

ALGORITMO E LÓGICA DE PROGRAMAÇÃO

Carlos Veríssimo

```
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
```

Foto: Pixabay

E-book 2

FAM
ONLINE

Neste E-Book:

Introdução	3
A escolha de linguagem de programação	4
As Principais Linguagens de desenvolvimento	4
Transformado o Algoritmo em Código	7
Uma Visão Geral da Linguagem C	7
O C é “Case Sensitive”	8
Estrutura da linguagem C	9
Primeiros passos	11
Expressões	16
Manipulação de Dados em Memória.....	16
Nomes de identificadores de Dados	17
Variáveis	19
Localização da declaração das variáveis.....	20
Variáveis Locais	20
Parâmetros Formais	22
Variáveis Globais.....	23
Constantes	24
Trabalhando Operadores	25
Operadores de Atribuição.....	26
Operadores Aritméticos.....	27
Operadores Relacionais e Lógicos	29

O Operador Ternário.....	31
Resolvendo juntos um problema prático	33
Considerações Finais.....	41
Síntese	43

INTRODUÇÃO

Nesta unidade, você terá um contato mais próximo da arte de programar computadores. Estudaremos dos conceitos e paradigmas da lógica de programação para a implementação de seus algoritmos em uma das linguagens de programação mais praticadas na atualidade: A linguagem de programação **C**. Começamos nossos estudos com as principais linguagens de programação da atualidade, para você entender a escolha da linguagem adotada neste curso. Daremos seguimento ao explorarmos a estrutura da linguagem de programação **C**, bem como exploraremos os **elementos básicos da lógica de programação**, implementados na referida linguagem. Nesse sentido, você aprenderá a manipular variáveis, manipular os operadores aritméticos, lógicos e relacionais. Você aprenderá a declarar e a utilizar função. Daremos também início à prática de programar, utilizando um ambiente on-line (acesso à internet), que proporcionará a você a liberdade de produzir seus programas, sem a necessidade de instalação de um ambiente de desenvolvimento em seu computador, e, ao final, praticaremos juntos um programa em linguagem **C** para a solução de um problema.

Vamos em frente, espero que você tenha um excelente aproveitamento deste maravilhoso mundo da programação de computadores.

Bons Estudos!

A ESCOLHA DE LINGUAGEM DE PROGRAMAÇÃO

As Principais Linguagens de desenvolvimento

Verificamos, anteriormente, os passos necessários para solução de problemas com a utilização do computador. Estudamos que devemos partir do entendimento geral, passando pela diagramação, até chegar ao algoritmo. Para que o algoritmo seja executado por um computador, faz-se necessário a codificação deste em uma linguagem que o computador possa interpretar e executar as instruções.

Com o desenvolvimento da engenharia da computação e seu virtuoso crescimento do poder de processamento, surgiram muitas linguagens de programação. A Figura 1 mostra uma lista da infinidade de linguagens que existem hoje.

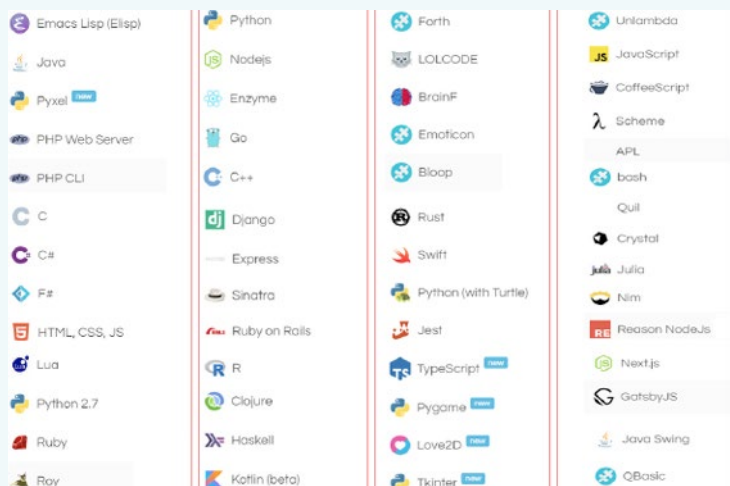


Figura 1: Relação de linguagens de Programação utilizadas na atualidade. **Fonte:** Elaboração Própria

A Figura 2 mostra o ranking das 10 linguagens de programação mais consultadas no ano de 2019. Podemos inferir que esta tabela nos mostra as linguagens de programação com maior índice de procura, como solução de desenvolvimento. Ou seja, entendemos que se programam mais nestas linguagens nos dias de hoje. Note o posicionamento da linguagem de programação **C**: é a **segunda linguagem mais procurada**.

Jun 2019	Jun 2018	Change	Programming Language	Ratings	Change
1	1		Java	15.004%	-0.36%
2	2		C	13.300%	-1.64%
3	4	▲	Python	8.530%	+2.77%
4	3	▼	C++	7.384%	-0.95%
5	6	▲	Visual Basic .NET	4.624%	+0.86%
6	5	▼	C#	4.483%	+0.17%
7	8	▲	JavaScript	2.716%	+0.22%
8	7	▼	PHP	2.567%	-0.31%
9	9		SQL	2.224%	-0.12%
10	16	▲	Assembly language	1.479%	+0.56%

Figura 2: Ranking das linguagens mais procuradas. **Fonte:** [tio.be](#).

SAIBA MAIS

Índice TIOBE (do inglês, TIOBE Programming Community Index) é uma lista ordenada de linguagens de programação, classificada pela frequência de pesquisa na web usando o nome da linguagem como a palavra-chave. O índice cobre buscas no Google, Google Blogs, MSN, Yahoo!, Wikipédia e no Youtube.

Fonte: [Wikipedia](#)

TRANSFORMADO O ALGORITMO EM CÓDIGO

Neste curso, adotaremos a linguagem **C**, por sua ampla procura pelos programadores atualmente (observe o ranking das 10 maiores linguagens) para a implementação de nossos algoritmos. Utilizaremos também um ambiente de desenvolvimento on-line (explicado na seção “Primeiros Passos”), pela facilidade de utilização deste. Como dito anteriormente, a utilização do referido ambiente dispensará você de instalar e configurar um ambiente de desenvolvimento em seu computador. Você poderá codificar e testar o seu programa de qualquer computador com acesso à internet.

Uma Visão Geral da Linguagem C

O **C** é uma linguagem de programação de uso amplo que é utilizada para a criação de programas com variados objetivos, como processadores de texto, planilhas eletrônicas, sistemas operacionais, programas de comunicação, programas para a automação (microprocessadores), programas para a solução de problemas nas diversas engenharias.

A linguagem **C** exige uma familiaridade com o compilador e experiência em solucionar problemas de

lógica nos programas. É importante, então, que você o compile e execute os exemplos apresentados. Para isso, sugiro que você utilize o ambiente on-line para executar todos os exemplos que constam nesta e em outras unidades.

O C é “Case Sensitive”

Devemos ressaltar, de início, que o **C** é “**Case Sensitive**”, isto é, maiúsculas e minúsculas fazem diferença. Ao declarar uma variável com o nome **soma** ela não equivale à **Soma**, **SOMA**, **SoMa** ou **sOmA**. Da mesma maneira, os comandos do **C** **if** e **for**, por exemplo, só podem ser escritos em minúsculas, pois, caso contrário, o compilador não irá interpretá-los como sendo comandos, mas sim como variáveis.

SAIBA MAIS

O C nasceu na década de 1970. Seu inventor, Dennis Ritchie, implementou-o, pela primeira vez, usando um DEC PDP-11 rodando o sistema operacional UNIX.

O C é derivado de uma outra linguagem: o B, criado por Ken Thompson. O B, por sua vez, veio da linguagem BCPL, inventada por Martin Richards.

Estrutura da linguagem C

- Um código em **C** pode ser estruturado com várias funções, mas a **principal**, a que obrigatoriamente deve aparecer é a **main()**
- A partir da função **main()** é possível chamar as demais funções que possam fazer parte do programa
- Não é obrigatório que a primeira função no código seja a **main()**
- Todas as funções e estruturas devem começar com { (Símbolo “abre chave”) e devem ser finalizadas por } (Símbolo “fecha chave”). O conteúdo que está entre estes dois símbolos é chamado de **bloco de comandos**.
- As linhas que contêm comandos devem ser finalizadas com símbolo ; (ponto e vírgula)
- Se o código for utilizar funções definidas em outros arquivos é necessário incluir o arquivo de definição
- O arquivo de definição possui a extensão **.h**
- O comando **#include** é uma indicação para o compilador adicionar o arquivo associado ao arquivo executável que será gerado ao final do processo.
Exemplo: **#include <stdio.h>**

• Utilizando Comentários

Não são considerados na compilação, apenas facilitam o entendimento do código

- Formatos do comentário:
 - `//` (Símbolo de duas barras) - Comentários simples - Tudo que aparecer após `//` será ignorado pelo compilador, até o final da linha)
 - `/* comentários */` - Comentário múltiplas linhas - Tudo que estiver entre `/*` e `*/` será ignorado pelo compilador, podendo conter **várias linhas**)
- **Boas Práticas:** Essencial colocar um comentário no início do código contendo as seguintes informações: nome do programa, propósito, nome do programador e da empresa, data, versão.

No código abaixo vemos um exemplo completo de uma estrutura de programa na linguagem **C**, em que podemos observar que, ao topo, a área de comentário é muito rica, e podemos entender o objetivo do programa e a autoria com muita clareza. Note também que, ao final de cada comando (Identificado pelo caractere `“;”` – ponto e vírgula), temos um comentário com a descrição do respectivo comando, para observar na prática a utilização destes.

SAIBA MAIS

Todo programa **C** inicia pela função principal ***main()***, e esta desenvolve a chamada das outras funções.

O sistema operacional espera que o programa retorne o resultado da execução, e é a função

main(), que se encarrega de fazê-lo. Se tudo de certo (normal), deve ser codificado o comando **return 0**.

```
/*  
quadrado.cpp : calcula o quadrado de um número inteiro  
Escrito por nome do autor - FAM Online  
Data: 12/junho/20xx - Versão 1.0  
*/  
#include <studio.h> //Definição de diretivas de biblioteca para printf e scanf  
#include <conio.h> //Definição de biblioteca para getch  
main() //Função principal  
{  
    int num, quad; //Definição de variáveis  
    printf("Digite um valor inteiro"); //Mensagem que aparece em tela  
    scanf("%d, &num); //Guarda valor digitado  
    quad = num * num; //Calcula o quadrado do número  
    printf("O quadrado de %d = %d\n", num, quad); //Apresenta o resultado em tela  
    printf("Pressione qualquer tecla para finalizar"); //Mensagem em tela  
    getch(); //Aguarda pressionar uma tecla  
}
```

Primeiros passos

Existe uma quantidade considerável de ambientes integrados de desenvolvimento que proporcionam a codificação e teste de um programa em linguagem **C**, por exemplo: *Eclipse*, *Code Books* ou *Visual Studio*. Para que possamos focar somente em seu aprendizado da linguagem, utilizamos um ambiente de desenvolvimento on-line, de acesso gratuito, pois este ambiente dispensa a necessidade de aprendizado de instalação dos ambientes de desenvolvi-

tos citados acima. Adicionalmente, o nosso foco é desenvolver algoritmos que sejam compilados e executados em alguma linguagem de programação. Isso facilita o nosso processo de atuação focado na prática de exercícios.

SAIBA MAIS

Você pode utilizar os seguintes Ambientes Integrados de Desenvolvimento (*IDE-Integrated Development Environment*) para desenvolvimento de programas na linguagem **C**

- **Eclipse**
- **Code Books** - <http://www.codeblocks.org/downloads>
- **Visual Studio** - <https://code.visualstudio.com>

Para fazer somente codificação, sem utilizar ambientes de desenvolvimento você pode utilizar os seguintes editores de texto:

- **Atom** - <https://atom.io>
- **Sublime Text** - <https://www.sublimetext.com>
- **Notepad++** - <https://notepad-plus-plus.org>

Caro estudante, vamos juntos desenvolver o nosso primeiro programa na linguagem **C**. Para isso, vamos seguir alguns passos e, ao final, você estará apto a resolver, com autonomia, os exemplos, no decorrer de nosso curso.

Passo 1

Acesse o ambiente de desenvolvimento on-line através da URL <https://repl.it/>. Você terá uma tela conforme demonstrado na Figura 3. Neste ponto, insira seu Usuário e Senhas cadastrados anteriormente.

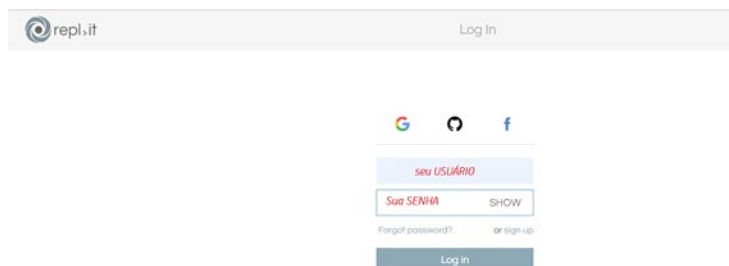


Figura 3: Autenticação do usuário no ambiente on-line de desenvolvimento. **Fonte:** Elaboração Própria.

Passo 2

Vamos criar um novo programa clicando no botão **"+ new repl"** (Passo 2.1), e, em seguida, selecionaremos a linguagem a ser codificada (Passo 2.2) e, finalmente, nomeamos o programa de **"MeuPrimeiroPrograma"** (Passo 2.3), informando também uma breve descrição, conforme demonstrado na Figura 4.

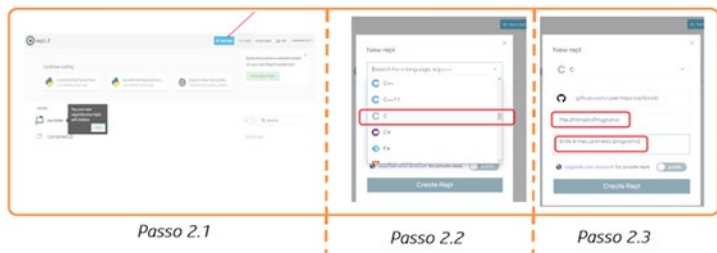


Figura 4: Passo para criar um novo provo programa no ambiente on-line. **Fonte:** Elaboração Própria.

Passo 3

Aqui você codifica o seu algoritmo. Cabe ressaltar que foi gerado automaticamente o código em destaque. Faça as alterações necessárias. Importante destacar que, do lado direito da tela, temos uma console para receber o resultado do programa, conforme demonstrado na Figura 5.

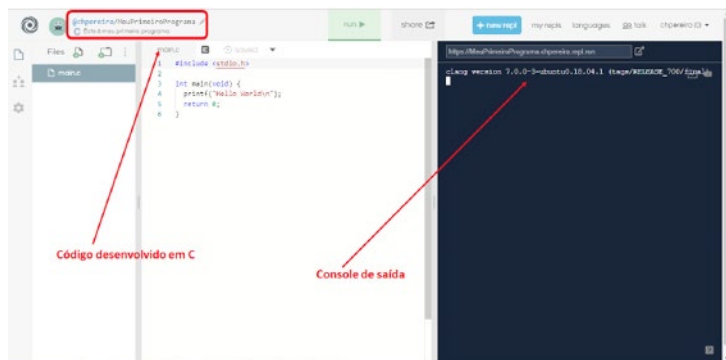


Figura 5: Tela de codificação do algoritmo no ambiente on-line. **Fonte:** Elaboração Própria.

Passo 4

Agora vamos testar o programa. Para isso, acione o botão **“run”**. Note que o resultado produzido pelo seu programa – que, neste caso, é imprimir a mensagem **“Hello World”** – aparece na console, conforme demonstrado na Figura 6.

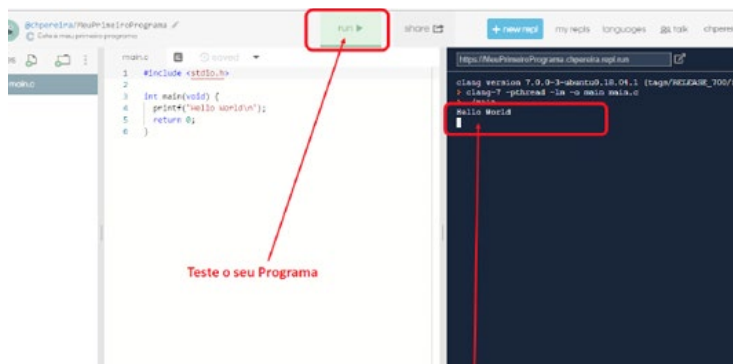


Figura 6: Teste do programa no ambiente on-line. **Fonte:** Elaboração Própria.

Pronto, você acabou de produzir um programa em linguagem **C**. De agora em diante, você está preparado para praticar os exemplos propostos neste curso, seguindo os **4 passos** que executamos. Independentemente da complexidade do problema, esses serão sempre os passos necessários para a construção de algoritmos em linguagem de programação.

Expressões

Nesta seção, abordamos o elemento mais fundamental da linguagem **C**: a expressão. As expressões são formadas pelos elementos mais básicos: **Dados** e **Operadores**. Os dados podem ser representados por variáveis ou constantes. Analisaremos que **C** manipula tipos diferentes de dados, bem como provê uma ampla variedade de operadores.

Manipulação de Dados em Memória

O **C** opera cinco tipos básicos de dados: caractere, inteiro, ponto flutuante, ponto flutuante de precisão dupla e sem valor, representados respectivamente por ***char***, ***int***, ***float***, ***double*** e ***void***. Importante destacar que o tamanho e a faixa desses tipos de dados variam de acordo com o tipo de processador e a implementação do compilador. Por exemplo, um caractere ocupa geralmente 1 byte e um inteiro ocupa 2 bytes. O padrão **ANSI** estipula apenas a faixa mínima de cada tipo de dados. Observe a Tabela 1, que mostra as faixas estipulas pelo padrão **ANSI**.

Tipo de dado	Significado	Tamanho (em bytes)	Intervalo de valores aceitos
char	Caractere	1	- de 128 a 127
unsigned char	Caractere não assinado	1	0 à 255
short int	inteiro curto	2	- de 32 768 a 32 767
unsigned short int	Inteiro curto não assinado	2	de 0 a 65 535
int	Inteiro	2 (no processador de 16 bits) 4 (no processador de 32 bits)	- de 32 768 a 32 767 - de 2 147 483 648 a 2 147 483 647
unsigned int	Inteiro não assinado	2 (no processador de 16 bits) 4 (no processador de 32 bits)	de 0 a 65 535 de 0 a 4 294 967 295
long int	Inteiro longo	4	- de 2 147 483 648 a 2 147 483 647
unsigned long int	Inteiro longo não assinado	4	de 0 a 4 294 967 295
float	Flutuante (real)	4	$3.4 \cdot 10^{-38}$ à $3.4 \cdot 10^{38}$
duplo	Flutuante duplo	8	de $1.7 \cdot 10^{-308}$ a $1.7 \cdot 10^{308}$
longo duplo	Flutuante duplo longo	10	$3.4 \cdot 10^{-4932}$ à $3.4 \cdot 10^{4932}$

Tabela 1: Tipos de Dados definidos pelo padrão ANSI. **Fonte:** Elaboração Própria.

Nomes de identificadores de Dados

Identificadores são os nomes dados às variáveis, funções, rótulos e vários outros tipos de objetos definidos pelo programador. Esses identificadores podem variar de um a diversos caracteres. O primeiro

caractere deve ser uma letra ou um sublinhado e os caracteres subsequentes devem ser letras, números ou sublinhados. A Tabela 2 mostra exemplos de identificadores corretos e incorretos:

Correto	Incorreto	Obs
count	1count	É incorreto começar com número
teste123	teste!123	É incorreto uso do caractere "!" (Exclamação), por ser um caractere especial
high_income	high...income	É incorreto uso do caractere "." (ponto), por ser um caractere especial

Tabela 2: Exemplos de identificadores corretos e incorretos. **Fonte:** Elaboração Própria

Importante aspecto a ser considerado, ao dar nome a um identificador: O **C** é uma linguagem **“Case Sensitive”**, conforme abordamos anteriormente, portanto, count, Count e COUNT são três identificadores distintos.

Um identificador não pode ser igual a uma palavra-chave (Palavra reservada) e não é permitido também que tenha o mesmo nome que as funções que você escreveu no programa, ou as funções que constam na biblioteca C, por exemplo, uma variável *printf* não é nome válido, porém *Printf* é válido para o compilador. Observe a lista de palavra-chave C à Tabela 3.

auto	int
break	long
case	register
char	return

const	short
continue	signed
default	sizeof
do	static
double	struct
else	switch
enum	typedef
extern	union
float	unsigned
for	void
goto	volatile
if	while

Tabela 3: Lista de Palavras-Chave (Palavras Reservadas) da linguagem C. **Fonte:** Elaboração Própria.

Variáveis

Conforme já abordamos, uma variável é uma posição de memória, utilizada para guardar um valor que pode ser acessado e modificado pelo programa. Toda variável em **C** deve ser declarada **antes** de usada. A sintaxe de uma declaração é:

tipo lista_de_variaveis;

Em que o **tipo** deve ser um tipo de dado válido em **C**, e **lista_de_variavies**, pode consistir em um ou mais nomes de identificadores separados por vírgula, conforme podemos observar nos exemplos abaixo:

- `int nota1, nota2, nota3;`
- `short int numero;`
- `int nota;`
- `double salario;`

Localização da declaração das variáveis

Uma variável em **C** deve ser declarada em três locais possíveis: Dentro de funções (Variáveis Locais), na definição dos parâmetros das funções (Parâmetros formais), e fora de todas as funções (Variáveis globais).

Variáveis Locais

São as variáveis declaradas dentro de uma função. Variáveis locais somente podem ser referenciadas por comandos que estão dentro do **bloco** (Designado por “{” e “}”) no qual as variáveis foram declaradas. Reveja o conceito de bloco de comandos na seção “*Estrutura da Linguagem C*”.

Variáveis locais existem apenas enquanto o bloco de comandos em que foram declaradas está sendo executado. Considere o seguinte trecho de código:

```
void funcao1(void)
{
    int valor;
    valor1 = 10;
}
void funcao2(void)
{
    int valor;
    valor = 321;
}
```

Figura 7: Variáveis Locais. **Fonte:** Elaboração Própria.

A variável inteira **valor** é declarada duas vezes, uma vez dentro da função **funcao1()** e outra dentro da função **funcao2()**. Neste exemplo, cada variável **valor** é reconhecida apenas pelo código do respectivo bloco de comandos da declaração da variável. Ou seja, qualquer manipulação da variável **valor**, dentro da **função1()**, está a tratar com o conteúdo 10 e a manipulação da variável **valor**, dentro da **funcao2()**, está a tratar o conteúdo 321.

Pelo fato de toda variável local ser criada e destruída a cada entrada e saída do bloco em que elas são declaradas, seu conteúdo é perdido quando o bloco deixa de ser executado. Quando uma função é chamada, suas variáveis locais são criadas e, ao retornar, elas são destruídas.

Parâmetros Formais

Parâmetros formais são as variáveis que receberão os valores dos argumentos tratados pela função. Essas variáveis se comportam como qualquer outra variável local dentro da função. Conforme demonstrado no código abaixo, podemos observar que as variáveis **operação**, **valor1** e **valor2** foram declaradas na criação da função **operaValores()**. Essas variáveis são utilizadas dentro da função e têm o mesmo comportamento das variáveis locais.

```
#include <stdio.h>
int operaValores(char operacao, int valor1, int valor2)
{
    int resultado = 0;

    if (operacao == 'A')
    {
        resultado = valor1 + valor2;
    }
    else
    {
        resultado = valor1 * valor2;
    }
    return resultado;
}

int main(void) {
    int operacao = operaValores('M', 2, 3);
    printf("O valor da Operação é:%d", operacao);
    return 0;
}
```

Figura 8: Parâmetros Formais. **Fonte:** Elaboração Própria.

Variáveis Globais

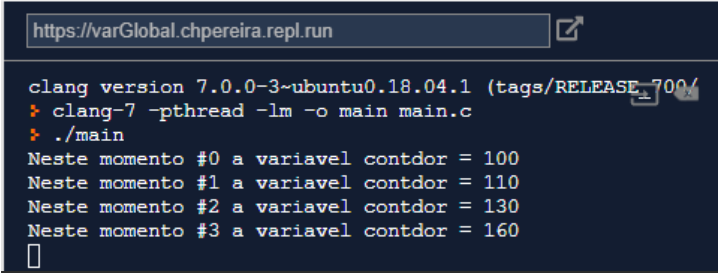
As variáveis globais são reconhecidas e podem ser manipuladas em toda a extensão do programa, ou seja, em todo o código. Adicionalmente, essas variáveis preservam o valor de seus conteúdos durante toda a execução do programa. Essas variáveis devem ser declaradas fora de qualquer função do programa.

No código abaixo podemos perceber que a variável **contador** foi declarada fora de qualquer função, no entanto, as funções **main()**, **funcao1AlteraContador()** e **funcao2AlteraContador()** manipularam e alteraram o valor da variável contador.

```
#include <stdio.h>
int contador = 100; //Declaração da variável Global "contador"
//
void funcao1AlteraContador()
{
    contador = contador + 10;
    printf("Neste momento #1 a variavel contdor = %d\n",contador);
    //
}
void funcao2AlteraContador()
{
    contador = contador + 20;
    printf("Neste momento #2 a variavel contdor = %d\n",contador);
    //
}
int main(void) {
    //
    printf("Neste momento #0 a variavel contdor = %d\n",contador);
    //
    funcao1AlteraContador();
    //
    funcao2AlteraContador();
    //
    contador = contador + 30;
    printf("Neste momento #3 a variavel contdor = %d\n",contador);
    //
    return 0;
}
```

Figura 9: Variáveis Globais. **Fonte:** Elaboração Própria.

Conforme podemos observar o resultado da execução na Figura 10, a variável foi alterada dentro de cada função na sequência de execução. A variável é declarada e inicializada com valor 100, e mostrada (*printf*). Em seguida, a função **funcao1AlteraContador()** é chamada e esta altera o valor da variável, somando +10. Logo depois, a função **funcao2AlteraContador()** é chamada e esta altera o valor da variável, somando +20 e, finalmente, a função **Main()** altera a variável, somando +30.



```
https://varGlobal.chpereira.repl.run
clang version 7.0.0-3~ubuntu0.18.04.1 (tags/RELEASE_700/
> clang-7 -pthread -lm -o main main.c
> ./main
Neste momento #0 a variavel contdor = 100
Neste momento #1 a variavel contdor = 110
Neste momento #2 a variavel contdor = 130
Neste momento #3 a variavel contdor = 160
□
```

Figura 10: Resultado do código que manipula variável global **Fonte:** Elaboração Própria.

Constantes

Na linguagem **C**, as constantes são valores fixos que o programa não pode alterar. Essas constantes podem ser de qualquer tipo de dado básico. As constantes do tipo caractere são envolvidas por aspas simples ('). Constantes inteiras são especificadas como números, sem componentes fracionários. Constantes em ponto flutuante requerem o ponto decimal seguida pela parte fracionária do número.

Na Tabela 4, podemos observar os vários exemplos de constantes com seus respectivos tipos.

Tipos de dados	Exemplos de Constantes		
int	1	1200	- 24
long int	3500L	- 34L	15L
short int	10	- 12	98
float	123.23F	0.5F	4.34e - 3F
double	120.12	12345678	- 0.987321
caractere	'x'	'a'	'1'

Tabela 4: Exemplos de Constantes. **Fonte:** Elaboração Própria.

Constantes String

Outro tipo de constante que o **C** suporta é a constante do tipo **string**. Uma string é um conjunto de caracteres colocado entre duas aspas duplas (""). Por exemplo: "*Cliente Especial*". Um comum de *string* é no comando `printf`, no qual, por exemplo podemos ter o seguinte comando `printf("Aluno Aprovado")`.

Trabalhando Operadores

A linguagem **C** é muito rica no tratamento de operadores, em que podemos encontrar quatro classes de operadores: aritméticos, relacionais, lógicos e bit a bit, além de um operador especial: Operador de atribuição.

Operadores de Atribuição

A linguagem **C** disponibiliza operadores de atribuição, que podem ser utilizados dentro de qualquer expressão válida em **C**. A sintaxe do operador de atribuição é dada por:

Nome_da_variável = expressão;

Em que expressão pode ser tão simples como uma única constante, ou tão complexa quanto necessária. É importante destacar que o **destino** (parte esquerda da atribuição), deve ser uma variável ou um ponteiro, e nunca uma função ou uma constante. Em **C** é permitido que se atribua o mesmo valor a muitas variáveis usando atribuições múltiplas em um único comando. Podemos observar na Tabela 5 exemplo de atribuições.

Atribuição	Tipos de Atribuições
codigoAluno = 1234;	Literal numérica
sexoAluno = 'F';	Tipo caractere
desconto = calculaDesconto();	Atribuição com valor retornado de uma função
strcpy(linguagem, "linguagem C");	Atribuição para uma variável do tipo string (char linguagem [20];)
salario = proventos - desconto;	Atribuição a partir de uma expressão aritmética
valor1 = valor2 = valor3 = 1000.00;	Atribuição do valor 1000,00 para múltiplas variáveis em um só comando

Tabela 5: Exemplos de Atribuição em C. **Fonte:** Elaboração Própria.

FIQUE ATENTO

Na atribuição de *string* não é permitida a atribuição direta, ou seja, para se fazer a atribuição, deve ser utilizado uma função que a linguagem **C** disponibiliza: **strcpy()**

Observe dois Exemplos: errado e correto

```
char linguagem[20];  
linguagem = "linguagem C"; /* ilegal em C */  
strcpy( linguagem, "linguagem C"); /* legal */
```

Para a primeira atribuição, o compilador aponta erro de compilação.

Operadores Aritméticos

Temos na linguagem **C** os cinco operadores aritméticos básicos: **+** (adição), **-** (subtração), ***** (multiplicação) e **/** (divisão) e mais os operadores especiais: **%** (módulo da divisão-resto), **++** (incremento) e **--** (decremento). Podemos usar mais de um operador na mesma expressão. A precedência é igual à usada na matemática comum.

Na Tabela 6 podemos observar alguns exemplos de operadores aritméticos com seus respectivos resultados.

Operação	Resultado da Operação
operacao1 = 20 + 40 * 10;	420
operacao2 = 20 + 400 / 2 + 50;	270
operacao3 = (20 + 40) * 10;	600
operacao4 = (1200 + 200) / (30 + 70);	14
operacao5 = 10 * 5 * 4 - 3 - 1;	196

Tabela 6: Exemplos de Operadores Aritméticos. **Fonte:** Elaboração Própria.

Operadores Aritméticos de Incremento e Decremento

O incremento é representado pelos sinais **++**, ou seja, ele **umenta em 1** o valor da variável que ele está acompanhando (seu operando). O conceito é semelhante para o sinal de **--**, com a diferença que ele **diminui** em uma unidade o valor da variável que acompanha (seu operando).

Podemos observar nos trechos de código abaixo os exemplos de incremento e decremento:

```
int iValor= 0;
iValor++;    //Neste momento iValor = 1
//É o mesmo que:
int iValor = 0;
iValor = iValor + 1;    //Neste momento iValor = 1

int dValor= 10;
dValor--;    //Neste momento dValor = 9
//É o mesmo que:
int dValor = 10;
dValor = dValor -1;    //Neste momento dValor = 9
```

Figura 11: Operadores Artméticos de Incremento e Decremento. **Fonte:** Elaboração Própria.

FIQUE ATENTO

Na atribuição por incremento e decremento, tome muito cuidado, pois podemos ter o pré-incremento e pós-incremento assim como temos também pré-decremento e pós-decremento.

Exemplos: Suponha a variável $c = 6$ para todos os exemplos;

$i = ++c$; o valor é incrementado, e só depois passado para i ($i=7$)

$i = c++$; o valor passado para i e somente depois é incrementado ($i=6$)

$d = --c$; o valor é decrementado, e só depois passado para d ($d=5$)

$d = c--$; o valor passado para d e somente depois é decrementado ($d=6$)

Operadores Relacionais e Lógicos

Quando falarmos Operador Relacional, o termo **relacional** refere-se às relações que os valores podem ter uns com os outros. E, ao falarmos Operador Lógico, o termo **lógico** se refere às maneiras como essas relações podem ser conectadas.

É muito importante o domínio do conceito da lógica booleana (verdadeiro ou falso), pois, este conceito

é a base dos conceitos dos operadores lógicos e relacionais.

Na Tabela 7 Operadores Relacionais e Operadores Lógicos. Fonte: Elaboração Própria." na página 30 temos a relação dos símbolos utilizados pela linguagem **C** para representar os operadores relacionais e lógicos.

Operadores Relacionais		Operadores Lógicos	
==	igualdade	&&	e lógico
!=	diferença		
>	maior que		ou lógico
<	menor que		
>=	maior ou igual a	!	não lógico
<=	menor ou igual a		

Tabela 7: Operadores Relacionais e Operadores Lógicos. **Fonte:** Elaboração Própria.

Para saber mais sobre os operadores lógicos e a tabela verdade, ouça o **Podcast – “Operadores Lógicos de Boole”**.

Podcast 1



O código do programa abaixo é um exemplo da utilização de operadores relacionais e operadores lógicos. Podemos observar a utilização dos operadores relacionais **“Maior que”**, **“Menor que”**, **“Maior ou igual a”** e o operador lógico **&&** (E logico)

```

#include <stdio.h>
int main()
{
    int iJ, iI;
    iJ = 18;    iI = 60;
    int idade = 59;
    if( (idade > iJ)&&(idade < iI) ){
        //se isso for V ou se isso for V
        //A meia-idade se caracteriza por qualquer pessoa que tenha
        //menos de 60 anos e mais de 17
        printf("A idade informada é de uma pessoa de meia-idade.");
    }else{
        if(idade >= iI){
            printf( "\nA idade informada é de uma pessoa idosa.\n" );
        }else{
            printf("\nA idade informada é de um jovem.\n");
        }
    }
    return 0;
}

```

Figura 12: Operadores relacionais e Operadores lógicos. **Fonte:** Elaboração Própria.

O Operador Ternário

A linguagem C possui um operador que possibilita substituir certas sentenças *if-then-else*. É o operador ternário **?**, e que possui a seguinte sintaxe:

Exp1 ? Exp2 : Exp3;

Onde **Exp1**, **Exp2** e **Exp3** são expressões e têm o seguinte comportamento: Exp1 é avaliada. Se for verdadeira, então Exp2 é avaliada e se torna o valor da expressão. Se **Exp1** for falsa, então **Exp3** é avaliada e se torna o valor da expressão.

Observe o seguinte exemplo:


```
#include <stdio.h>
int main(void) {
}

int cod = 101;
int resultado = cod > 100 ? 200 : 300;
printf("Valor de resultado %d\n", resultado);
return 0;
}
```

Figura 13: Exemplo de Operador Ternário. **Fonte:** Elaboração Própria.

À variável **resultado** é atribuído o valor de 200 pois, neste caso, **Exp1** é positivo. Se **cod** fosse menor do que 100, a variável **resultado** teria recebido o valor 300.

Observe o código abaixo, resolvendo o mesmo problema, com a estrutura “if-then-else”:

```
#include <stdio.h>
int main(void)
{
    int cod = 101;
    int resultado;
    if (cod > 100)
    {
        resultado = 200;
    } else {
        resultado = 300;
    }
    printf("Valor de resultado %d\n", resultado);
    return 0;
}
```

Figura 14: Exemplo com estrutura “if-then-else”. **Fonte:** Elaboração Própria.

Resolvendo juntos um problema prático

Caro estudante, convido você a resolvermos juntos um problema proposto, para que você pratique os conceitos estudados até o momento. Para este fim, propomos um problema de simples solução:

“Uma determinada escola, cujo cálculo da média é realizado com as quatro notas bimestrais que determinam a aprovação ou reprovação dos seus alunos. Considere ainda que o valor da média deve ser maior ou igual a 7 para que haja aprovação.”

Para resolver este problema, vamos praticar os passos que aprendemos para a resolução sistemática de um problema computacional.

Passo 1: Entendimento macro do problema

Neste passo, você deve obter um entendimento preciso da **questão central** do problema para, posteriormente, poder fazer o detalhamento dele. Neste momento, entendemos que o problema central é a **apuração do resultado final do rendimento escolar de cada aluno**. Tendo este entendimento, você desce mais um nível e verifica que seu programa deverá informar a situação do aluno como “Aprovado” ou “Reprovado”. Descendo mais um nível, você descobre que o que determina a situação do aluno é um cálculo composto por suas notas durante o período, e que

este cálculo é a média aritmética de quatro notas. Portanto, você já tem os elementos para fazer seu algoritmo.

■ Passo2: Elaboração do algoritmo

Agora você já possui todos os elementos que compõem seu problema, então, vamos ao algoritmo:

- Solicitar a entrada das quatro notas do aluno
- Calcular a média aritmética para as quatro notas
- Verificar se a média é inferior à 7
- Para a média menor do que 7 informar que o aluno foi **“Reprovado”**
- Para a média maior ou igual a 7 informar que o aluno foi **“Aprovado”**

Cabe ressaltar que, neste momento, você deverá ter o domínio das **“regras de negócio”**. Em nosso caso, você deverá ter o domínio de que, para se obter a média aritmética das notas, você deverá implementar a seguinte equação:

$$Media = \frac{(N1 + N2 + N3 + N4)}{n}$$

Onde:

Media = Média das quatro notas

N1 = Nota da **Avaliação 1** que o aluno obteve no período

N2 = Nota da **Avaliação 2** que o aluno obteve no período

N3 = Nota da **Avaliação 3** que o aluno obteve no período

N4 = Nota da **Avaliação 4** que o aluno obteve no período

n = Quantidade de avaliações do período

O algoritmo desenvolvido em pseudolinguagem abaixo:

1	PROGRAMA TrocaValores
2	INICIO
3	Criar VARIÁVEIS
4	N1; N2; N3, N4, Media tipo real
5	EXIBIR ("Digite o valor de N1:")
6	LEIA teclado e atribua o valor a N1
7	EXIBIR ("Digite o valor de N2:")
8	LEIA teclado e atribua o valor a N2
9	EXIBIR ("Digite o valor de N3:")
10	LEIA teclado e atribua o valor a N3
11	EXIBIR ("Digite o valor de N4:")
12	LEIA teclado e atribua o valor a N4
13	$Media = (N1 + N2 + N3 + N4) / 4$
14	Se (Media >= 7) então
15	EXIBIR ("Aprovado")
16	Senão
17	EXIBIR ("Reprovado")
18	FIM

SAIBA MAIS

“Regras do Negócio são declarações sobre a forma de a empresa fazer negócio. Elas refletem as políticas do negócio. Organizações têm políticas para satisfazer os objetivos do negócio, satisfazer clientes, fazer bom uso dos recursos, e obedecer às leis ou convenções gerais do negócio. Regras do Negócio tornam-se requisitos, ou seja, podem ser implementados em um sistema de software como uma forma de requisitos de software desse sistema”. (LEITE; LEONARDI, 1998)

Neste momento, não se afobe para já sair codificando: Lembra-se do teste de mesa? Com essa “ferramenta” você poderá validar se a sua solução contempla todas as necessidades propostas no problema. Então, mãos a obra:

Vamos criar uma tabela com linhas e colunas em que:

- Cada coluna represente uma variável
- Criamos coluna para representar o(s) dispositivo(s) de saída (Vídeo/Arquivo)
- Então, seguiremos cada instrução para refletir as alterações de memória nas respectivas colunas (variáveis).
 - Atribuiremos valores hipotéticos para as quatro notas

Observe o resultado de nosso teste de mesa, na Tabela 8:

Estados									
Memória						Entrada		Teste(if)	Saída
Linha	N1	N2	N3	N4	Média	LEIA	Teclado		Vídeo
5	--	--	--	--	--	--	--	--	Digite a nota N1
6	6,5	--	--	--	--	6,5	6,5	--	Digite a nota N1
7	6,5	--	--	--	--	--	--	--	Digite a nota N2
8	6,5	7	--	--	--	7	7	--	Digita a nota N2
9	6,5	7	--	--	--	--	--	--	Digita a nota N3
10	6,5	7	5,5	--	--	5,5	5,5	---	Digita a nota N3
11	6,5	7	5,5	--	--	--	--	--	Digita a nota N4
12	6,5	7	5,5	9	--	9	9	--	Digita a nota N4
13	6,5	7	5,5	9	7,00	--	--	--	--
14	6,5	7	5,5	9	7,00	--	--	VERDADEIRO	--
15	6,5	7	5,5	9	7,00	--	--	--	"Aprovado"
17	--	--	--	--	--	--	--	--	--

Tabela 8: Teste de mesa do nosso algoritmo. **Fonte:** Elaboração Própria.

Passo3: Diagramação do Algoritmo

Agora que já temos um bom domínio da solução do problema, com sua particularidade, vamos diagramar para obter um refinamento da solução e, assim, ter mais segurança na implementação. Conforme podemos observar na Figura 15, as quatro notas são

solicitadas para que o operador as informe pelo teclado. Logo em seguida, as quatro notas são utilizadas no cálculo da média aritmética, conforme regra de negócio. Em seguida, é feita a apuração do resultado da situação do aluno.

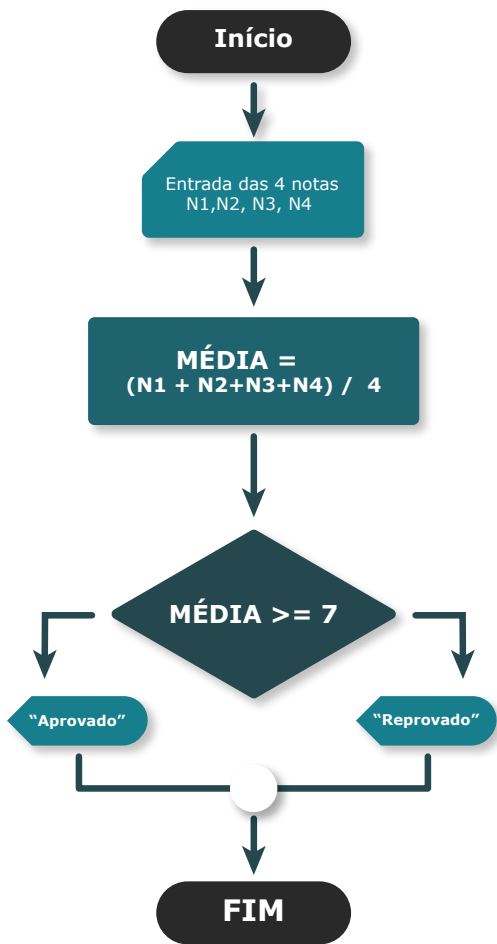


Figura 15: Diagramação da Solução do programa que Apura a situação do Aluno. **Fonte:** Elaboração Própria.

Passo 4: Codificação do Programa

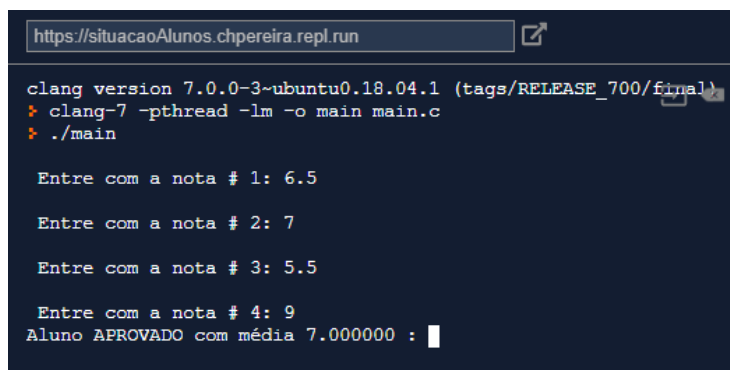
Agora que já temos o domínio completo da solução do problema, com suas particularidades obtidas nos processos percorrido até a diagramação, vamos implementar na linguagem **C**, com foco nos conceitos aprendidos nesta unidade.

```
1  /*-----*
2  *PROGRAMA: situacaoAluno.cpp - Verifica a situação final
3  *do aluno a partir de suas quatro notas obtidas no período
4  *AUTOR: FAM Online
5  *DATA: Junho/20xx - Versão 1.0
6  *-----*/
7  #include <stdio.h>
8  float entradaDados(int numeroEntrada)
9  {
10     /* O Objetivo desta função é obter uma entrada de entradaDados.
11        Em nosso caso irá obter uma nota de aluno digitado pelo teclado */
12     float nota;
13     printf("\n Entre com a nota # %d: ", numeroEntrada);
14     scanf("%f",&nota);
15     return nota;
16 }
17 /* Aqui cimeça o programa - Função main() - Todo programa em C começa por aqui
18 int main(void)
19 {
20     float n1, n2, n3, n4, media; //Declara as variaveis para receber as notas
21
22     n1 = entradaDados(1); //chama função para obter a nota #1
23     n2 = entradaDados(2); //chama função para obter a nota #2
24     n3 = entradaDados(3); //chama função para obter a nota #3
25     n4 = entradaDados(4); //chama função para obter a nota #4
26
27     media = (n1 + n2 + n3 + n4) / 4;
28
29     if (media >= 7.0)
30     {
31         printf("Aluno APROVADO com média %f : ", media);
32     } else {
33         printf("Aluno REPROVADO com média %f : ", media);
34     }
35     return 0;
36 }
```

Figura 16: Codificação do Programa. **Fonte:** Elaboração Própria.

Passo 5: Teste da solução do problema implementada em C

Agora faça os testes com as mais variadas situações (cenários). Por exemplo, entre com notas em que o aluno será reprovado ou aprovado. Explore também os testes de limite, ou seja, com a nota 7,0. Em nosso caso, a figura mostra somente um teste. Neste caso, note que reproduzimos o mesmo teste feito em nosso teste de mesa.



```
https://situacaoAlunos.chpereira.repl.run

clang version 7.0.0-3~ubuntu0.18.04.1 (tags/RELEASE_700/final)
> clang-7 -pthread -lm -o main main.c
> ./main

Entre com a nota # 1: 6.5

Entre com a nota # 2: 7

Entre com a nota # 3: 5.5

Entre com a nota # 4: 9
Aluno APROVADO com média 7.000000 : █
```

Figura 17: Teste final da solução implementa na linguagem C. **Fonte:** Elaboração Própria.

Para saber mais sobre qualidade de software e a produção de programas com qualidade, ouça o Podcast “A engenharia de software com qualidade”.

Podcast 2



CONSIDERAÇÕES FINAIS

Caro estudante, analisamos juntos a diversidade de linguagens de programação que proliferaram nestes últimos tempos. Observamos que este fenômeno se deu por conta vertiginosa do desenvolvimento tecnológico (processadores e periféricos). Aprendemos a utilizar um ambiente de desenvolvimento on-line. Isso foi muito importante para você se concentrar somente no aprendizado dos paradigmas da programação e de implementá-los em uma linguagem de programação muito utilizada pelo mercado de desenvolvedores. Adotamos uma linguagem específica; em nosso caso, a **C**, e entramos nos princípios básicos desta, para termos sólidos conhecimentos e poder progredir para posteriores conceitos avançados da programação. Você aprendeu a manipular variáveis de memória e atribuir a ela os tipos de dados necessários para processá-los. Você aprendeu a manipular os operadores aritméticos, relacionais e lógicos, por serem a alma de todas as linguagens de programação.

Com esses conhecimentos básicos, você está preparado a dar continuidade nos conceitos avançados da programação de computadores e implementá-los em qualquer linguagem (que, como estudamos, não são poucas). Por fim, resolvemos e implementamos, juntos, um problema computacional, utilizando as

técnicas de resolução, desde o entendimento até a codificação em linguagem **C**.

Parabéns por ter chegado até este ponto, pois a jornada só está começando, mas você está plenamente fortalecido para lograr êxito no campo da programação de computadores.

Síntese



ELEMENTOS BÁSICOS DA LÓGICA DE PROGRAMAÇÃO

Nesta unidade aprendemos a passar do conceitual para a prática: Do entendimento do problema ao teste do programa. Aprendemos a implementar a solução do problema em uma linguagem de programação muito utilizada pelo mercado: A linguagem de programação C. Mostramos as principais linguagens de programação da atualidade, bem como abordamos os ambientes de programação nesta linguagem. Exploramos a estrutura da linguagem de programação C, e seus elementos básicos da lógica de programação, implementados na referida linguagem. Abordamos a manipulação de variáveis, manipulação dos operadores aritméticos, lógicos e relacionais. Aprendemos também a declarar e a utilizar função. Finalizamos com um caso prático de programação, utilizando um ambiente online (Acesso à internet).

Para este fim, estruturamos este material da seguinte forma:

A escolha de linguagem de programação

As Principais Linguagens de desenvolvimento

Transformado o Algoritmo em Código

Uma Visão Geral da Linguagem C

Primeiros passos em linguagem C

O ambiente de desenvolvimento em linguagem C

Expressões

Manipulação de Dados em Memória

Nomes de identificadores de Dados

Variáveis

Trabalhando Operadores

Operadores de Atribuição

Operadores Aritméticos

Operadores Aritméticos de Incremento e decremento

Operadores Relacionais e Lógicos

O Operador Ternário

Resolvendo, juntos, um problema prático

Entendimento macro do Problema

Diagramação do Algoritmo

Codificação do Programa

Teste da solução do Problema Implementada em C

Referências

ALVES, W. P. **Linguagem e Lógica de Programação**. São Paulo: Érica, 2015.

EDELWEIS, N.; LIVI, M. A. C.; LOURENÇO, A. E. **Algoritmos e programação com exemplos em Pascal e C**. São Paulo: Bookman, 2014.

FORBELLONE, A. L. V; EBERSPACHER, H. F. **Lógica de Programação**: a construção de algoritmos e estruturas de dados. São Paulo: Prentice Hall, 2015.

LEITE, J.C.S.P.; OLIVEIRA, A. P. **A client oriented Requirements Baseline**. In Proceedings of the Second IEEE International Symposium on Requirements Engineering, IEEE Computer Society Press, pp. 108-115 (1995).

LEITE, J.C.S.P., LEONARDI, C. **Business rules as organizational policies**. Proceedings of IEEE Ninth International Workshop Specification and Design. Los Alamitos, CA: IEEE Computer Society Press (1998).

MANZANO, J. A. N. G; MATOS, E; LOURENÇO, A. E. **Algoritmos**: Técnicas de Programação. São Paulo: Érica, 2015.

MANZANO, J. A. N. G; OLIVEIRA, J. F. **Algoritmos:** Lógica para desenvolvimento de programação de computadores. São Paulo: Saraiva, 2016

PERKOVIC, L. **Introdução à computação usando Python.** Rio de Janeiro: LTC-Livros Técnicos e Científicos, 2016.

PUGA, S.; RISSETTI, G. **Lógica de programação e estrutura de dados:** com aplicação em Java. São Paulo: Pearson Education, 2004.

ZIVIANI, N. **Projeto de Algoritmos:** Com implementações em Pascal e C. São Paulo: Cengage Learning, 2011.

FaM
ONLINE