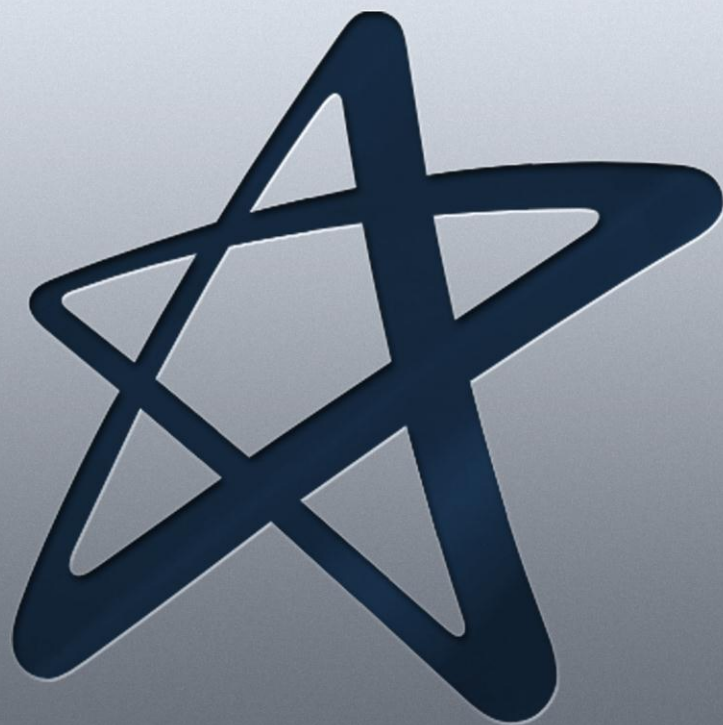


Programação de Computadores



Educação a Distância
Cruzeiro do Sul Educacional
Campus Virtual

Material teórico



Tecnologia em Análise e Desenvolvimento de Sistemas

Responsável pelo Conteúdo:

Prof. Ms. Amilton Souza Martha

Revisão Textual:

Profª Esp. Vera Lidia de Sá Cicaroni

UNIDADE

Tecnologia em Análise e Desenvolvimento de Sistemas



Sejam bem-vindos!

Nesta disciplina, vamos treinar a formalização do pensamento lógico e entender a criação de algoritmos e programas.

Vamos, também, ver um pouco dos conceitos fundamentais que envolvem a programação de computadores e um pouco de história além de conhecer nosso ambiente de programação, que usaremos em todo o decurso desta disciplina: o Scratch.



Atenção

Para um bom aproveitamento do curso, leia o material teórico atentamente antes de realizar as atividades. É importante também respeitar os prazos estabelecidos no cronograma.

Contextualização

Criar um software (ou um programa de computador) não é uma tarefa trivial e envolve conceitos de lógica de programação que se adquirem com muito treino. A formalização de um algoritmo na forma de programa envolve o conhecimento de ferramentas lógicas, como comandos de decisão, comandos de repetição, variáveis, listas, etc.



O aprendizado da lógica de programação para a criação de algoritmos permite-nos programar em qualquer linguagem de programação, já que todas seguem os mesmos princípios lógicos. Sendo assim, vamos, nesta unidade, entender esses termos que envolvem a programação de computadores, como, por exemplo, software, programa, algoritmo, etc., e conhecer o ambiente que usaremos durante toda a disciplina.

Nesta unidade, vamos introduzir você no mundo da programação!!



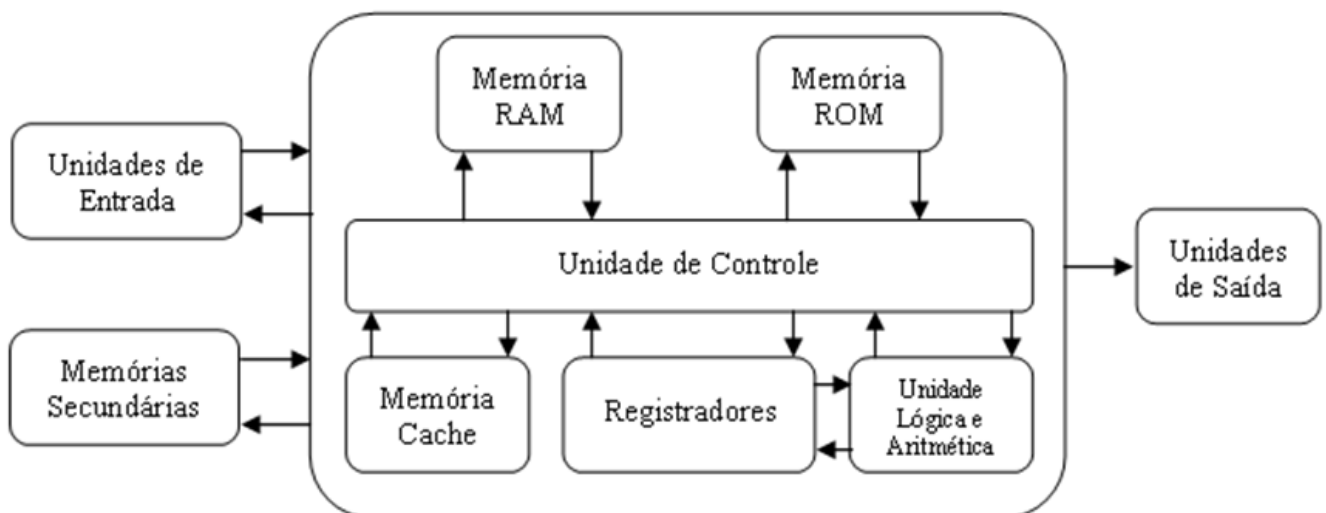
Programação de Computadores

A programação de computadores é uma atividade que leva à representação dos passos necessários à resolução de um problema em linguagem de programação. Para dar início ao aprendizado dessa atividade, é importante compreender seu contexto, seu propósito, os conceitos básicos subjacentes, bem como tomar contato com o ferramental necessário à sua realização.

De acordo com Miyazawa (2001), um computador é uma coleção de componentes que realizam operações lógicas e aritméticas sobre um grande volume de dados. Um computador pode ser dividido em Hardware e Software.

Na Figura 1, podemos visualizar a organização básica de um computador quando se trata de Hardware, que consiste na parte física dos componentes, como periféricos (mouse, teclado, monitor), CPU (processador) e memórias (RAM, ROM, HD).

Figura 1 – Organização Básica de um Computador (Miyazawa, 2001)



Programa de Computador

O complemento do hardware é o software. Um software é que dá instruções para que o hardware execute instruções através de um programa de computador. Um programa de computador é uma coleção de instruções que descrevem uma tarefa a ser realizada por um computador.

O programa é a formalização de um algoritmo em qualquer linguagem capaz de ser transformada em instruções que serão executadas por um computador, gerando os resultados esperados.

A grosso modo, o hardware sem um software é o mesmo que um corpo humano sem a mente, pois o corpo não será capaz de executar nenhuma ação sem que a mente envie comandos para ele.

“It has often been said that a person does not really understand something until he teaches it to someone else. Actually a person does not really understand something until he can teach it to a computer, ie., express it as an algorithm”

Donald Knuth

De forma bem genérica, podemos dizer que construir um programa envolve as seguintes etapas:

- 1) Verificar o problema apresentado em busca de uma solução;
- 2) “Traduzir” essa solução para um algoritmo;
- 3) Implementar esse algoritmo na linguagem escolhida;
- 4) Testar;
- 5) Corrigir erros tanto sintáticos como lógicos;
- 6) Gerar pacote de instalação.

Algoritmo

Em computação, algoritmo pode ser definido como uma sequência de instruções ou operações básicas cuja execução, em tempo finito, resolve um problema computacional. Ele pode ser representado na forma narrativa, graficamente, como um fluxograma ou em pseudocódigo (português estruturado).

Qualquer sequência de passos lógicos que represente a solução de um problema específico é considerada um algoritmo, por exemplo, uma receita de bolo, um manual de instruções de um DVD Player, a maneira pelo qual as contas de água e luz são calculadas ou as regras para o cálculo do seu imposto de renda.

Um algoritmo precisa ter quatro itens importantes. O primeiro é ter fim; o segundo é não dar margens à dupla interpretação, ou seja, deve estar claro qual será o passo seguinte; o terceiro item é gerar informações de saída para o ambiente externo e, por último, é ser efetivo, ou seja, todas as etapas especificadas no algoritmo devem ser alcançáveis em um tempo finito.

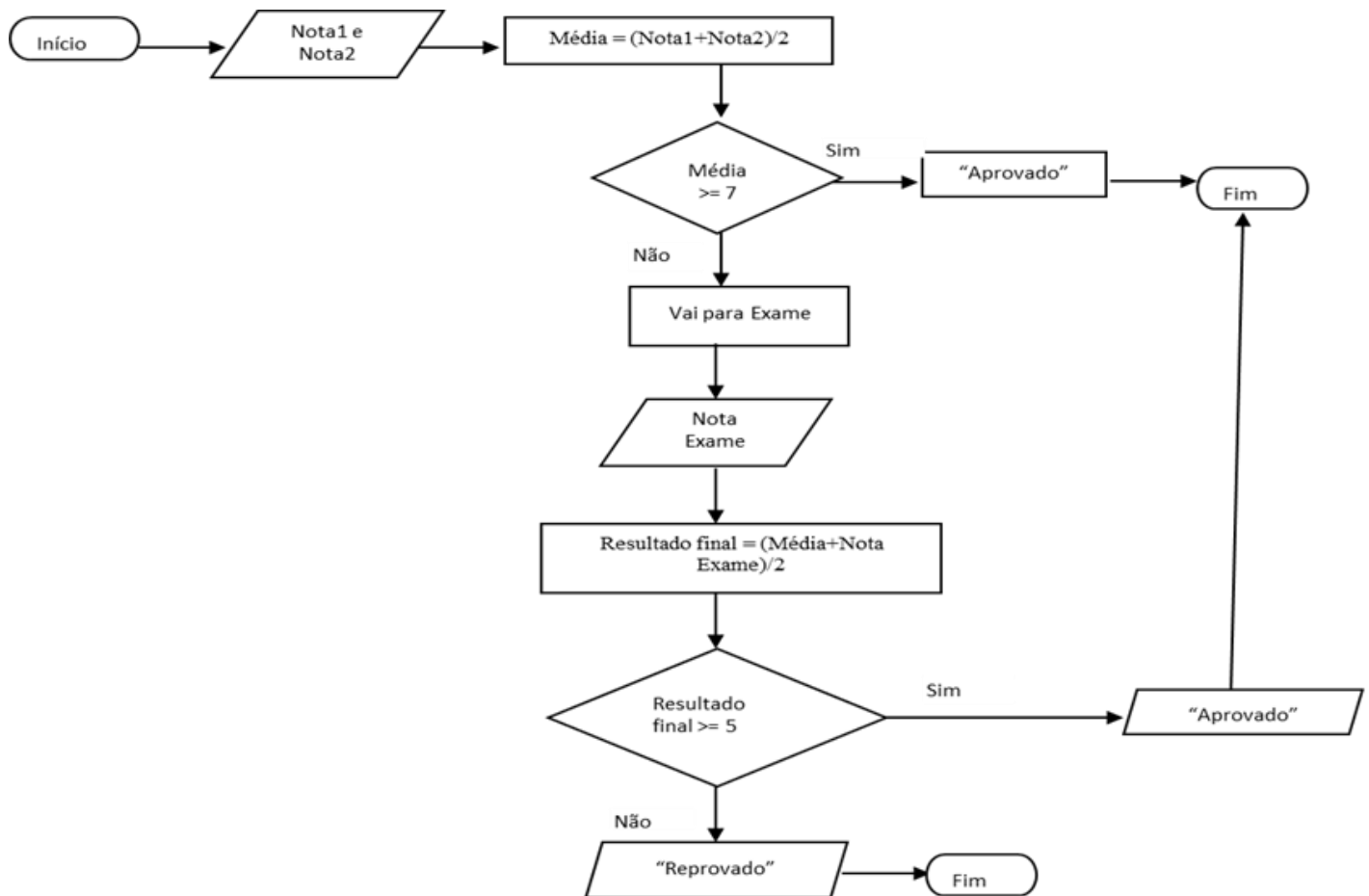
Para entender como criar e representar um algoritmo, vamos tomar um exemplo. Veja o texto a seguir:

“Em uma universidade, a média final de um aluno é calculada pela média aritmética de duas de suas notas. Caso o aluno tenha média final maior ou igual a sete, ele é aprovado diretamente; caso contrário, ele precisará fazer uma prova de exame. Seu resultado final será a média aritmética entre sua média final e a nota do exame e, nesse caso, se o resultado for maior ou igual a cinco estará aprovado; se não, será reprovado”

O texto anterior mostra um problema cuja solução pode ser representada na forma de algoritmo.

Na Figura 2 a seguir, temos o algoritmo do problema representado em forma de Fluxograma

Figura 2 – Exemplo de Algoritmo representado como Fluxograma



O mesmo algoritmo pensado para a resolução do nosso problema de média pode ser representado na forma de pseudocódigo, ou seja, português estruturado, conforme ilustrado na Figura 3. Essa forma assemelha-se muito a uma linguagem de programação formal, porém com comandos em português.

Figura 3 – Exemplo de Algoritmo na forma de Pseudocódigo

```
início
  real media, nota1, nota2, exame, final;
  escreva "Digite a 1ª nota: ";
  leia nota1;
  escreva "Digite a 2ª nota: ";
  leia nota2;
  media ← (nota1 + nota2)/2;
  se (media >= 7)
    escreva "Aprovado";
  senão
    escreva "Digite a nota de exame";
    leia exame;
    final ← (media + exame)/2;
    se (final >= 5)
      escreva "Aprovado";
    senão
      escreva "Reprovado";
    fim se
  fim se
fim
```

História da Programação

Para podermos entender melhor a importância da programação de computadores, vamos voltar um pouco no tempo.

Primeiramente vamos entender que o conceito de computação e programação não está diretamente relacionado aos computadores eletrônicos que conhecemos hoje. Computação é definida como a busca de uma solução para um problema a partir de entradas (inputs) e a obtenção de resultados (outputs), depois de trabalhadas, através de um algoritmo.

Durante milhares de anos, a computação foi executada com caneta e papel ou com giz e ardósia ou mentalmente, por vezes, com o auxílio de tabelas ou utensílios artesanais.

No século XVIII, surgiram os cartões perfurados, que foram o meio de incluir dados e comandos nos computadores por muitos anos.

Em 1890, Hermann Hollerith inventou a máquina de tabular (equipamento com uma espécie de pente metálico em que cada dente estava conectado a um circuito elétrico) utilizada para ler cartões perfurados, os quais são considerados como os precursores da programação que conhecemos hoje.

A partir da segunda metade do século XX, com o advento dos computadores eletrônicos, a Computação passou a ter uma presença cada vez mais marcante na sociedade, influenciando a vida diária de parte da população mundial. A partir da década de 1950, a computação ganhou o status de Ciência, surgindo, então, o termo Ciência da Computação

Antes da década de 50, os programadores criavam códigos diretamente em Linguagem de Máquina: 0110010110011011010110011010111010110101. Assim surgiu a linguagem Assembly, para facilitar a programação de computadores. Apesar de não ser muito simples, era melhor do que ter que usar código de máquina.

Ainda na década de 50, surgiu a Linguagem Fortran, que foi desenvolvida para solucionar alguns problemas encontrados no uso da Assembly. Ela foi considerada uma das melhores da época e é uma linguagem de Alto Nível, o que significa ser uma linguagem mais próxima da do ser humano.

Nos anos 60, surgiu a Linguagem Pascal, bem estruturada e com regras rígidas. Era uma linguagem, até pouco tempo, muito utilizada em cursos de Programação. Ainda nos anos 60, surgiu a Linguagem Basic com o objetivo de familiarizar os iniciantes em programação.

Nos anos 70, foi criada a Linguagem Cobol, usada para a criação e estruturação de bancos de dados. Existem empresas que ainda mantêm sistemas em Cobol, principalmente programas legados de MainFrames. Ainda na década de 70, surgiu a Linguagem C. Pode-se dizer que ela é uma das maravilhas das linguagens de programação. Muitos dos programas existentes hoje foram escritos nessa linguagem.

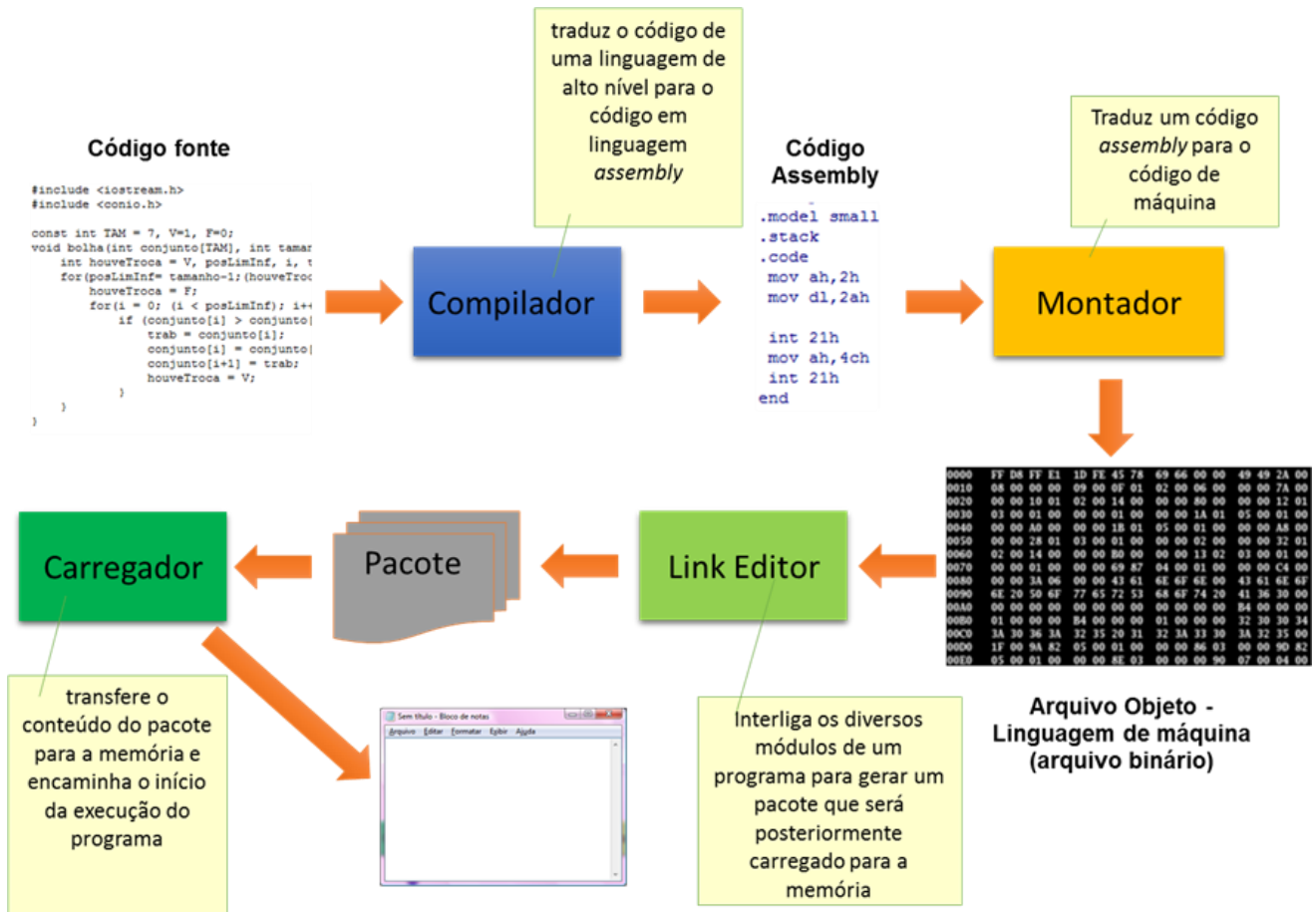
Nos anos 80, C era a linguagem mais utilizada por programadores, por permitir a escrita intensiva de todas as características das linguagens anteriores. O UNIX e o Linux foram escritos em C, assim como o front-end do MS-DOS, Windows e as aplicações Office mais usadas no mundo (OpenOffice.org, Microsoft Office, embora cada uma delas incluísse suas próprias linguagens de script). Depois do C, foi desenvolvida a C++, que é o C orientado a objetos. Desde sua criação, ela passou por atualizações e padronizações até que foi aprovado pelo ANSI. Vários projetos como o KDE (front-end para UNIX, Linux, BSD e recentemente para Windows) são escritos em C++.

Na década de 90, surgiu um aperfeiçoamento da Linguagem Basic, o Visual Basic da Microsoft, que rapidamente foi adotada pelos programadores por ser de fácil entendimento e uso. Ainda na década de 90, surgiu a Linguagem Java, que nasceu já condenada, mas eis que a World Wide Web explodiu em popularidade e a Sun (desenvolvedora do Java) viu uma oportunidade de desenvolver a linguagem para ser aplicada à Internet e a dispositivos móveis, entre outras coisas. Ela é uma linguagem totalmente Orientada a Objetos.

Recentemente houve o surgimento de novas linguagens como aperfeiçoamento de outras antigas, pensando no uso do paradigma de Orientação a Objetos; as mais populares, além do Java, são as linguagens da família .NET (C#, VB.NET, J#, etc.)

Veja, na Figura 4, os passos para a criação de um programa de computador.

Figura 4 – Passos para a criação de um programa



Linguagem Compilada x Interpretada

Uma linguagem de programação pode ser convertida, ou traduzida, em código de máquina por compilação ou interpretação, dois processos que, juntos, podem ser chamados de tradução.

O dicionário da língua portuguesa define compilar, do latim *compilare*, como v. tr. reunir; ajuntar.

Um compilador converte o código fonte em código de máquina somente depois de não haver mais erros de sintaxe. Isso permite que o programa seja executado tantas vezes quantas necessárias sem necessidade de recompilação.

Alguns exemplos de linguagens de programação compiladas são C, C++, Pascal, Cobol, etc.

A definição de Interpretar, do latim *interpretare*, é tornar claro o sentido de; explicar; traduzir; fazer juízo a respeito de. No caso de linguagem interpretada, o código fonte é

traduzido à medida que vai sendo executado. Sempre que é executado novamente, o código fonte passa pelo interpretador, verificando sempre erros de sintaxe, o que, geralmente, torna esse programa mais lento que os programas compilados.

As linguagens interpretadas também são conhecidas como linguagens de script e alguns exemplos são Javascript, Python, Perl, PHP, ASP, Prolog, etc.

Reunindo a vantagem de linguagens compiladas e interpretadas, hoje temos linguagens mistas, que são compiladas para um código de máquina virtual e que precisam ser interpretadas por essa máquina virtual. Temos, nessa categoria, as linguagens Java e .NET.

No caso de .NET, os programas são desenvolvidos em uma linguagem da plataforma .NET como C#, VB.NET ou outra qualquer, compilados para bytecode em uma linguagem intermediária (MSIL code) e interpretados pela máquina virtual (.NET Framework).

No caso de Java, os programas são compilados para bytecode, gerando uma linguagem intermediária (.class), e interpretados pela máquina virtual Java (JVM).

Scratch

Dependendo da linguagem de programação, o código pode ser escrito até no Bloco de Notas, mas existem ambientes chamados IDEs (*Integrated Development Environment* ou Ambiente Integrado de Desenvolvimento) que facilitam o desenvolvimento do programa devido à interface amigável que o compõe.

Alguns exemplos:

Para Java:

JBuilder, JCreator, Visual J++, Eclipse, NetBeans;

Para C++:

Dev C++, Turbo C++, C++ Builder, Visual C++;

Para Javascript:

JSide, Dreamweaver;

O desenvolvimento do raciocínio lógico, o uso de ferramentas de computação, como variáveis, comandos de decisão, comandos de repetição, listas e outras estruturas das linguagens de programação, não dependem de uma linguagem específica e estão presentes em todas elas.

No nosso curso, vamos usar o ambiente Scratch para o aprendizado da lógica de programação sem ficarmos presos a uma sintaxe rígida de uma linguagem formal de programação.

Ao final deste curso, os alunos serão capazes de começar o desenvolvimento em uma linguagem de programação formal como Java, C#, PHP ou qualquer outra linguagem comercial, pois os conceitos essenciais de lógica de programação já estarão fundamentados.

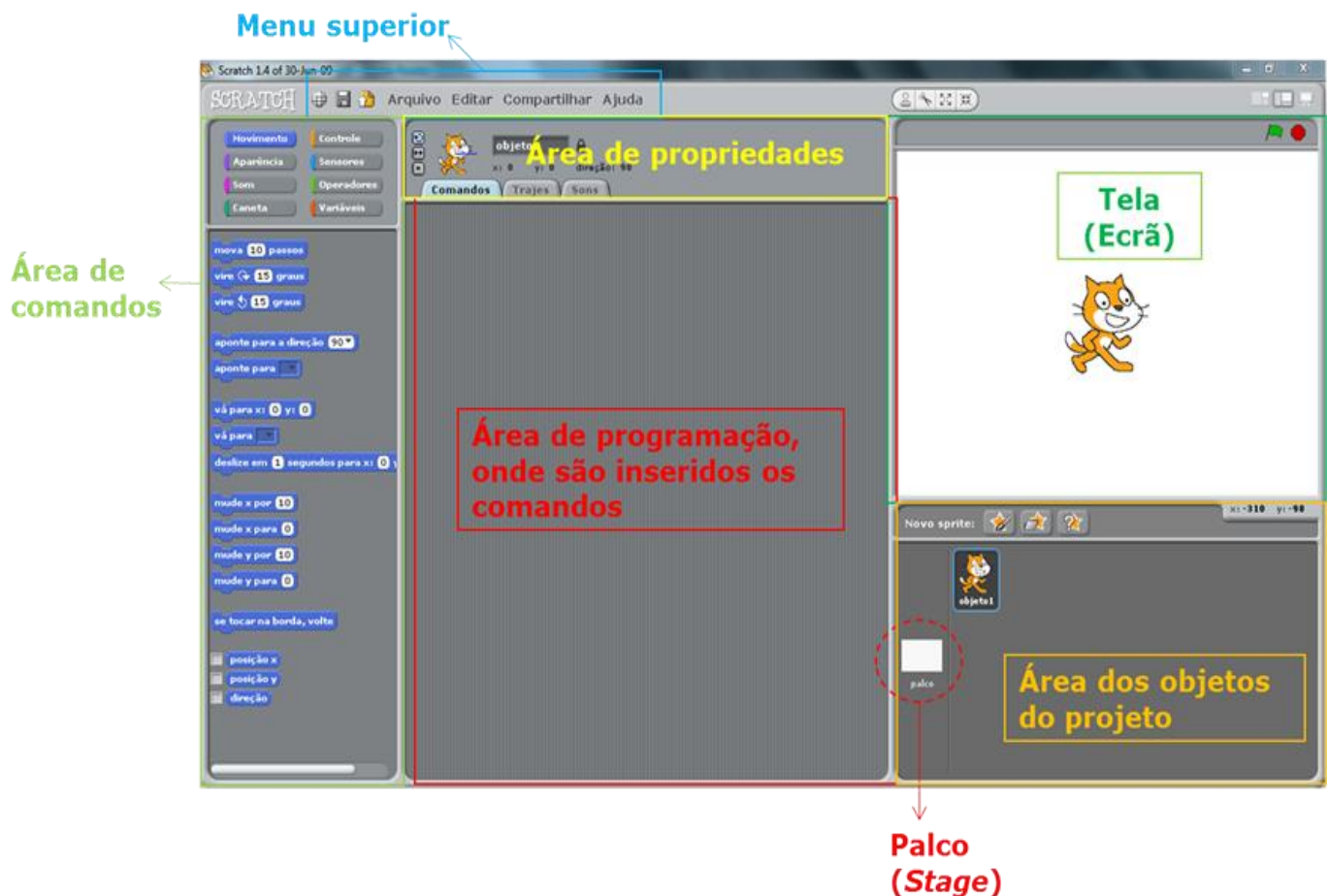


Scratch é uma nova linguagem de programação na forma visual desenvolvida pelo *Lifelong Kindergarten Group* no MIT Media Lab (<http://llk.media.mit.edu>) em 2007, com o apoio financeiro da *National Science Foundation*, Microsoft, *Intel Foundation*, Nokia e consórcios de investigação do MIT Media Lab.

A programação é efetuada através da criação de sequências de comandos simples que correspondem a blocos de várias categorias, encaixados e encadeados de forma a produzirem as ações desejadas. O Scratch pode ser baixado no endereço <http://scratch.mit.edu/download>.

Veja, na Figura 5, a tela do Scratch e seus principais componentes.

Figura 5 – Tela Inicial do Scratch



Na área de comandos, encontramos vários tipos de instruções agrupadas como sendo de Movimento, Aparência, Som, Caneta, Controle, Sensores, Operadores ou Variáveis. Conforme fomos avançando na programação, iremos conhecendo cada um dos itens.

Como foi dito, temos um grupo de 8 categorias de comandos, conforme pode ser visto na Figura 6. Elas são:

Figura 6 – Categorias de Comandos Scratch



Movimento: determina o movimento do Sprite, fazendo-o se movimentar dentro da área da Tela.

Aparência: serve principalmente para substituição dos trajes, fazer aparecer ou desaparecer ou, ainda, fazer que apareçam diálogos.

Som: tem como principal finalidade importar sons ou músicas.

Caneta: responsável pelos traços deixados pelo objeto que está se movimentando, podendo ser modificadas cor, espessura e tonalidade.

Controle: possui comandos pré-definidos, responsáveis pela estrutura lógica de conexão entre outros comandos.

Sensores: serve para perceber cores, distâncias e são, normalmente, combinados com outros comandos.

Operadores: serve para fazer operações matemáticas entre outros.

Variáveis: serve para criar variáveis para armazenar um determinado valor para que possa ser usado posteriormente assim como também a questão de criação de listas.

Para inserir um comando, basta selecioná-lo e arrastá-lo para a área de programação. Vamos criar um programa que apresente uma mensagem e faça a leitura de dados fornecidos pelo usuário. O algoritmo em pseudocódigo ficará assim:

Início

caracter resposta

escreva “Sejam bem vindos ao mundo da programação”

escreva “Digite seu nome”

leia resposta

escreva resposta

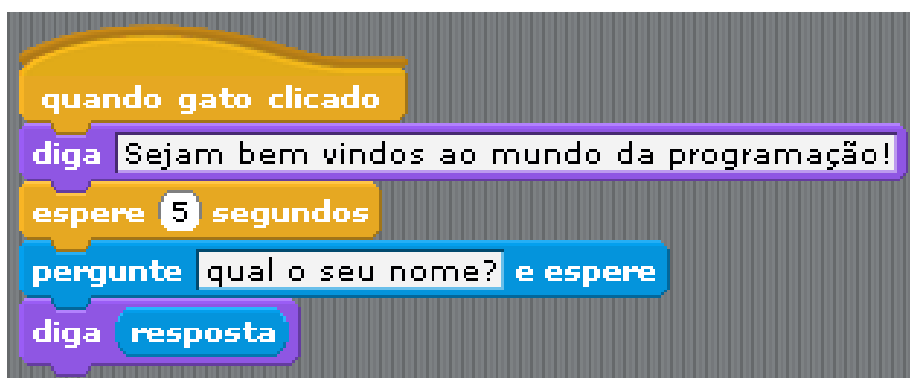
Fim

Vamos transformar nosso algoritmo, em pseudocódigo, em programação Scratch seguindo os seguintes passos:

- 1) Abrindo o ambiente, já há um projeto padrão. Por enquanto vamos utilizá-lo.
- 2) Vamos renomear o objeto1 para gato. Para isso, basta clicar sobre o nome dele na área de propriedades.
- 3) Clique na opção “Controle” da área de comandos; escolha e arraste para a área de programação o comando “Quando gato clicado”.
- 4) Clique na opção “Aparência” da área de comandos; escolha e arraste o comando “Diga <olá>” e encaixe no anterior.
- 5) Para mudar a mensagem, clique sobre ela e escreva a frase “Sejam bem-vindos ao mundo da programação”
- 6) Vá até a aba “Controle” e arraste o comando “espere <1> segundo” abaixo do comando anterior. Altere o valor de <1> para <5> segundos.
- 7) Clique na aba “Sensores” e arraste o comando “Pergunte <qual o seu nome?> e espere”
- 8) Vá até a aba “Aparência” e arraste “diga <Olá!>” para a área de programação
- 9) Volte à aba “Sensores” e arraste o comando “resposta” para o lugar da palavra <Olá!>




A Figura 7 mostra o programa final em Scratch. Para executá-lo, basta clicar no gato e os comandos começarão a rodar.

Figura 7 – Algoritmo no Scratch

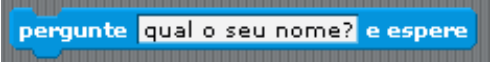



À medida que formos avançando na disciplina, vamos conhecendo mais ferramentas do Scratch. Veja alguns exemplos:


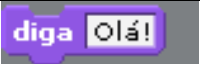
★ ABA CONTROLE:

	Os blocos com os topos arredondados são colocados no início dos algoritmos e indicam o evento que fará correr o programa. Nesse caso, quando a bandeira verde for clicada.
	Faz parte dos blocos citados acima e, nesse caso, o evento que dispara o programa é o clique no objeto gato.
	Faz com que o programa fique parado durante uma quantidade de segundos, que pode ser alterada.

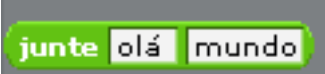
★ ABA SENSORES:

	Esse bloco exibe uma pergunta na tela e abre uma caixa de diálogo para que o usuário digite a resposta.
	A resposta digitada no item anterior é armazenada nessa variável.

★ ABA APARÊNCIA:

	Mostra uma mensagem de diálogo do objeto durante um período de tempo, que pode ser configurado.
	Mostra uma mensagem de diálogo do objeto.

★ ABA OPERADORES:

	Concatena (ou combina) duas Strings na mesma mensagem.
---	--

Material Complementar

1) Sites da Wikipédia:

<http://pt.wikipedia.org/wiki/Algoritmo>

http://pt.wikipedia.org/wiki/Programa_de_computador

2) Site oficial do Scratch:

<http://scratch.mit.edu/>

3) Apostila em português do Scratch:

http://oficinas.pensamentodigital.org.br/apostila_iniciacao_programacao.pdf

4) Manual de Referência do Scratch

http://info.scratch.mit.edu/Support/Reference_Guide_1.4

5) Livros disponíveis na biblioteca



FORBELLONE, A. L. V.; EBERSPACHER, H. F. **Lógica de Programação: a Construção de Algoritmos e Estrutura de Dados.** 3. ed. São Paulo: Makron Books do Brasil, 2005.

MANZANO, J. A. N. G. **Algoritmos: lógica para desenvolvimento de programação.** 20. ed. São Paulo: Erica, 2007.

VILARIM, G. O. **Algoritmos: programação para Iniciantes.** Rio de Janeiro: Ciência Moderna, 2004.



Referências

Scratch: <http://scratch.mit.edu/>

MIYAZAWA, Flávio K. **Nota de Aula de Algoritmos e Programação de Computadores**. Instituto de Computação, Unicamp, 2001.

http://pt.wikipedia.org/wiki/Programa_de_computador

<http://pt.wikipedia.org/wiki/Algoritmo>

Anotações

[illegible]



Educação a Distância

Cruzeiro do Sul Educacional

Campus Virtual

www.cruzeirodosulvirtual.com.br

Campus Liberdade

Rua Galvão Bueno, 868

CEP 01506-000

São Paulo SP Brasil

Tel: (55 11) 3385-3000

