkinkan Anda menemukan dan melakukan pratinjau project dengan lebih mudah.

Scratch 2.0 Offline Editor

Perkuliahan ini akan menggunakan Scratch 2.0 offline editor. Ini berarti Scratch 2.0 offline editor tidak bergantung pada cepatnya koneksi internet. Berikut ini akan diuraikan tahapan menginstal Scratch 2.0 offline editor.

Instalasi Scratch

Anda dapat mengunduh Scratch di https://scratch.mit.edu/download .



Berikut ini tahapan instalasi scratch:

1. Instal Adobe AIR: Unduh dan jalankan Adobe AIR installer.

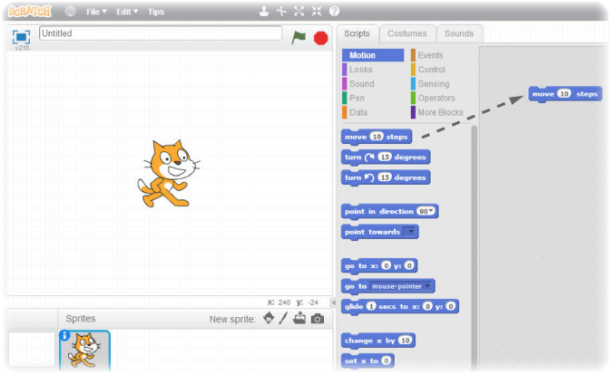
   Ini akan mengunduh installer tersebut ke dalam folder unduhan Anda. Berikutnya klik dua kali installer yang ada dalam folder tersebut. Adobe AIR akan memulai instalasi.
2. Instal Scratch offline editor: Unduh dan jalankan Scratch installer.

   Ini akan mengunduh installer tersebut ke dalam folder unduhan Anda. Berikutnya klik dua kali installer yang ada dalam folder tersebut. Scratch akan memulai instalasi.

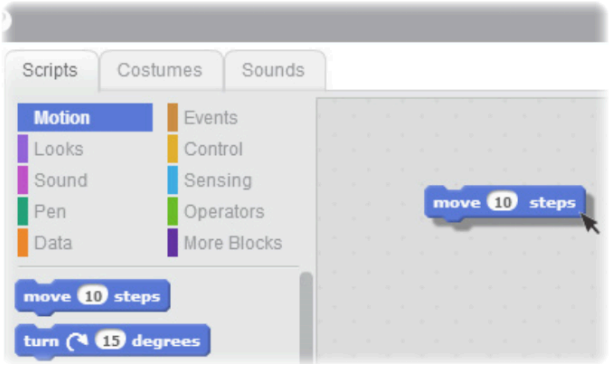
Memulai Scratch

Berikut ini panduan untuk membuat sebuah project di Scratch yang disusun oleh kelompok penelitian Lifelong Kindergarten:

1. **Start Moving**

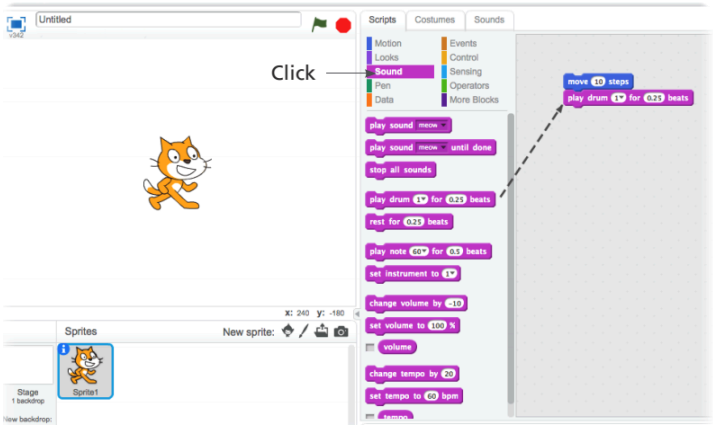


Drag block Move ke dalam area Scripts.



Klik blok tersebut

2. Add a Sound



Drag Play Drum ke dalam area Scripts dan pasangkan dengan block Move.

Klik dan dengarkan.

Jika Anda tidak dapat mendengarnya, periksa bahwa sound pada komputer Anda dalam keadaan on.

Anda dapat memilih drum yang berbeda dari menu seperti gambar di atas.

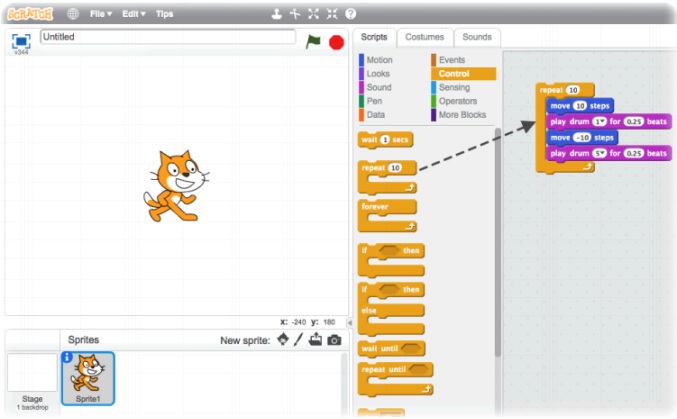
3. Start a Dance

Tambahkan block Move. Klik di dalam block tersebut dan ketikkan tanda minus.

Klik pada block tersebut untuk menjalankannya.

Tambahkan block Play Drum lain, kemudian pilih sebuah dari menu tersebut. Klik untuk menjalankannya.

4. Again and Again



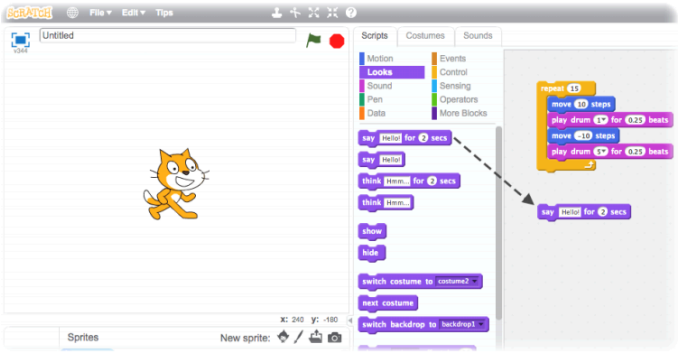
Drag block Repeat dan letakkan pada bagian atas tumpukan block. Anda ingin mulut dari Repeat meliputi block lainnya.

Anda dapat merubah banyaknya perulangan (repeat).

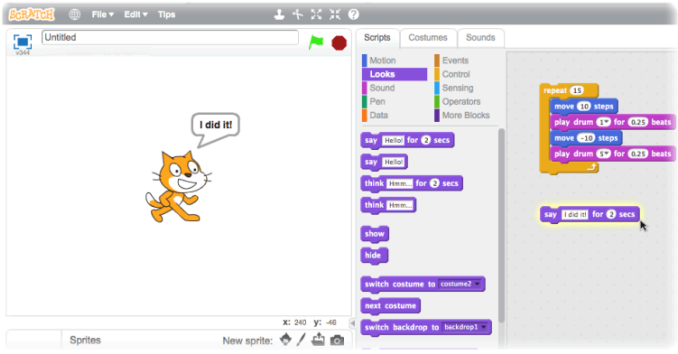
Klik untuk menjalankan.

Untuk menjalankan, Anda dapat mengklik bagian mana saja pada tumpukan block tersebut.

5. Say Something



Klik kategori Looks dan drag block Say.

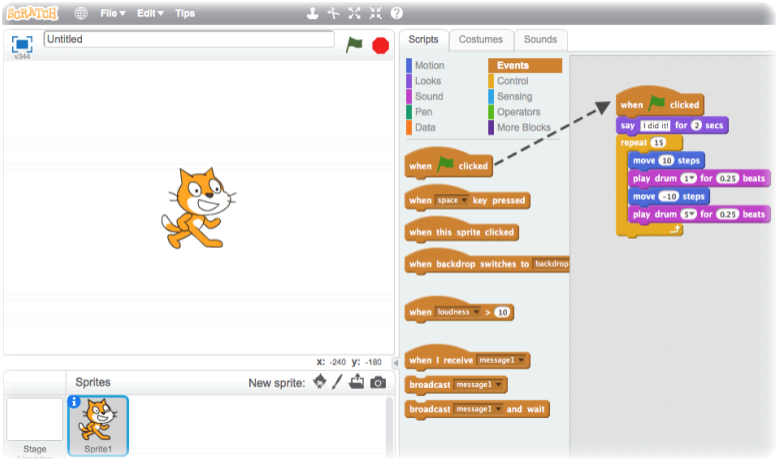


Klik di dalam block Say dan ketikkan untuk merubah kata.

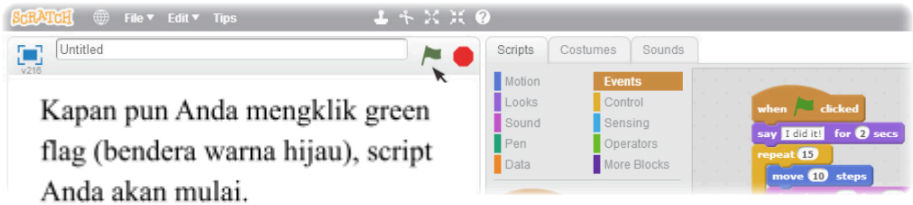
Klik untuk mencobanya.

Kemudian pasangkan block Say pada bagian atas.

6. Green Flag



Drag block  dan pasangkan di bagian atas.

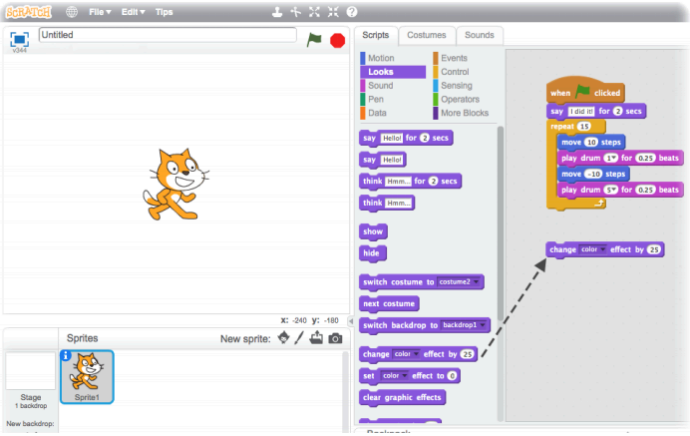


Kapan pun Anda mengklik green flag (bendera warna hijau), script Anda akan mulai.

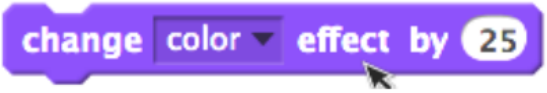
Untuk berhenti, klik tombol stop.

7. Change Color

Sekarang coba sesuatu yang berbeda …

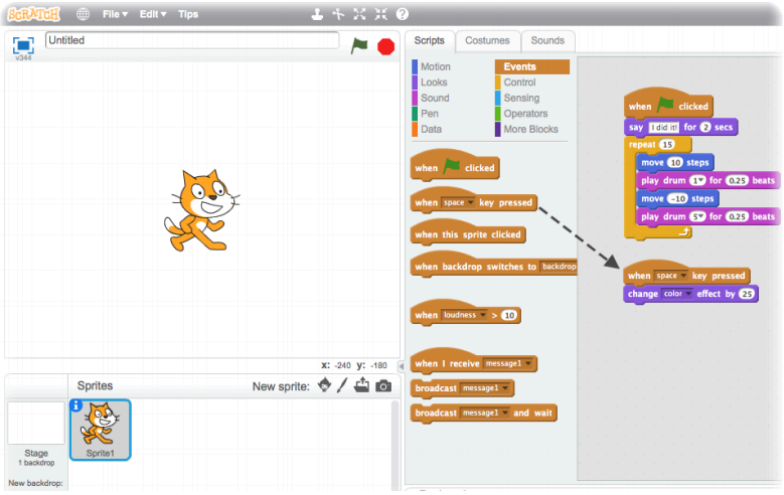


Drag block Change Effect ke dalam area script.



Klik untuk melihat yang terjadi.

8. Key Press



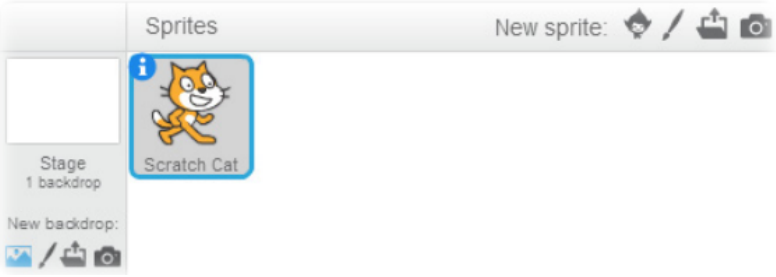
Klik 

Sekarang tekan spasi pada keyboard Anda.

Anda dapat memilih key yang berbeda dari menu tersebut.

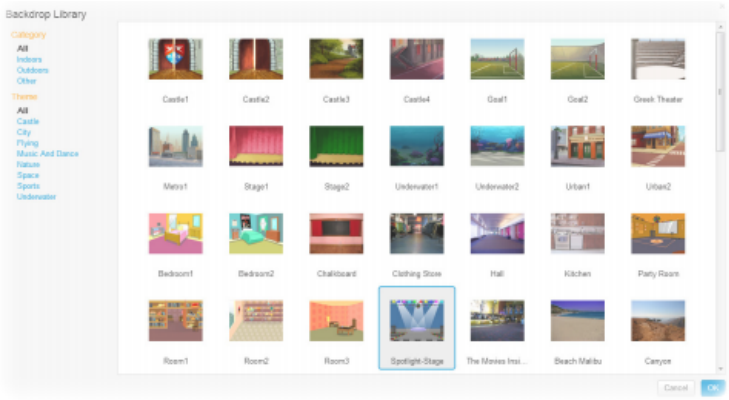
9. Add a Backdrop

Anda dapat menambahkan sebuah backdrop ke dalam Stage.



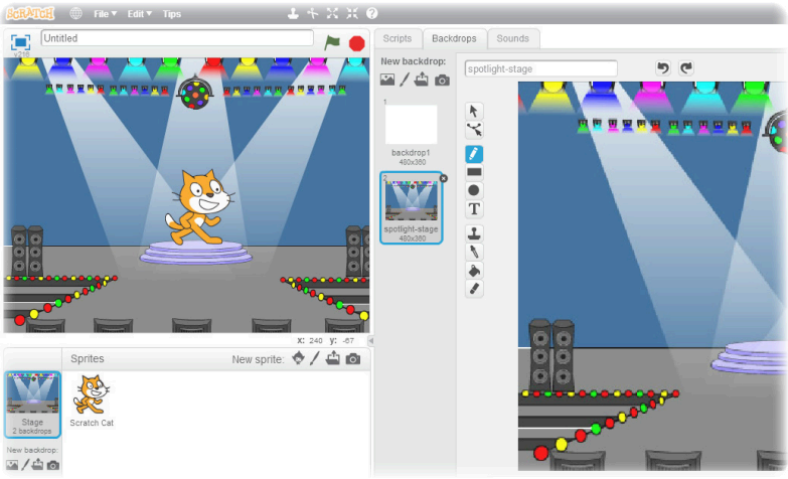
Klik  untuk memilih backdrop baru.

Pilih sebuah backdrop dari library (seperti “Spotlight-Stage”).



Klik Ok.

Sekarang backdrop baru muncul di Stage.



10. Add a Sprite

Setiap objek di dalam Scratch disebut sprite.

Untuk menambahkan sprite baru,

klik salah satu tombol tersebut.

Tombol Sprite:

 Memilih dari library

 Melukis sprite milik Anda

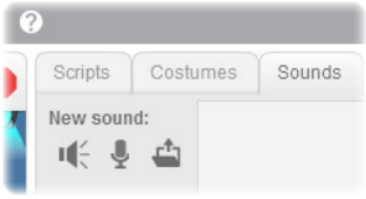
 Unggah sprite atau gambar milik Anda

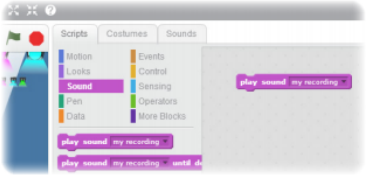
 Mengambil gambar (dari webcam)

Untuk menambahkan sprite ini, klik  kemudian klik People dan pilih "Cassy Dance". Anda dapat menggeser karakter ke mana yang Anda inginkan.

11. Explore!

Sekarang Anda dapat mengatakan kepada sprite apa yang harus dilakukan. Cobalah yang berikut ini, atau jelajahi sendiri.





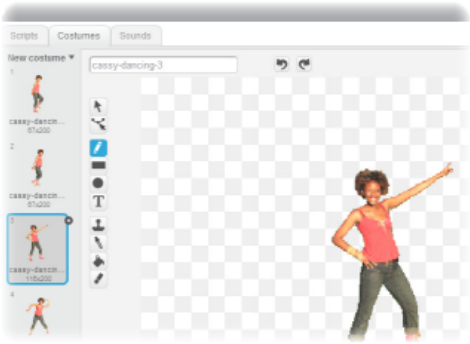
Tambahkan Suara

Klik tab Sounds. Anda dapat memilih

Record suara milik Anda

Atau Import file suara. (format MP3, AIF, atau WAV)

Kemudian, klik tab Scripts, dan drag block Play Sound ke dalam area script.

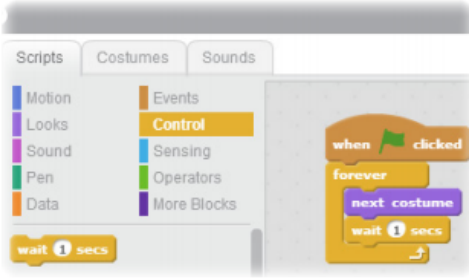


Ganti Kostum

Setiap sprite memiliki lebih dari sat kostum.

Untuk mengganti kostum, klik tab Costumes.

Kemudian klik kostum yang berbed untuk sprite tersebut.



Buatlah Animasi

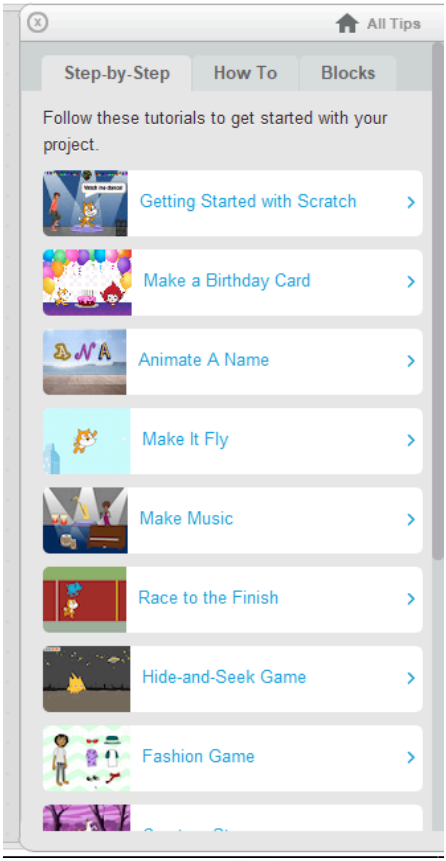
Anda dapat membuat animasi untul sebuah sprite dengan mengganti kostum.

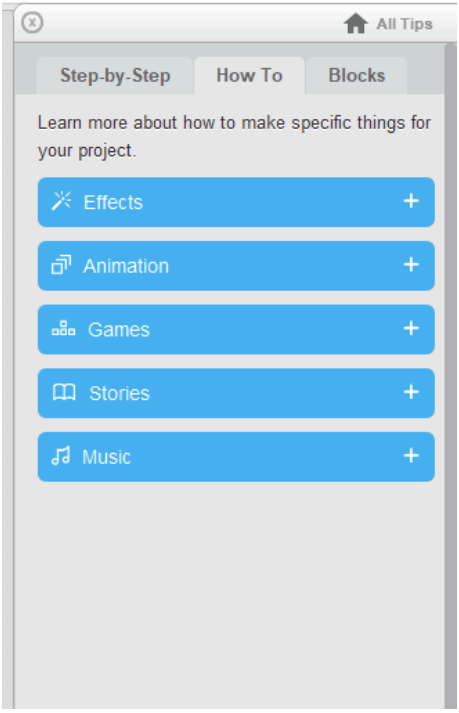
Klik tab Scripts.

Buatlah sebuah script yang bergant kostum.

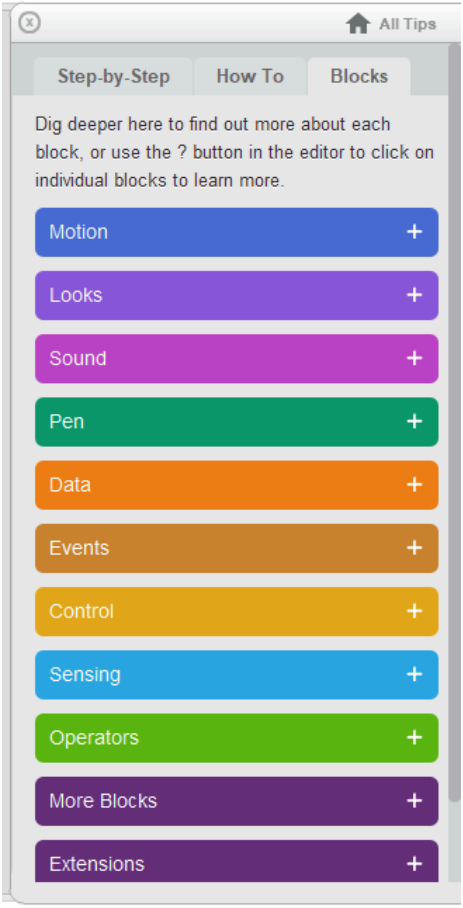
12. Tips!

Untuk lebih banyak ide, klik Tips:


All Tips
Step-by-Step
How To
Blocks
Follow these tutorials to get started with your project.
Getting Started with Scratch
Make a Birthday Card
Animate A Name
Make It Fly
Make Music
Race to the Finish
Hide-and-Seek Game
Fashion Game



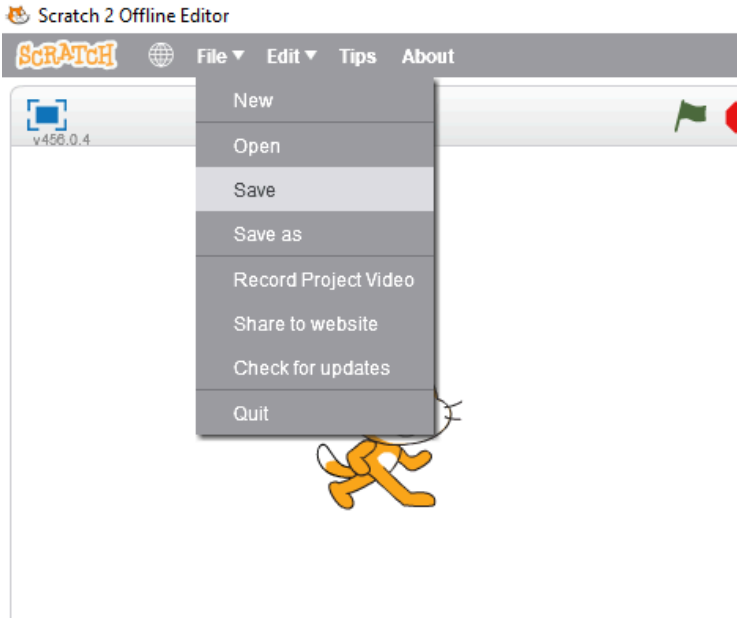
Tips Window menampilkan contoh script yang dapat Anda gunakan untuk project Anda.



Juga menjelaskan apa yang dilakukan tiap block di dalam Scratch.

Save and Share

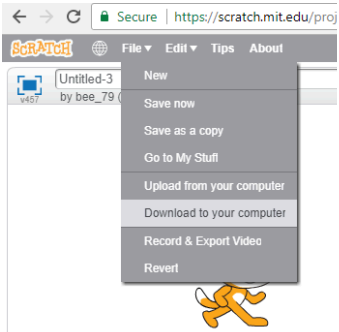
Untuk menyimpan project Anda di Scratch offline editor:



Klik File, kemudian Save.

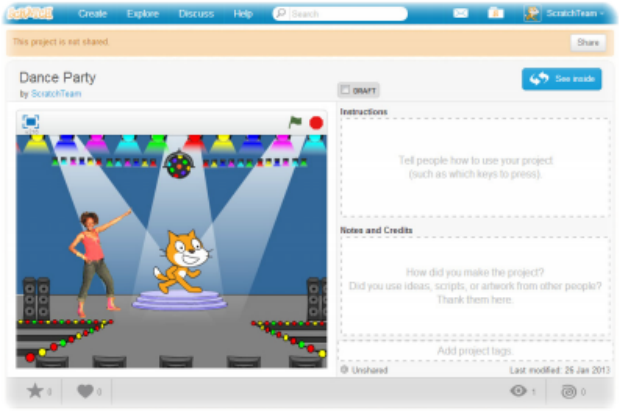
Untuk menyimpan project online Anda, yakinkan bahwa Anda telah Sign In. 

(jika Anda ingin menyimpan file tersebut ke dalam komputer Anda, klik File dan pilih “Download to Your Computer”)



Ketika Anda siap, klik 

Project Page



Klik  untuk menampilkan full screen.

Klik 

agar yang lain melihat dan bermain dengan project Anda.

Ketika Anda berbagi (share), orang lain dapat berkunjung dan berinteraksi dengan project Anda.

Sekarang apa? Anda dapat 

sebuah project baru atau 

ide-ide.

Untuk lebih tahu, klik 

atau kunjungi http://scratch.mit.edu/help .

BAGIAN 3. PEMROGRAMAN SCRATCH

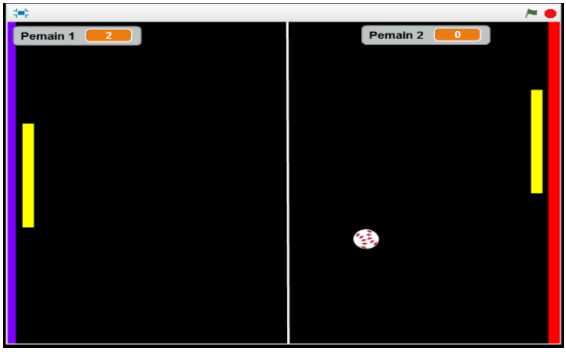
Bagian ini akan menjabarkan pembuatan permainan sederhana berikut dengan logikanya. Permainan pertama akan menampilkan beberapa fungsi pada Scratch misal penghitungan waktu, pergantian latar belakang pada kondisi-kondisi tertentu dan beberapa kegiatan lainnya. Sedangkan contoh permainan berikutnya, menampilkan permainan yang dapat dimainkan oleh dua orang.

Permainan Sederhana

Permainan sederhana yang akan dibuat disini dibagi menjadi dua, yakni: permainan untuk satu orang dan permainan dengan dua orang. Pada permainan yang dilakukan oleh dua orang biasanya setiap pemain saling

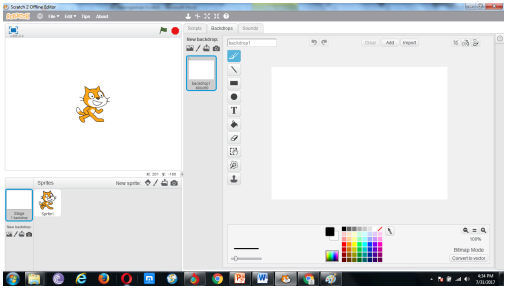
Pong

Permainan ini merupakan permainan yang dapat dilakukan oleh dua orang secara bersamaan, satu lawan satu. Setiap pemain (Pemain 1 dan Pemain 2) dapat mengendalikan tongkat yang digunakan untuk memukul bola, score pemain bertambah apabila dapat menempatkan bola sehingga lawan tidak dapat menjangkaunya. Pemain dengan score tertinggi menang.

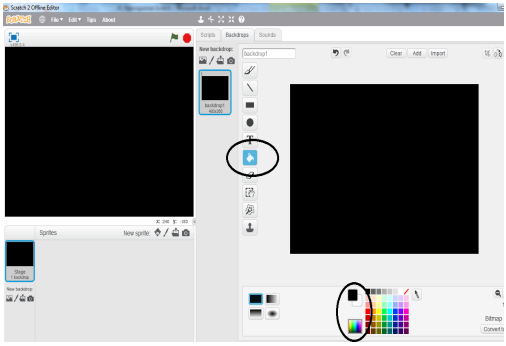


Langkah 1. Membuat Latar Belakang, Papan Score, Bola dan Tongkat.

Membuat Latar Belakang.



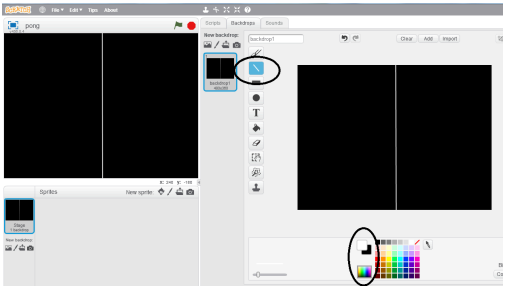
1. Buka Software Scratch.
2. Pilih “Stage”.
3. Plih “Background”



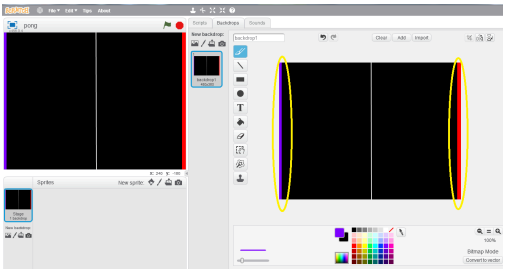
4. Klik kanan pada “sprite 1” dan klik “delete” untuk menghilangkan sprite 1 (kucing).
5. Gunakan “paint bucket” untuk mewarnai.

6. Pilih warna hitam dan klik pada latar belakang.

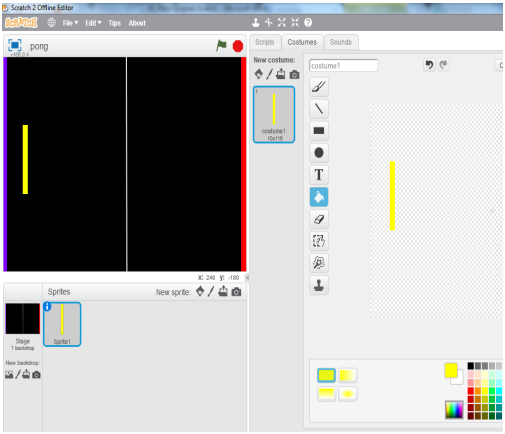
7. Gunakan “line tool” untuk membuat garis putih di tengah latar.

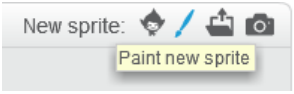


8. Gunakan “rectangle tool” untuk membuat wilayah “score” dengan dua warna yang kontras pada dua sisi yang saling berlawanan (contoh digunakan warna merah dan biru)

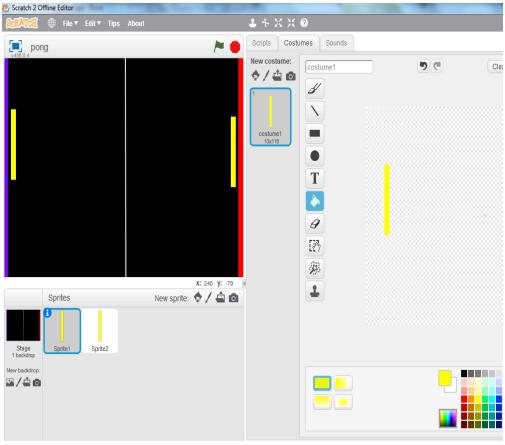


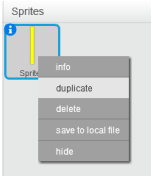
9. Pemukul dibuat pada "sprite" yang baru. Dengan klik pada "Paint new sprite", dan gambar pemukul dengan memilih "Rectangle". Kemudian gambar pemukul.



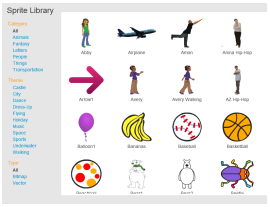


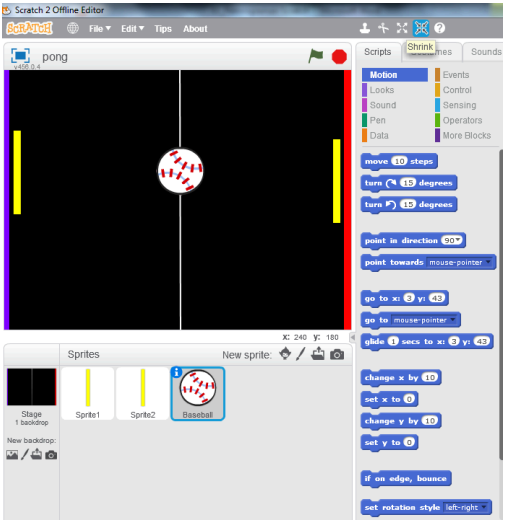
10. Setelah menggambar pemukul, klik kanan pada "sprite" pilih "duplicate" untuk membuat pemukul yang digunakan pemain ke 2.



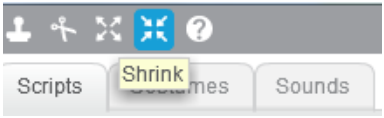


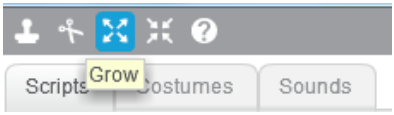
11. Berikutnya membuat bola. Gambar bola dibuat pada “sprite” baru dengan klik “Choose sprite from library”. Kemudian memilih benda yang menurut Anda sesuai.





12. Gambar yang dihasilkan dapat diperbesar atau perkecil sesuai kebutuhan dengan menggunakan fitur “shrink”. Klik kemudian arahkan ke “sprite” yang akan diperkecil dan klik untuk memperkecil



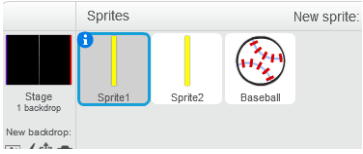


“sprite”. Klik 
kemudian arahkan ke
“sprite” yang akan
diperbesar dan klik
untuk memperbesar.

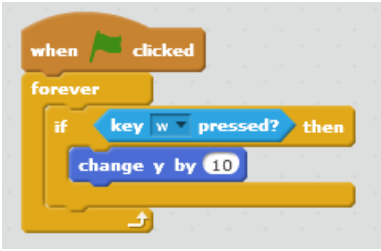
Langkah 2. Membuat program untuk pemukul.

Pada langkah ini kita akan membuat kendali/ kontrol pada pemukul, pemukul pertama dikendalikan oleh pemain pertaman dan pemukul kedua dikendalikan oleh pemain kedua. Kendali dari pemukul berupa gerakan keatas-bawah.

1. Pilih pemukul yang pertama (sprite 1). Sprite 1 nantinya akan dikendalikan oleh pemain pertama.

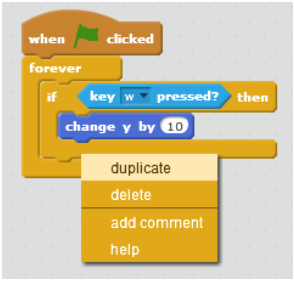


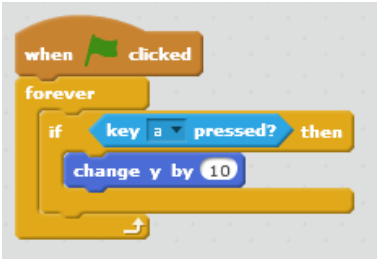
2. Buat script yang memberikan perintah pada pemukul untuk bergerak ke atas apabila ditekan huruf w.



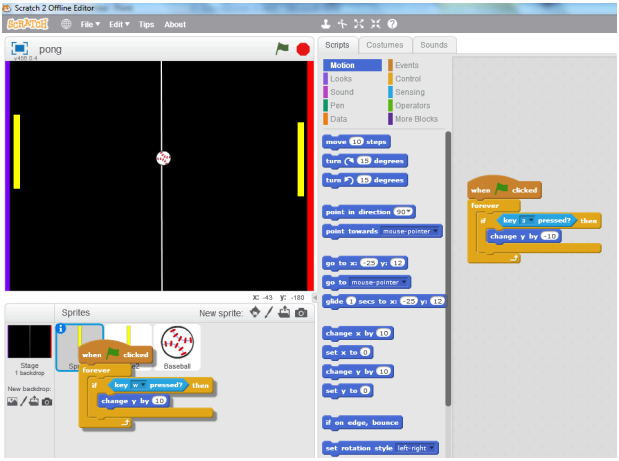
3. Pembuatan script yang memerintahkan pemukul untuk bergerak ke bawah apabila ditekan huruf a, dapat dibuat dengan memodifikasi perintah sebelumnya.

   Pada script klik kanan dan pilih “duplicate”, kemudian ganti huruf w dengan huruf a.





4. Script pada pemukul kedua (sprite 2) dapat dilakukan dengan cara yang hamper serupa, script pada sprite 1 di-drag dan letakkan pada script ke 2. Hal tersebut juga akan membuat duplikasi script tetapi dilakukan pada sprite yang berbeda.

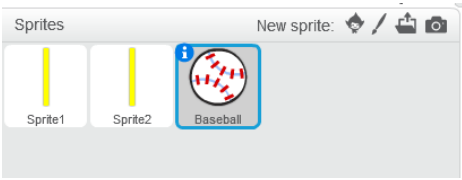


5. Kemudian ubah dari w menjadi up arrow, untuk gerakan ke atas. Dan down arrow untuk gerakan ke bawah.

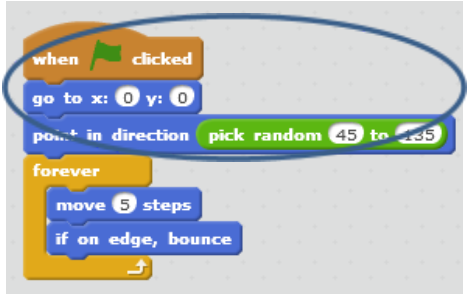
Langkah 3. Membuat program untuk bola

Pada permainan ini, kita akan memprogram bola agar memantul ketika menyentuh tepi. Selain itu, untuk menambah kesulitan dari permainan ini jalur pantulan bola dapat dibuat secara acak.

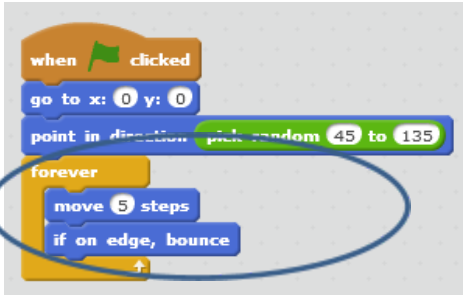
1. Pilih sprite Baseball.



2. Script tersebut memastikan bahwa pada setiap kali permainan dimulai, posisi bola berada di koordinat x=0 dan y=0. Selanjutnya, setelah bola pada posisi tersebut memastikan jalur bola secara acak diantara 45 sampai 135 (derajat).



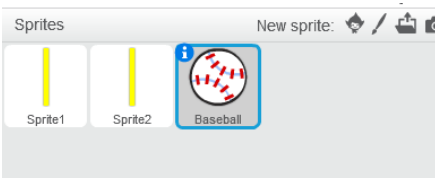
3. Script tersebut memastikan agar bola bergerak secepat 5 langkah dan pada saat mencapai tepi, bola tersebut akan memantul.



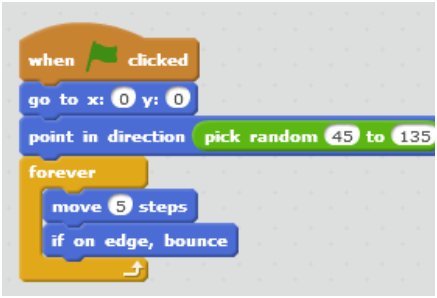
Langkah 4. Membuat program untuk permainan

Pada langkah ini kita memastikan agar pada saat bola menyentuh pemukul maka bola tersebut akan memantul dan permainan dapat dilakukan.

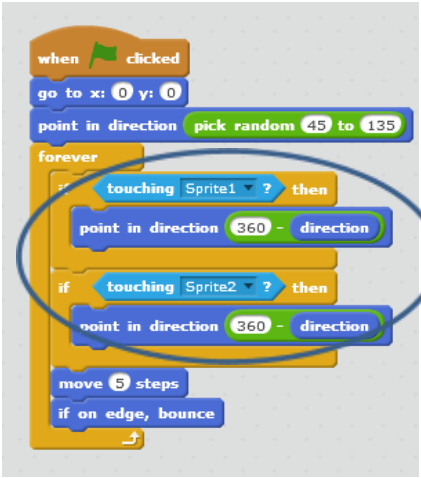
1. Pilih sprite Baseball.



2. Pada script yang telah ada sebelumnya, akan ditambahkan fungsi lainnya seperti memastikan bahwa bola akan memantul juga apabila mengenai pemukul.



3. Script tersebut memastikan bahwa pada saat bola menyentuh sprite 1 dan sprite 2 maka bola akan memantul dengan arah yang berlawanan dengan arah datangnya bola. Kemudian sisipkan script tersebut pada script yang telah dibuat sebelumnya.

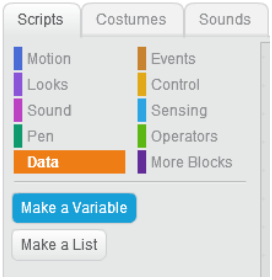


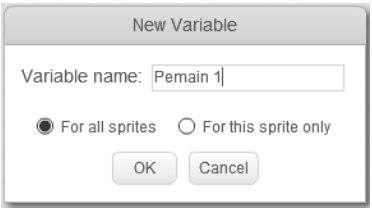
Kemudian cobalah apa yang terjadi….

Langkah 5. Membuat program untuk score

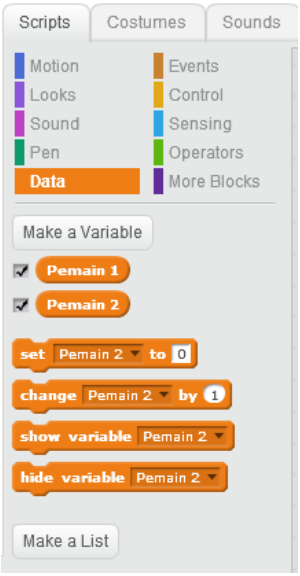
Pada langkah ini kita memastikan bahwa sistem score berjalan dengan baik, sehingga permainan berjalan dengan baik pula.

1. Sistem score dibuat dengan menggunakan variabel yakni: variabel pemain 1 dan pemain 2. Dengan klik “Make a Variable” maka akan muncul kolom “New Variable”, isi “Variable name” dengan Pemain 1 klik “OK”. Kemudian dengan cara yang sama, buat variabel untuk Pemain 2.





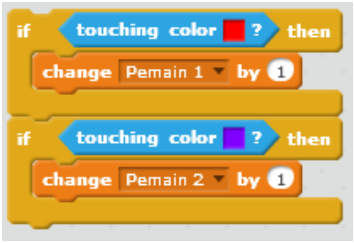
2. Hasil akhirnya di bawah kolom “Make a Variable” akan muncul variabel yang dibuat, yakni: Pemain 1 dan Pemain 2.



3. Script tersebut memastikan bahwa apabila bola menyentuh warna tertentu maka score akan berubah. Agar penentuan warna sesuai dan script berjalan seperti yang

Ingat: Apabila bola menyentuh warna merah maka lawan (Pemain 1) akan mendapat tambahan score, dan sebaliknya.

diharapkan maka pada saat pemilihan warna klik pada warna yang ada di tepi.



4. Sisipkan script tersebut pada script untuk sprite "Baseball" yang telah ada sebelumnya.

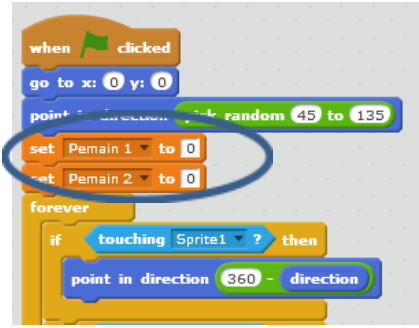
5. Kemudian cobalah apa yang terjadi….

Langkah 6. Pengembangan Program.

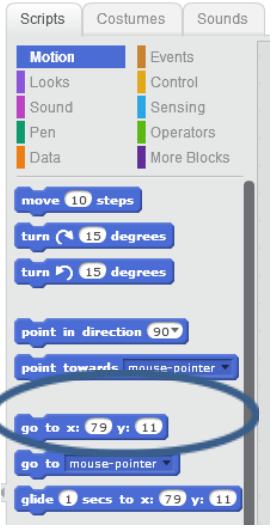
Pada permainan yang telah dibuat masih terdapat beberapa hal yang perlu diperbaiki, misal: sistem “score” tidak kembali dari 0-0 apabila pemainan dimulai dari awal atau bola tidak kembali ke tengah setelah salah satu pemain bertambah angkanya serta beberapa perbaikan lainnya agar permainan semakin menarik.

1. Perbaikan pertama, setiap pemainan dimulai score menunjukkan Pemain 1 :0 dan Pemain 2: 0. Klik pada Data dan pastikan bahwa set Pemain 1 to 0 dan set Pemain 2 to 0.

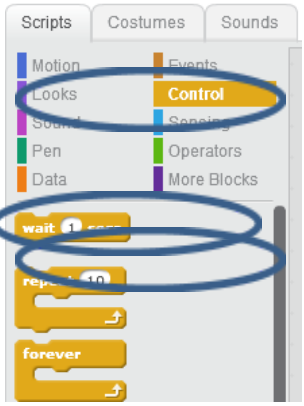
2. Kemudian sisipkan script tersebut pada script yang telah ada. Kemudian cobalah apa yang terjadi….

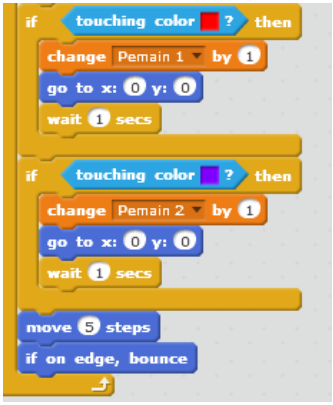


3. Perbaikan berikutnya, kedua, adalah mengembalikan posisi bola ke tengah ((x,y) = (0,0)) setiap salah satu pemain mendapatkan angka. Kemudian sisipkan script tersebut pada script yang telah ada. Kemudian cobalah apa yang terjadi….



4. Perbaikan berikutnya, ketiga, adalah perbaikan sistem score. Pada permainan yang telah dibuat, ternyata masih terdapat sedikit masalah, yakni kadang pada saat bola menyentuh bagian merah (atau biru) permainan mulai terlalu cepat sehingga ada baiknya diberikan sedikit jeda dengan script “wait 1 secs”. Kemudian sisipkan script tersebut pada script yang telah ada. Kemudian cobalah apa yang terjadi….

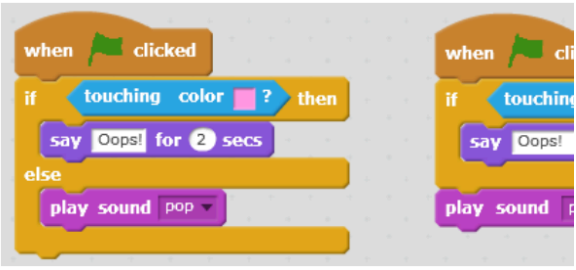




Evaluasi 1

1 Apakah kedua script ini sama?

.

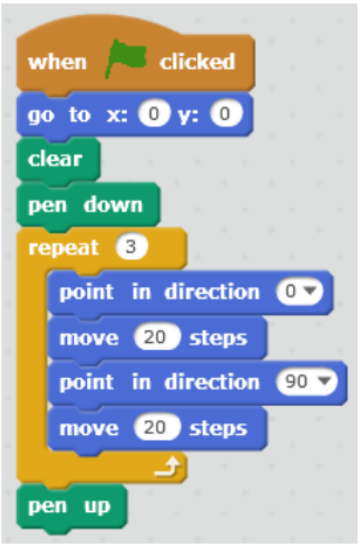


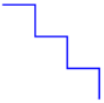
a. Ya

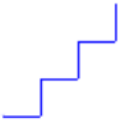
b. Tidak

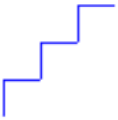
2 Apakah yang akan dibentuk oleh script berikut:

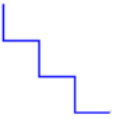
.



a. 

b. 

c. 

d. 

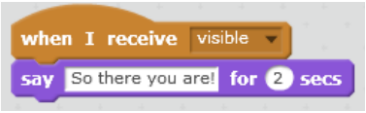
3. Kita akan membuat script dengan dua karakter seekor kucing dan seekor anjing. Script untuk kucing seperti berikut:

Akan dibuat script untuk anjing sedemikian sehingga anjing akan berkata “So there you are!” tepat pada saat kucing muncul dilayar. Manakah script berikut yang sesuai?

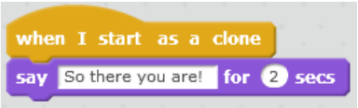
a.

b.

c.

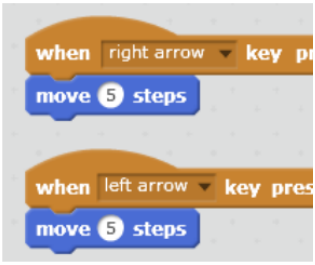


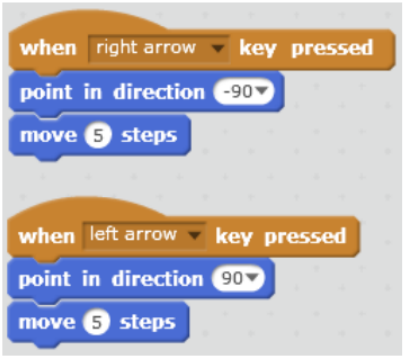
d.



4. Kita akan membubat sebuah permainan baru, dengan menggunakan panah untuk mengendalikannya. Apabila ditekan tombol panah kanan maka karakter pada permainan tersebut akan bergerak ke kanan sejauh 5 langkah, begitu juga sebaliknya apabila ditekan tombol panah kiri. Kita membuat script sebagai berikut, setelah dicoba ternyata terdapat kesalahan.

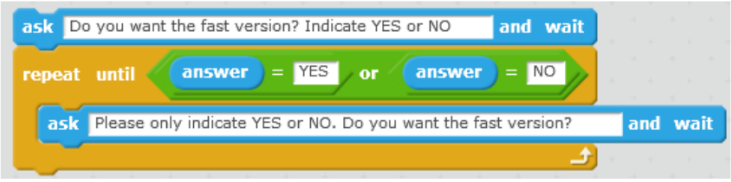
Manakah script dibawah yang dapat berjalan dengan baik.





5 Apa yang dilakukan oleh script berikut:

.



a. Memeriksa apakah jawabannya YES atau NO

b. Memastikan program tidak berlanjut sebelum user menjawab YES atau NO

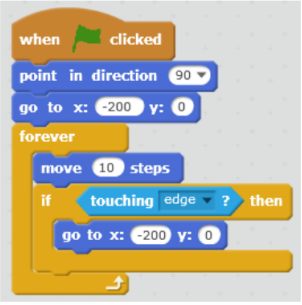
c. Merupakan sebuah cara untuk memverifikasi data yang dimasukkan oleh user

d. Semua jawaban benar

e. Semua jawaban salah

Evaluasi 2

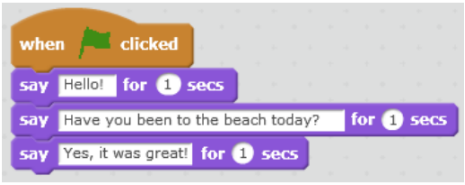
1. Apa yang terjadi apabila kita membuat script seperti di bawah



2. Kita akan membuat animasi berupa dialog antara dua orang.

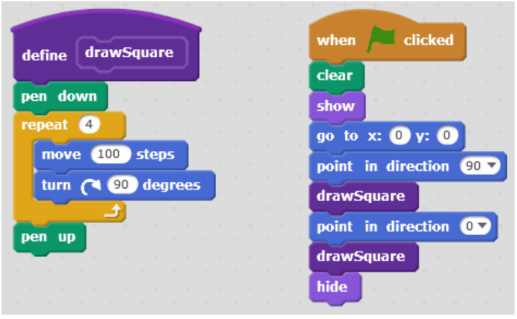
   Script untuk orang pertama:

Script untuk orang kedua:



Apakah kedua script tersebut akan menghasilkan percakapan yang benar?? Apabila terdapat kesalahan, bagaimana memperbaikinya??

3. Apakah hasil dari script di bawah?



Project

1. Dua permainan telah dikonstruksi dan dibahas bagaimana pembuatan dan prosesnya. Sekarang buatlah sebuah permainan.

DAFTAR PUSTAKA

Chapin, N. (1970). Flowcharting With the ANSI Standard: A Tutorial . *Computing Surveys*, 119-146.

Ford, J. L. (2009). *Scratch Programming for Teens.* Boston: Course Technology.

Kindergarten, L. (2013). *Scratch.* Retrieved September 14, 2017, from MIT: https://resources.scratch.mit.edu/www/guides/en/Getting-Started-Guide-Scratch2.pdf

Lero. (2014). *Training.* Retrieved September 15, 2017, from PDST Technology in Education: http://www.pdsttechnologyineducation.ie/en/Training/ICT-in-Classroom-PDFs/An-Introduction-to-Scratch-in-the-Classroom-PDST-Technology-in-Education-Lero-/An-Introduction-to-Scratch-in-the-Classroom-PDST-Technology-in-Education-Lero.pdf

Levitin, A. (2012). *Introduction to The Design and Analysis of Algorithms.* London: Pearson.

Schneider, G. M., & Gersting, J. L. (2010). *Invitation to Computer Science.* Boston: Course Technology.

Biodata Penulis

Abdul Baist. Lahir di Jakarta pada tanggal 8 Mei 1979. Diterima sebagai mahasiswa Pendidikan Matematika di IKIP Bandung pada tahun 1998. Melanjutkan studi Magister Matematika Terapan pada tahun 2009 di Institut Pertanian Bogor. Mengajar di SMK Muhammadiyah 3 Tomang pada tahun 2006 sampai 2009. Saat ini aktif sebagai dosen di Program Studi Pendidikan Matematika Universitas Muhammadiyah Tangerang.

Aji Raditya. Lahir di Jakarta pada tanggal 2 Desember 1986. Diterima sebagai mahasiswa Program Studi Matematika di Institut Pertanian Bogor pada tahun 2004. Melanjutkan studi Magister Pendidikan Matematika Universitas Pendidikan Indonesia di Bandung. Saat ini aktif sebagai dosen di Program Studi Pendidikan Matematika Universitas Muhammadiyah Tangerang.