

288S - Linguagem de Técnicas de Programação

Prof. MSc. Bruno Iran Ferreira Maciel

Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas

IESO
INSTITUTO DE ENSINO
SUPERIOR DE OLINDA

faccor
Olinda

**Olinda
facir**



29 de agosto de 2019

UNIDADE I

UNIDADE II

- 1 Introdução e motivação
 - 2 Introdução a programação
 - 3 Diagrama de Blocos
 - 4 Introdução à lógica
 - 5 Tipos de Dados e Instruções Primitivas
 - 6 Introdução à linguagem c
 - 7 Ambiente de desenvolvimento
-

Unidade I

Conteúdo da prova I

Introdução e motivação

Apresentação pessoal, integração com a turma, introdução de conceitos básicos de programação de computadores e despertar curiosidade dos alunos sobre o tema.

Datas importantes

- xx/xx/2019 - Simulado da Primeira Avaliação (1A)
- **xx/xx/2019 - Primeira Avaliação**
- xx/xx/2019 - Apresentação dos resultados da 1A
- xx/xx/2019 - Simulado da Segunda Avaliação (2A)
- **xx/xx/2019 - Segunda Avaliação**
- xx/xx/2019 - Apresentação geral dos resultados
- **xx/xx/2019 - Segunda Chamada**

Compromisso semanal

- Encontros: quintas e sextas
- Período: 08/08/2019 à 20/12/2019
- Início: 19h
- Térmico: 21h30m
- Intervalo: 20h10m até 20h20m (a discutir)
- Sala: 28
- Cada dia são três aulas:
 - 19h-19h50m
 - 19h50m-20h40m
 - 20h40m-21h30m

- Resolução de dúvidas gerais: 19h até 19h20m (20min)
- Revisão da aula passada: 19h20m até 19h40m (20min)
- Exercício em sala de aula: 19h40m até 19h50m (10min)
- Adição de novo conteúdo: 19h50m até 21h20m (1h20m)
- Revisão/dúvidas do novo conteúdo: 21h20m até 21h30m (10m)

- Primeira nota: (exercício + simulado + prova)
 - Exercícios (peso +2)
 - Simulado (peso +1)
 - Prova escrita (peso 10)
- Segunda nota: (exercício + simulado + prova + a definir)
 - Exercícios (peso +1)
 - Simulado (peso +1)
 - Prova escrita (peso a definir)
 - Projeto (peso a definir)

Código da turma **92oykpw**

Vamos ter como recurso complementar o Google Sala de Aula [classroom]. Por lá vou deixar o material (slides, referências, atividades, etc.) que vocês vão utilizar durante o curso. Não preocupem-se, também vou deixar uma cópia do material (slides e atividades) disponível em pendrive e levarei comigo durante as aulas para os alunos que não puderem acessar o Google Classroom.

Como usar o google classroom?

Use este tutorial <https://www.youtube.com/watch?v=l4oSdhLS5fQ> [vídeo] para te ajudar a entender o Google Classroom.

Clique <https://classroom.google.com> para entrar no Google Classroom.

- SOFFNER, R. Algoritmos e programação em linguagem C. Saraiva, 2013.
- MIZRAHI, V. V. Treinamento em linguagem C. 2.ed. São Paulo Pearson.2008.
- PEREIRA, S. L. Algoritmos e lógica de programação em C: uma abordagem didática. São Paulo, Érica, 2010.
- Mais informações é possível encontrar na ementa.

Introdução a programação

Objetivo principal:

- **Reconhecer algumas estruturas de programação** - sequências, ciclos (loops), condições, através da exploração e desafios.
- **Compreender o conceito de algoritmo** - sequência de instruções.

Vocabulário

- **sequência** - ordenação de fatos ou ações;
- **ciclo (loop)** - sequência de ações, fatos constituintes de um processo periódico que, partindo de um ponto inicial, acabam em um ponto-final que nada mais é que o retorno a esse ponto inicial e consequente recomeço;
- **código** - regras usadas para converter instruções ou dados de uma forma para outra; e
- **condição** - circunstância necessária para que ocorra determinado fato ou situação.

Perguntas para a participação da classe:

- O que significa “repetir” um “bloco” de código?
- Quando você deve usar um bloco “se”?

Discussão com o parceiro de equipe:

- Você consegue pensar em um motivo pelo qual gostaria de usar um bloco se/senão, em vez de usar apenas o “se”?

Objetivo principal:

- apresentar o modelo de “pensamento computacional” como uma forma de preparar problemas do mundo real para a representação digital.

Frequentemente, os cientistas da computação descobrem que são responsáveis por programar soluções para coisas com as quais as pessoas sequer já sonharam – coisas que nunca foram criadas. Enfrentar um problema que nunca foi solucionado antes pode ser assustador, mas com essas simples ferramentas, tudo é possível.

Para ter sucesso nesse tipo de solução, nós vamos praticar um método chamado Pensamento Computacional. O pensamento computacional se baseia em quatro etapas para ajudar a resolver vários tipos de problemas diferentes.

- **Etapa 1)** Decomposição – Nós não estamos falando de zumbis! Estamos falando de transformar um problema grande e difícil em algo muito mais simples. Geralmente, problemas grandes são apenas diversos problemas pequenos que foram unidos.
 - **Etapa 2)** Padrões – Normalmente, quando um problema tem muitas partes menores, você perceberá que essas partes têm algo em comum. Se não tiverem, elas poderão, pelo menos, ter algumas semelhanças evidentes em relação a algumas partes de outro problema solucionado anteriormente. Se conseguir identificar esses padrões, compreender as partes ficará muito mais fácil.
-

- **Etapa 3)** Abstração – Depois de reconhecer um padrão, você poderá “abstrair” (ignorar) os detalhes que são responsáveis pelas diferenças e usar a estrutura geral para encontrar uma solução que seja válida para mais de um problema.
- **Etapa 4)** Algoritmo – Quando sua solução estiver completa, você poderá escrevê-la de um modo que ela possa ser processada passo a passo, para que seja fácil atingir os resultados.

Vocabulário

- **Pensamento Computacional** - um método de resolução de problemas que ajuda cientistas da computação a preparar problemas para soluções digitais;
- **Abstração** - ação de ignorar os detalhes de uma solução de modo que ela possa ser válida para diversos problemas;
- **Algoritmo** - uma lista de etapas que permitem que você complete uma tarefa;
- **Decompor** - dividir um problema difícil em problemas menores e mais fáceis;
- **Padrão** - um tema que se repete diversas vezes; e
- **Programa** - instruções que podem ser compreendidas e seguidas por uma máquina.

Perguntas para a participação da classe:

- Você consegue nomear alguma das etapas do pensamento computacional?
- Você consegue se lembrar de algum dos padrões que encontramos nos exemplos?

Discussão com o parceiro de equipe:

- O que mais poderíamos descrever com os mesmos conceitos “abstraídos”? Seria possível descrever uma vaca? Um pássaro?

Diagrama de Blocos

Ferramenta usada e pelos profissionais da área de análise de sistemas.

- Tem por objetivo descrever o fluxo de ação de um determinado trabalho lógico, seja manual ou mecânico, especificando os suportes usados para os dados e para as informações.
- Pode ser feito em qualquer nível de abstração.
- Utiliza símbolos convencionais (norma ISO 5807:1985) e permite poucas variações. São desenhos geométricos básicos.

Também conhecido como diagrama de fluxo (não confundir com fluxograma), é uma ferramenta usada pelo profissional de desenvolvimento de programas, seja ele o programador e afins.

- Tem por objetivo descrever o método e sequência de ações a serem estabelecidas para um computador.
- Pode ser feito em qualquer nível de abstração.
- Utiliza símbolos geométricos, os quais estabelecerão as sequências de operações a serem efetuadas em processamento computacional.

Diagrama de Bloco + Codificação

Modo de solucionar um problema por meio um algoritmo.

Segundo [Manzano and de Oliveira(2008)], muitas pessoas gostam de falar ou julgar que possuem e sabem usar o **raciocínio lógico**, porém quando questionadas direta ou indiretamente, perdem essa linha de raciocínio, pois ele depende de inúmeros fatores para completá-lo, tais como: calma, conhecimento, vivência, versatilidade, experiência, criatividade, ponderação, responsabilidade, autodisciplina, entre outros.

Para usar a lógica, é necessário ter domínio sobre o pensamento, bem como saber pensar, ou seja, possui a “Arte do pensar”. Alguns definem o raciocínio lógico como um conjunto de estudos que visa determinar os processos intelectuais que são as condições gerais do conhecimento verdadeiro. Outros preferem dizer que é a sequência coerente, regular e necessária de acontecimentos, de coisas ou fatos, ou até mesmo, que é a maneira do raciocínio particular que cabe a um indivíduo ou a um grupo.

Logo, pode-se entender que lógica é a ciência que estuda as leis e critérios de validade que regem o pensamento e a demonstração, ou seja, ciência dos princípios formais do raciocínio.

Necessidade de usar a lógica

Usar a lógica é um fator a ser considerado por todos, principalmente pelos profissionais da área da Tecnologia de Informação (programadores, analistas de sistemas e suporte), pois seu dia-a-dia dentro das organizações é solucionar problemas e atingir os objetivos apresentados por seus usuários com eficiência e eficácia, utilizando recursos computacionais e/ou automatizados mecanicamente. Saber lidar com controle, de planejamento e de estratégia requer atenção e boa performance de conhecimento de nosso raciocínio. Porém, é necessário considerar que ninguém ensina ninguém a pensar, pois todas as pessoas normais possuem esse “dom”.

Necessidade de usar a lógica



O objetivo aqui é mostrar como desenvolver e aperfeiçoar melhor essa técnica, lembrando que para isso você deve ser persistente e praticá-la constantemente, chegando à exaustão sempre que julgar necessário.

É comum que profissionais da área de Tecnologia da Informação (desenvolvimento) prefiram preparar um programa iniciando o seu projeto com um para demonstrar de forma concreta sua linha de raciocínio lógico (que é um elemento abstrato).

O diagrama de blocos é um instrumento que visa estabelecer visualmente a sequência de operações a ser efetuada por um programa de computador.

Aplicabilidade da lógica no auxílio do desenvolvimento de programas



A técnica de desenvolvimento de diagrama de blocos permite ao desenvolvimento uma grande facilidade na posterior codificação do programa em qualquer uma das linguagens de programação existentes, pois elaboração não se leva em que uma linguagem utiliza.

O diagrama de blocos é uma ferramenta que possibilita definir o detalhamento operacional que um programa deve executar sendo um instrumento tão valioso quanto é uma planta para um arquiteto.

A técnica mais importante no projeto da lógica de programas denomina-se programação estruturada, a qual consiste em uma metodologia de projeto, objetivando:

- Agilizar a codificação da escrita da programação;
- Facilitar a depuração da sua leitura;
- Permitir a verificação de possíveis falhas apresentadas pelo programas;
- Facilitar as alterações e atualizações dos programa.

Aplicabilidade da lógica no auxílio do desenvolvimento de programas



E deve ser composta por quatro passos fundamentais:

- Escrever as instruções em sequências ligadas entre si apenas por estruturas sequenciais, repetitivas ou de selecionamento;
- Escrever instruções em grupos pequenos e combiná-las;
- Distribuir módulos do programa entre os diferentes programadores que trabalharão sob a supervisão de um programador sênio, ou chefe de programação;
- Revisar o trabalho executado em reuniões regulares e previamente programadas, em que compareçam apenas programadores de um mesmo nível.

Introdução à lógica

Primeiramente é preciso entender e compreender a palavra “problema”. Pode-se dizer que problema é uma proposta duvidosa, que pode ter numerosas soluções, ou questão não resolvida e que é objetivo de discussão.

Problema é uma questão que foge a uma determinada regra, ou melhor, é o desvio de um percurso, o qual impede de atingir um determinado objetivo com eficiência e eficácia.

Os diagramas de blocos são realmente um bom instrumento para avaliação do problema do fluxo de informações de um dado sistema. Por esse motivo deve-se resolver um problema de lógica usando um procedimento de desenvolvimento.

Princípios para uso do diagrama de blocos:

- 1 Os diagramas de blocos devem ser feitos e quebrados em vários níveis. Os primeiros diagramas devem conter apenas as ideias gerais, deixando para as etapas posteriores os demais detalhes.
- 2 Para o desenvolvimento correto de um diagrama de bloco, ele deve ser iniciado de cima para baixo.
- 3 É incorreto e “proibido” ocorrer o cruzamento de linhas de fluxo de dados entre os símbolos.

Uma faculdade qualquer, cujo cálculo da média é realizado com três notas que determinam a aprovação ou reprovação dos seus alunos. Há pesos diferentes para as notas, as duas primeiras notas tem peso 4, logo a última tem peso 2. Considere ainda que o valor da média deve ser maior ou igual a 7 para que haja aprovação.

Exemplo

Uma faculdade qualquer, cujo cálculo da média é realizado com três notas que determinam a aprovação ou reprovação dos seus alunos. Há pesos diferentes para as notas, as duas primeiras notas tem peso 4, logo a última tem peso 2. Considere ainda que o valor da média (ponderada) deve ser maior ou igual a 7 para que haja aprovação.

$$\text{Média final} = \frac{\text{Nota1} * 4 + \text{Nota2} * 4 + \text{Nota3} * 2}{10}$$

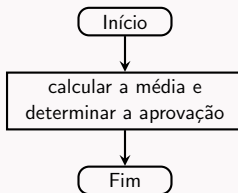


Figura: Diagrama de blocos para o cálculo da média.

Exemplo

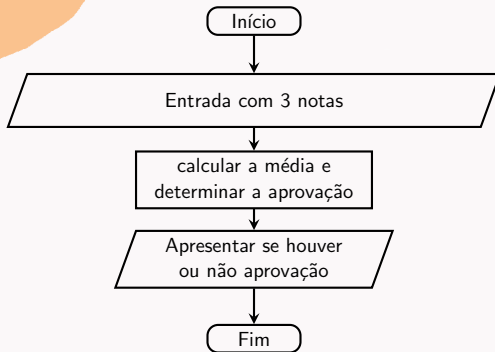


Figura: Diagrama de blocos para o cálculo da média.

Exemplo

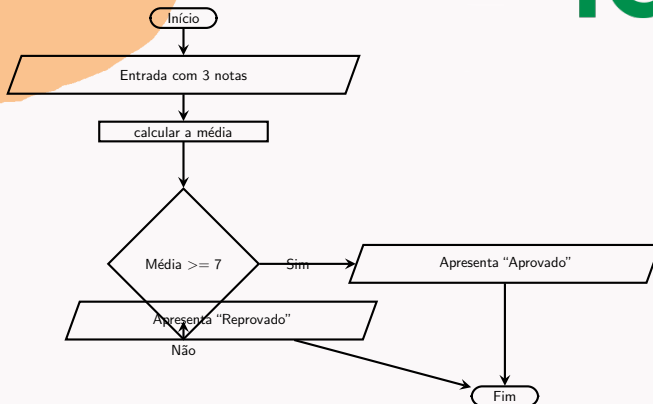


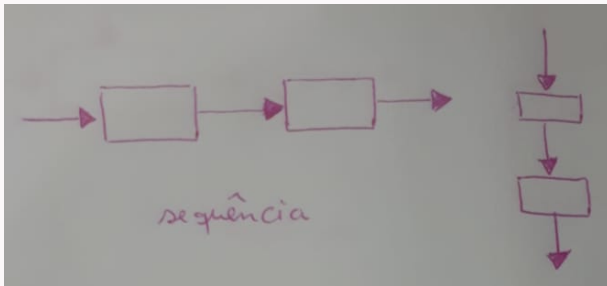
Figura: Diagrama de blocos para o cálculo da média.

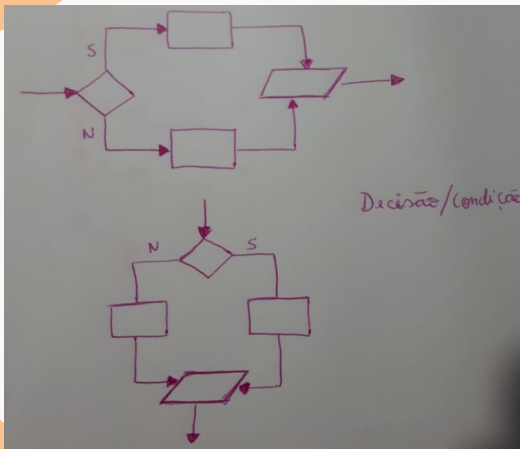
Particularidades entre lógicas

- **Linear:** a técnica lógica **linear** é conhecida como um modelo tradicional de desenvolvimento e resolução de um problema.
 - **Estruturada:** a técnica de lógica **estruturada** é mais usada pelo profissionais de desenvolvimento de sistemas. Possui características e padrões da técnica de lógica linear.
 - **Modular:** a técnica da lógica **modular** é usada decompor um diagrama em partes independentes.
 - **Linguagem C:** linguagem de programação compilada de propósito geral, estruturada, imperativa, procedural.
-

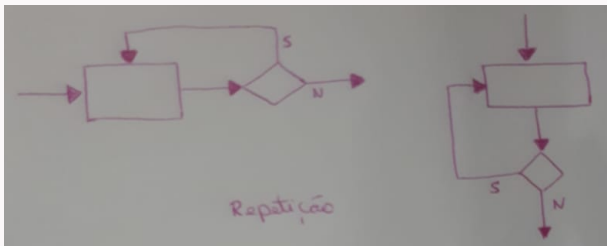
É uma forma de lógica com ênfase no uso de subrotinas, ciclo de repetição, condicionais e estruturas em bloco.

Sequência

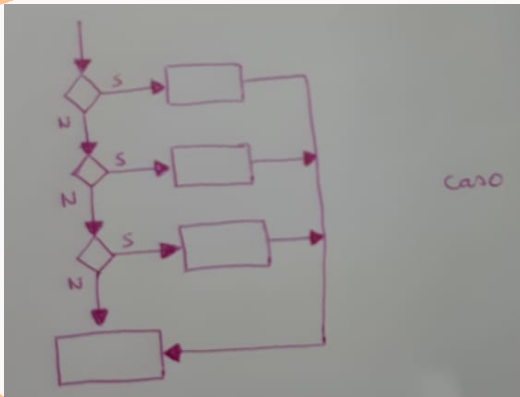




Repetição

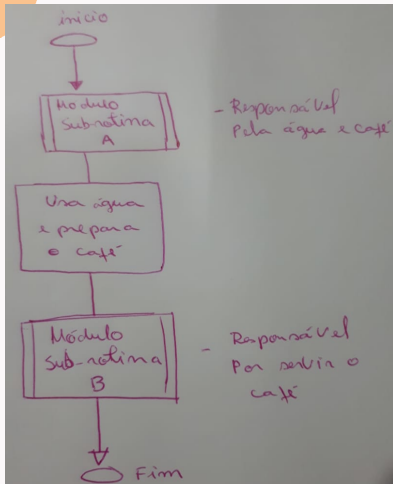


Caso



A técnica da lógica **modular** é elaborada como uma estrutura de partes independentes, denominadas de módulos, cujo procedimento segue regras, a saber:

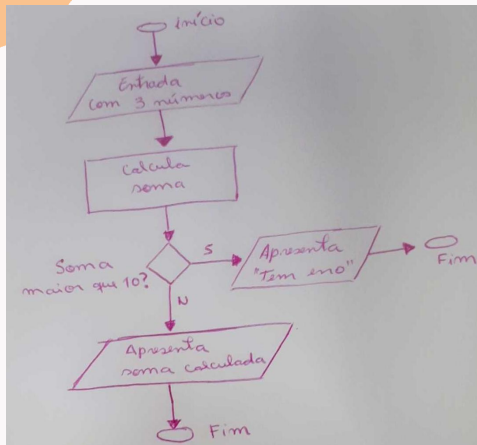
- decompor um diagrama em partes independentes; e
- dividir um problema em outros menores.



Exercício Lista 2 - Questão 5

I2-q5) Utilize diagrama de blocos para resolver o seguir problema: leia três números inteiros e calcule a soma. Considerar que a condição, se a soma for maior que 10, escreva “tem erro”, do contrário escreva o valor da soma.

Exercício Lista 2 - Questão 5

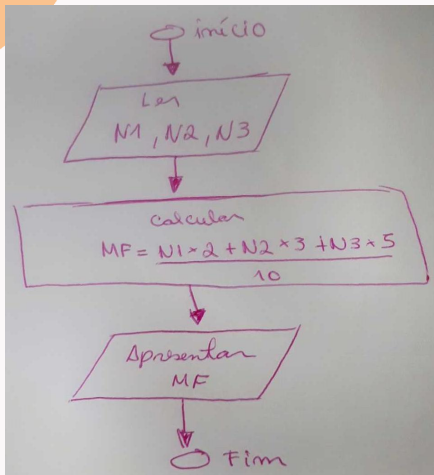


Exercício Lista 2 - Questão 6

I2-q6) Utilize diagrama de blocos para resolver o seguir problema: leia três notas de um aluno, calcule e escreva a média final deste aluno. Considerar que a média é ponderada e que o peso das notas é 2, 3 e 5.

$$\text{Média Final (MF)} = \frac{N1 \times 2 + N2 \times 3 + N3 \times 5}{10}$$

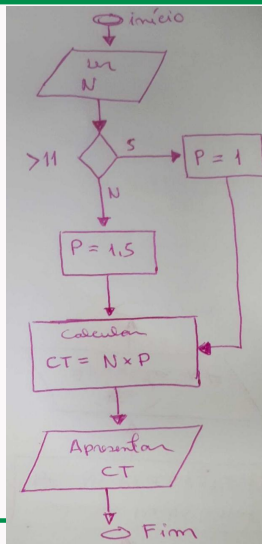
Exercício Lista 2 - Questão 6



Exercício Lista 2 - Questão 7

l2-q7) Utilize diagrama de blocos para resolver o seguir problema: as maçãs custam R\$ 1,50 cada se forem compradas menos de uma dúzia, e R\$ 1,00 se forem compradas pelo menos 12. leia o número de maçãs compradas, calcule e escreva o custo total da compra.

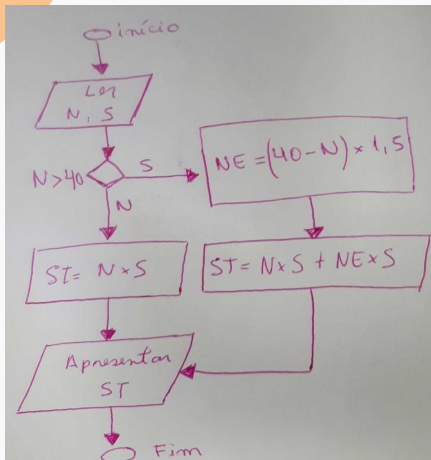
Exercício Lista 2 - Questão 7



Exercício Lista 2 - Questão 8

12-q8) Utilize diagrama de blocos para resolver o seguir problema: a jornada de trabalho semanal de um funcionário é de 40 horas. O funcionário que trabalhar mais de 40 horas receberá hora extra, cujo cálculo é o valor da hora regular com um acréscimo de 50%. leia o número de horas trabalhadas em um mês, o salário por hora e escreva o salário total do funcionário, que deverá ser acrescido das horas extras, caso tenham sido trabalhadas (considere que o mês possua 4 semanas exatas).

Exercício Lista 2 - Questão 8



Tipos de Dados e Instruções Primitivas

A partir deste ponto de estudo, teremos contato direto com a parte mais prática desta disciplina. Anteriormente, apresentou-se uma teoria básica de alguns pontos que, por vezes, levaram dúvidas até a alguns profissionais experientes da área de tecnologia da informação.

Daqui pra frente é necessário considerar que um computador é uma ferramenta para solucionar problemas que envolvam a manipulação de informação, as quais se classificam, grosso modo, em dois tipos básicos: dados e instruções.

Os dados são representados por elementos advindos do mundo externo, os quais representam as informações que os seres humanos manipulam. Eles devem ser abstraídos para serem processados em um computador. Os dados podem ser categorizados em três tipos: numéricos, caracteres e lógicos.

Tipos de dados

- **Numéricos** - representados por valores inteiros ou reais.
- **Caracteres** - representados por valores alfabéticos ou alfanuméricos os quais não são utilizados em operações de cálculo matemático.
- **Lógicos** - representados por valores dos tipos falsos ou verdadeiro.

*Os tipos de dados **numérico inteiro**, **numérico real**, **caracteres** e **lógico** são conhecidos como tipos de **dados primitivos** ou tipos de **dados básicos**.

Tipos de dados NUMÉRICOS

São tipos de dados numéricos possuem duas classes, números inteiros e reais.

- **INTEIRO** - são inteiros os dados numéricos positivos e negativos pertencentes ao **conjunto de números inteiros**, excluindo qualquer valor numérico fracionário, como por exemplo: R\$ 3,19 (não inteiro, fracionário).
 - **REAL** - são reais os dados numéricos positivos e negativos pertencentes ao **conjunto de números reais**, incluindo todos os valores fracionários e também os valores inteiros, como por exemplo: R\$ 3,19 (não inteiro, fracionário, real).
-

Os tipos de caracteres são sequência de valores delimitados por aspas formadas por letras de (A até Z), números (|de 0 até 9|) e símbolos - todos os símbolos do teclado. Ele também é conhecido como **alfanumérico**, **string** (em inglês - cordão, colar), **literal** ou **cadeia**.

Por exemplo, “Rua João e Maria”.

São lógicos os dados com valores que sugerem uma única opção entre duas possibilidades existentes, normalmente representados pelos valores **FALSO** ou **VERDADEIRO**, podendo também ser representados por **SIM** ou **NÃO**, **1 (um)** ou **0 (zero)**, entre outros, desde que mantida a relação de escolher apenas uma das duas opções de possibilidade existentes.

Ele também é conhecido por **booleano**, devido à contribuição de George Boole.

Tem-se como definição de variável tudo aquilo que é sujeito a variação, que é incerto, instável ou inconstante.

Para utilizar o conceito de variável, imagine que a memória de um computador é um grande arquivo com várias gavetas, e cada gaveta pode armazenar um único valor (seja ele numérico, caractere ou lógico).

O uso de variáveis



O uso de variáveis

Se a memória do computador for comparado a um grande arquivo com várias gavetas, e em cada gaveta é possível guardar um único valor por vez. Como em um arquivo, as gavetas devem estar identificadas com uma etiqueta contendo um nome.

Você há de concordar que é necessário identificar com um nome a gaveta que se pretende utilizar. Desta forma o valor armazenado pode ser utilizado a qualquer momento.

O uso de variáveis



O uso de variáveis

O nome de uma variável é utilizado para sua identificação e representação dentro de um programa de computador. É necessário estabelecer e seguir algumas regras de definição e uso de variáveis, a saber:

- Os nomes de identificação de uma variável podem utilizar um ou mais caracteres, limitando-se a restrição da própria linguagem formal de programação em uso. No caso do diagrama de blocos essa restrição não existe.

O uso de variáveis (continuação)

- O primeiro caractere de identificação do nome de uma variável não pode ser, em hipótese alguma, numérico. O primeiro caractere de identificação do nome de uma variável deve sempre ser alfabético, os demais podem ser alfanuméricos.
 - Na definição de um nome composto de uma variável não podem existir espaços em branco entre os nomes.
 - Jamais uma variável pode ser definida com o mesmo nome de uma palavra que representa os comandos de uma linguagem de programação de computador, ou seja, com as palavras reservadas de uma linguagem.
-

Por fim, do ponto de vista computacional pode-se definir uma variável de forma bem simplista como uma representação de uma região de memória utilizada para armazenar um determinado valor por um espaço de tempo. O tempo de armazenamento de um valor está relacionado ao tempo de duração da execução de um programa.

Constante é tudo que é fixo, estável, inalterável, imutável, contínuo, invariável, de um valor fixo e que é aplicado sob diversos pontos de vista. Assim sendo, do ponto de vista computacional, que é semelhante ao matemático ou científico, uma constante é uma grandeza numérica que mantém-se inalterado.

Os operadores aritméticos

Os operadores aritméticos são as ferramentas responsáveis pelo estabelecimento das operações matemáticas a serem realizadas em um computador. Tanto variáveis como constantes são utilizadas na elaboração dos cálculos matemáticos. Eles são classificados em:

- **unários** - quando atuam na inversão do estado de um valor numérico, que pode este ser passado de positivo para negativo ou de negativo para positivo; ou
 - **binários** - quando utilizados em operações matemáticas de exponenciação, divisão, multiplicação, adição e subtração.
-

Os operadores aritméticos

Operador	Operação	Categoria	Prioridade	Resultado
=	Atribuição	-o-	-o-	-o-
+	Manutenção de sinal	unário	1	-o-
-	Inversão de sinal	unário	1	-o-
^	Exponenciação	binário	2	inteiro ou real
/	Divisão	binário	3	real
*	Multiplicação	binário	3	inteiro ou real
+	Adição	binário	4	inteiro ou real
-	Subtração	binário	4	inteiro ou real

*o quadro apresenta os operadores aritméticos segundo a ordem de prioridade matemática em que as operações aritméticas são realizadas. Para alterar o nível de prioridade deve utilizar-se parênteses. Por exemplo, $X = 10 * 10 + 2$ e $Y = 10 * (10 + 2)$, X tem resultado 102 e Y tem resultado 120.

As instruções a serem implementadas para execução de um determinado programa são representadas por um conjunto de palavras-chave (vocabulário). Uma instrução pode se:

- **formal** - quando baseia-se em linguagem real de programação de computadores, tais como C, Java, PHP; ou
- **informal** - quando baseia-se em pseudocódigo, como é o caso da linguagem usada até, diagrama de blocos.

Para criar um programa que seja executável dentro de um computador, é preciso ter em mente três pontos de trabalho:

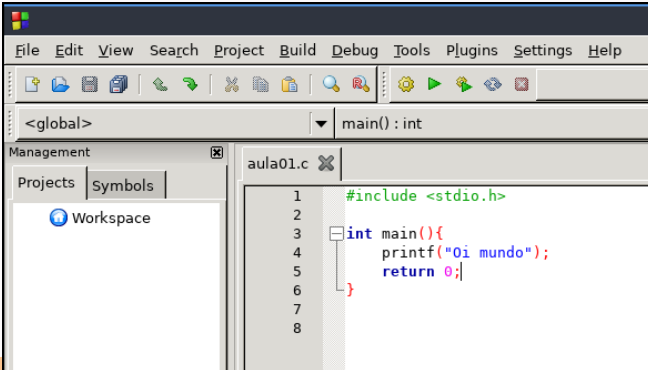
- entrada de dados - alimenta o programa;
- processamento - processa a(s) entrada(s); e
- saída de dados - apresenta o resultado.

Introdução à linguagem c

Meu primeiro programa

Estrutura básica de um programa em C.

Code::Blocks IDE www.codeblocks.org



The screenshot shows the Code::Blocks IDE interface. The menu bar includes File, Edit, View, Search, Project, Build, Debug, Tools, Plugins, Settings, and Help. The toolbar contains icons for file operations, execution, and settings. The 'Management' panel on the left shows 'Projects' and 'Symbols' tabs, with 'Workspace' selected. The main editor window, titled 'aula01.c', displays the following C code:

```

1  #include <stdio.h>
2
3  int main(){
4      printf("Oi mundo");
5      return 0;
6  }
7
8

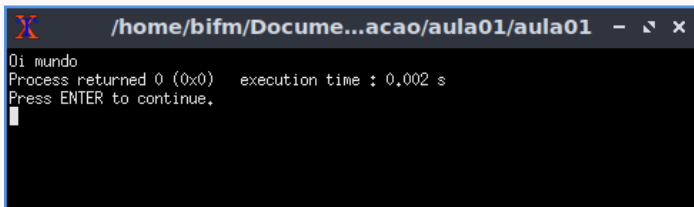
```

```
1 #include <stdio.h>
2 /*
3  * Programa 001 — Exemplo oi mundo
4  * imprime na tela a mensagem oi mundo
5  */
6 int main(){
7     printf("Oi mundo.");
8     return 0;
9 }
```

Listing 1: Exemplo de Código

Meu primeiro programa

Estrutura básica de um programa em C.



```
X /home/bifm/Docume...acao/aula01/aula01 - [icon] x
Oi mundo
Process returned 0 (0x0)   execution time : 0.002 s
Press ENTER to continue.
█
```

São blocos de códigos ignorados pelo compilador.

// comentário de linha

/*

comentário de bloco

*/

Identar um código nada mais é que separar os códigos em blocos através de tabulação.

Não indentado:

```
#include <stdio.h>int main(){printf("Oi mundo");return 0;}
```

*O código acima não está indentado. Note como está complicado de ler, apesar de ser um código extremamente simples.

Tipos de Dados

A linguagem C possui 5 (cinco) tipos básicos de dados: **char**, **int**, **float**, **void** e **double**.

O **void** é um tipo de dado vazio (sem valor).

Para cada tipo de dado existem modificadores de tipo, estes são 4 (quatro): **signed**, **unsigned**, **long** e **short**.

Para o **float** nenhum modificador pode ser aplicado; assim como para o **double** podemos aplicar APENAS o **long**.

Nome	Característica	Intervalo
char	carácter/literal	-128 a 127
int	número inteiro	-32.768 a 32.767
float	ponto flutuante em simples precisão	3.4 E-38 a 3.4E+38
double	ponto flutuante em dupla precisão	1.7 E-308 a 1.7E+308
void	vazio	(sem valor)

Qualificadores de tipo **unsigned** e **signed** podem ser aplicados a tipos **char** e **int**:

unsigned - considera apenas valores positivos.

signed - consideram valores positivos e negativos. por padrão todos são signed.

Qualificadores de tipo **long** e **short** podem ser aplicados a tipos **int** e **double**. Entretanto, para **double**, apenas o **long** é aplicado.

short - aplicável apenas para **int**. Reduz pela metade o tamanho.

long - aplicável para **int** e **double**.

Tipos de dados

Tipo	Número de bytes	Formato Sintaxe printf	Intervalo	
			Início	Fim
char	1	%c	-128	127
unsigned char	1	%c	0	255
signed char	1	%c	-128	127
int	4	%d	-32.768	32.767
unsigned int	4	%u	0	65.535
signed int	2	%i	-32.768	32.767
short int	2	%hi	-32.768	32.767
unsigned short int	2	%hu	0	65.535
signed short int	2	%hi	-32.768	32.767
long int	8	%li	-2.147.483.648	2.147.483.647
signed long int	8	%li	-2.147.483.648	2.147.483.647
unsigned long int	8	%lu	0	4.294.967.295
double	8	%d	-	-
long double	16	%d	-	-
float	4	%f	-	-
long (inteiro)	4	%f	-	-
short (inteiro)	2	%f	-	-

Declaração de variável:

tipo_da_variavel nome_da_variavel = valor_inicial_da_variavel;

Declaração de variáveis de um mesmo tipo:

tipo_da_variavel nome_da_variavel1 = valor1, nome_var2 = valor2;

Ao nomear uma variável seja objetivo, use nomes fáceis de entender e se necessário faça um comentário acima da variável explicando sua utilidade. Em nomes compostos separe-os utilizando underline.

Constantes de barra invertida

Constante	Representação
<code>\n</code>	Nova Linha
<code>\t</code>	Tab Horizontal
<code>\v</code>	Tab Vertical
<code>\"</code>	Aspas Duplas
<code>\'</code>	Aspas Simples
<code>\\</code>	Barra Invertida
<code>\a</code>	Sinal sonoro/Beep

Operadores aritméticos e de atribuição

Operador	Ação
+	Soma
-	Subtração
*	Multiplicação
/	Divisão
%	Resto da divisão
++	Incremento
--	Decremento

Operadores de Racionais/lógicos

Operador	Ação
>	Maior do que
>=	Maior ou igual a
<	Menor do que
<=	Menor ou igual a
==	Igual a
!=	Diferente de
&&	AND (E)
	OR (OU)
!	NOT (NÃO)

Tabela verdade

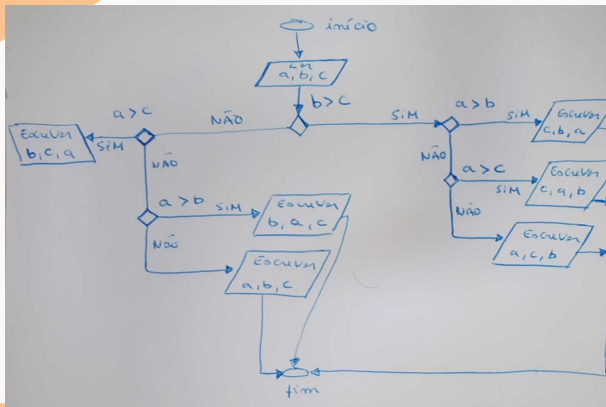
p	q	$p \ \&\& \ q$	$p \ \ q$
verdadeiro	verdadeiro	verdadeiro	verdadeiro
verdadeiro	falso	falso	verdadeiro
falso	verdadeiro	falso	verdadeiro
falso	falso	falso	falso

Tabela de Precedência

Maior precedência
() [] ->
! ~ ++ -- . -(unário)
(cast) *(unário)
&(unário) sizeof
* / %
+ -
<< >>
< <= > >=
== !=
&
^
&&
?
= += -= *= /=
,
Menor precedência

Resolução L3Q7

Resolução da questão 7 (Q7), lista 3 (L3).



scanf(), **gets()**, **fgets()** - utilizados para entrada de dados.

printf() - utilizado para saída de dados.

Código	Formado
%c	Receber um caractere
%d	Receber número inteiro
%f	Receber número de ponto flutuante
%s	Receber uma cadeia de caracteres

Entrada e saída de dados

scanf() - a entrada é finalizada assim que um espaço em branco (whitespace) é encontrado. É possível burlar usando `scanf("%[^\n]s", x)`

gets() - considera um espaço em branco como parte da cadeia de entrada e termina a entrada ao encontrar newline ou EOF.

Exemplo 002

```
10 #include <stdio.h>
11 /*
12  * Programa 002 — Exemplo de entrada e saída de dados
13  * para receber dois valores, nome e sobrenome
14  * após leitura os valores são impressos na tela do usuário
15  */
16 int main(){
17     char nome[10]; // nome da pessoa
18     char sobrenome[10]; // sobrenome da pessoa
19     printf("saída de dados, digite seu nome: ");
20     scanf("%s", nome); // entrada de dados
21     scanf("%*[^\\n]%*c"); // para limpar o buffer de entrada
22     printf("saída de dados, digite seu sobrenome: ");
23     scanf("%s", sobrenome); // entrada de dados
24     printf("saída de dados na tela: bem vindo, %s %s\\n", nome, sobrenome);
25     return 0;
26 }
```

Listing 2: Exemplo de Código

Exemplo 003

```
27 #include <stdio.h>
28 /*
29  * Programa 003 — Exemplo de entrada e saída de dados
30  * para receber dois valores, nome e sobrenome
31  * após leitura os valores são impressos na tela do usuário
32  */
33 int main(){
34     char nome[10]; // nome da pessoa
35     char sobrenome[10]; // sobrenome da pessoa
36     printf("saída de dados, digite seu nome: ");
37     gets(&nome); // entrada de dados
38     printf("saída de dados, digite seu sobrenome: ");
39     gets(&sobrenome); // entrada de dados
40     printf("saída de dados na tela: bem vindo, %s %s\n", nome, sobrenome);
41     return 0;
42 }
```

Listing 3: Exemplo de Código

- São responsáveis por controlar o fluxo do programa
- Testam condições.
- Algumas são conhecidas como “loops”.

A estrutura **if-else** é utilizada para tomada de decisões, quando uma condição é válida ou não.

```
if ( condicao ) {  
    bloco_de_comando  
} else {  
    bloco_de_comando  
}
```

Exemplo 004

```
43 #include <stdio.h>
44 /*
45  * Programa 004 — Exemplo if-else
46  */
47 int main(){
48     int idade=20;
49     if(idade > 17){
50         printf("Maior de idade.");
51     }else{
52         printf("Menor de idade");
53     }
54     return 0;
55 }
```

Listing 4: Exemplo de Código

switch

O **switch** também é utilizado para tomada de decisões, porém cria um código mais limpo. Com ele você pode testar uma variável em relação a diversos valores pré- estabelecidos.

```
switch ( variável ) {  
    case 1:  
        bloco_de_comando  
        break;  
    default:  
        bloco_de_comando  
}
```

Exemplo 005

```
56 #include <stdio.h>
57 /*
58  * Programa 005 — Exemplo switch
59  */
60 int main(){
61     int idade=18;
62     switch(idade){
63         case 18:
64             printf("idade 18");
65             break;
66         default:
67             printf("outra idade");
68     }
69     return 0;
70 }
```

Listing 5: Exemplo de Código



switch

*obs: trabalha com números inteiros.

while

O **while** é uma estrutura de repetição, utilizada para criar os chamados “loops” de um programa. O código dentro do bloco repetirá enquanto a condição não for verdadeira.

```
while ( condicao ) {  
    bloco_de_comando  
}
```

*É utilizando quando não sabe-se a quantidade de repetições.

Exemplo 006

```
71 #include <stdio.h>
72 /*
73  * Programa 006 — Exemplo while
74  */
75 int main(){
76     int idade=17;
77     while(idade % 2 != 0){
78         printf("idade %i\n", idade);
79         idade = idade + 1;
80     }
81     return 0;
82 }
```

Listing 6: Exemplo de Código

for

Assim como o **while** o **for** é utilizado para criar estruturas de repetição.

```
for ( inicializacao; condicao; incremento ) {  
    bloco_de_comando  
}
```

Exemplo 007

```
83 #include <stdio.h>
84 /*
85  * Programa 006 — Exemplo while
86  */
87 int main(){
88     int idade=17;
89     while(idade % 2 != 0){
90         printf("idade %i\n", idade);
91         idade = idade + 1;
92     }
93     return 0;
94 }
```

Listing 7: Exemplo de Código

Ambiente de desenvolvimento

- Funcionamento; e
- configuração do ambiente para execução;

288S - Linguagem de Técnicas de Programação

Prof. MSc. Bruno Iran Ferreira Maciel

Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas

IESO
INSTITUTO DE ENSINO
SUPERIOR DE OLINDA

faccor
Olinda

**Olinda
facir**



29 de agosto de 2019



J. A. N. G. Manzano and J. F. de Oliveira.

Algoritmos: Lógica para desenvolvimento de programação de computadores.

Érica, 21th edition, 2008.

Olinda
facir