



Programação de computadores

Programação é o processo de escrita, teste e manutenção de um programa de computador. O programa é escrito em uma linguagem de programação, embora seja possível, com alguma dificuldade, o escrever diretamente em linguagem de máquina. Diferentes partes de um programa podem ser escritas em diferentes linguagens.

Diferentes linguagens de programação funcionam de diferentes modos. Por esse motivo, os programadores podem criar programas muito diferentes para diferentes linguagens; muito embora, teoricamente, a maioria das linguagens possa ser usada para criar qualquer programa.

Há várias décadas se debate se a programação é mais semelhante a uma arte (Donald Knuth), a uma ciência, à matemática (Edsger Dijkstra), à engenharia (David Parnas), ou se é um campo completamente novo.

Algoritmos

Um algoritmo é uma sequência lógica finita de passos para realizar uma tarefa ou resolver um problema. Em nosso dia a dia utilizamos algoritmos para realizar nossas atividades, definindo a sequência de atividades que devemos fazer para atingir um objetivo. Um exemplo simples é uma receita. Um algoritmo é, num certo sentido, um programa abstrato — dizendo de outra forma, um programa é um algoritmo concretizado. Os programas são visualizados mais facilmente como uma coleção de algoritmos menores combinados de um modo único — da mesma forma que uma casa é construída a partir de componentes.^[1]

Dessa forma, um algoritmo é uma descrição passo a passo de como o computador irá executar uma operação específica, como, por exemplo, uma ordenação. Um programa, por outro lado, é uma entidade que na verdade implementa uma ou mais operações de forma que seja útil para as pessoas que o utilizam.^[1]

Engenharia de software

```
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <math.h>
4
5 int main(int argc, char *argv[])
6 {
7     int num, sr, flag, i;
8
9     if (argc != 2) return 1;
10    num = atoi(argv[1]);
11    sr = (int)sqrt(num);
12    if (num < 2)
13        flag = 0;
14    else
15    {
16        flag = 1;
17        for (i=2; i<=sr; i++)
18            if (num%i == 0)
19            {
20                flag = 0;
21                break;
22            }
23    }
24    if (flag) printf("%d e' primo\n", num);
25    else printf("%d nao e' primo\n", num);
26    return 0;
27 }
```

Pequeno programa na linguagem de programação C que imprime na tela se o número passado a ele como argumento é primo ou não. O código fonte está sendo visualizado em um IDE com suporte a coloração de sintaxe e indentação de código.

A criação de um programa de computador consiste de cinco passos principais:

1. Reconhecer a necessidade de um programa para resolver um problema ou fazer alguma coisa
2. Planejar o programa e selecionar as ferramentas necessárias para resolver o problema
3. Escrever o programa na linguagem de programação escolhida
4. Compilação: tradução do código fonte legível pelo homem em código executável pela máquina, o que é feito através de compiladores e outras ferramentas
5. Testar o programa para ter a certeza de que funciona; se não, regressar ao passo 3

Estes cinco passos são colectivamente conhecidos como engenharia de software. A programação põe ênfase nos passos 2, 3 e 4. A codificação põe ênfase no passo 3. O termo *coder*, por vezes usado como sinónimo para programador, pode tornar-se aviltante porque ignora as capacidades necessárias para lidar com os outros quatro passos.

História

Heron de Alexandria no século primeiro inventou teatros automatizados que usavam programação análoga para controlar os fantoches, portas, luzes e efeitos de som.

A mais antiga programadora de computadores que se conhece é Ada Lovelace, filha de Anabella e de Lord Byron (o poeta). Ao serviço do matemático Charles Babbage, traduziu e expandiu uma descrição da sua máquina analítica. Muito embora Babbage nunca tenha completado a construção de nenhuma das suas máquinas, o trabalho que ele e Ada desenvolveram sobre elas, garantiu a Ada o título de primeira programadora de computadores do mundo (veja as notas de Ada Byron sobre a máquina analítica).^[2] A linguagem de programação Ada recebeu o seu nome em homenagem à Ada.^[3]



Um bug (falha), que foi depurado em 1947.

Um dos primeiros programadores que se tem notícia de ter completado todos os passos para a computação sem auxílio, incluindo a compilação e o teste, é Wallace J. Eckert. O trabalho deste homem antecede a ascensão das linguagens de computador, porque ele usou a linguagem da matemática para solucionar problemas astronômicos. No entanto, todos os ingredientes estavam lá: ele trabalhou um laboratório de computação para a Universidade de Colúmbia com equipamentos fornecidos pela IBM, completos com uma divisão de serviço de atendimento ao cliente, e consultores de engenharia para propósitos especiais, na cidade de Nova York, na década de 1930, usando cartões perfurados para armazenar os resultados intermediários de seus cálculos, e então formatando os cartões perfurados para controlar a impressão das respostas, igual ao trabalho para os censos décadas antes. Tinha técnicas de *debug* tais como códigos de cores, bases cruzadas, verificação e duplicação. Uma diferença entre Eckert e os programadores dos dias de hoje é que o exemplo do seu trabalho influenciou o projeto Manhattan. Seu trabalho foi reconhecido por astrônomos do Observatório da Universidade de Yale, Observatório da

Universidade de Princeton, Observatório da Marinha dos EUA, Observatório da Faculdade Harvard, Observatório dos estudantes da Universidade da Califórnia, Observatório Ladd da Universidade de Brown e Observatório Sproul da Faculdade de Swarthmore.

Alan Turing é frequentemente encarado como o pai da ciência de computadores e, por afinidade, da programação. Ele foi responsável por ajudar na elaboração e programação de um computador destinado a quebrar o código alemão ENIGMA durante a Segunda Guerra Mundial — ver Máquina Enigma.

Aprendizagem da Programação

A aprendizagem da programação tem enfrentado vários desafios. Por ser de difícil aprendizagem, vários estudos propõe soluções para ajudar no processo de aprendizagem da programação, quer a nível do ensino secundário, quer universitário^[4] por diversas razões.^{[5][6][7]} De entre as soluções, destacam-se sistemas de apoio,^[8] uns que permitem que os estudantes visualizem de imediato o resultado do código que vão escrevendo,^[9] outros estudos também sugerem o uso de artefatos como a robótica para que os alunos interajam com algo tangível como o robot, melhorando a interação e motivando ao mesmo tempo.^[10] Foram realizados estudos que provam que o uso da gamificação^{[11][12]} em contextos de aprendizagem da programação, produziu resultados com sucesso,^[13] aumentando o nível de interação dos alunos, bem como a motivação para continuar a aprender.^{[14][15][15]}

Ver também

- Callback
- Ciência da computação inovadora
- Documentação de software actual
- Engenharia de software
- Falha de segmentação
- Linguagem de programação
- Lista de linguagens de programação
- Orientação a objetos
- Programação baseada em ARS
- Programação estruturada
- Programação funcional
- Programação imperativa
- Programação orientada a aspecto
- Programação orientada por acontecimentos
- Software
- Testes de caixa negra

Referências

1. Moura, Arnando V. (3 de Novembro de 2009). «MC102 - Algoritmos» (<http://www.ic.unicamp.br/~mc102/algoritmos.html>) (PDF). Instituto de Computação da Universidade Estadual de Campinas. Consultado em 17 de Agosto de 2017

2. Fuegi, J.; Francis, J. (2003). «Lovelace & babbage and the creation of the 1843 'notes' ». *IEEE Annals of the History of Computing*. **25** (4). 16 páginas. doi:10.1109/MAHC.2003.1253887 (<https://dx.doi.org/10.1109%2FMAHC.2003.1253887>)
3. «The Ada Programming Language» (<https://web.archive.org/web/20160522063844/http://groups.engin.umd.umich.edu/CIS/course.des/cis400/ada/ada.html>) (em inglês). Universidade de Michigan. Consultado em 17 de agosto de 2017. Arquivado do original (<http://groups.engin.umd.umich.edu/CIS/course.des/cis400/ada/ada.html>) em 22 de maio de 2016
4. Ferreira, Fabio; Costa, Carlos J.; Aparicio, Manuela; Aparicio, Sofia (junho de 2017). «Learning programming: A continuance model» (<https://ieeexplore.ieee.org/document/7975815/?reload=true>). IEEE. *2017 12th Iberian Conference on Information Systems and Technologies (CISTI)* (em inglês). ISBN 9789899843479. doi:10.23919/cisti.2017.7975815 (<https://dx.doi.org/10.23919%2Fcisti.2017.7975815>)
5. Costa, Carlos (2012). «Web-Based graphic environment to support programming in the beginning learning process» (https://link.springer.com/chapter/10.1007/978-3-642-33542-6_41). International Conference on Entertainment Computing (pp. 413-416)
6. Piteira, Martinha; Costa, Carlos (11 de junho de 2012). «Computer programming and novice programmers» (<http://dl.acm.org/citation.cfm?id=2311917.2311927>). ACM: 51–53. ISBN 9781450312943. doi:10.1145/2311917.2311927 (<https://dx.doi.org/10.1145%2F2311917.2311927>)
7. Piteira, Martinha; Costa, Carlos (11 de julho de 2013). «Learning computer programming: study of difficulties in learning programming» (<http://dl.acm.org/citation.cfm?id=2503859.2503871>). ACM: 75–80. ISBN 9781450322997. doi:10.1145/2503859.2503871 (<https://dx.doi.org/10.1145%2F2503859.2503871>)
8. Pierce, Robert; Aparício, Manuela (8 de novembro de 2010). «Resources to support computer programming learning and computer science problem solving» (<http://dl.acm.org/citation.cfm?id=1936755.1936766>). ACM: 35–40. ISBN 9781450304801. doi:10.1145/1936755.1936766 (<https://dx.doi.org/10.1145%2F1936755.1936766>)
9. Costa, Carlos J.; Aparicio, Manuela; Cordeiro, Carlos (11 de junho de 2012). «A solution to support student learning of programming» (<http://dl.acm.org/citation.cfm?id=2316936.2316942>). ACM: 25–29. ISBN 9781450315258. doi:10.1145/2316936.2316942 (<https://dx.doi.org/10.1145%2F2316936.2316942>)
10. Aparicio, Joao Tiago; Costa, Carlos J. (junho de 2018). «A virtual robot solution to support programming learning an open source approach» (<https://ieeexplore.ieee.org/document/8399263/?reload=true>). IEEE. *2018 13th Iberian Conference on Information Systems and Technologies (CISTI)* (em inglês). ISBN 9789899843486. doi:10.23919/cisti.2018.8399263 (<https://dx.doi.org/10.23919%2Fcisti.2018.8399263>)
11. Costa, Carlos J.; Aparicio, Manuela; Aparicio, Sofia; Aparicio, Joao Tiago (11 de agosto de 2017). «Gamification usage ecology» (<http://dl.acm.org/citation.cfm?id=3121113.3121205>). ACM. 2 páginas. ISBN 9781450351607. doi:10.1145/3121113.3121205 (<https://dx.doi.org/10.1145%2F3121113.3121205>)
12. Piteira, Martinha; Costa, Carlos J.; Aparicio, Manuela (6 de abril de 2018). «Computer Programming Learning: How to Apply Gamification on Online Courses?» (<https://doi.org/10.20897/jisem.201811>). *Journal of Information Systems Engineering & Management* (em inglês). **3** (2). ISSN 2468-4376 (<https://www.worldcat.org/issn/2468-4376>). doi:10.20897/jisem.201811 (<https://dx.doi.org/10.20897%2Fjisem.201811>)
13. Costa, Carlos J.; Aparicio, Manuela (16 de maio de 2014). «Evaluating success of a programming learning tool» (<http://dl.acm.org/citation.cfm?id=2618168.2618180>). ACM: 73–78. ISBN 9781450327138. doi:10.1145/2618168.2618180 (<https://dx.doi.org/10.1145%2F2618168.2618180>)

14. Pereira, Ricardo; Costa, Carlos J.; Aparicio, Joao Tiago (junho de 2017). «Gamification to support programming learning» (<https://ieeexplore.ieee.org/document/7975788/?reload=true>). IEEE. *2017 12th Iberian Conference on Information Systems and Technologies (CISTI)* (em inglês). ISBN 9789899843479. doi:10.23919/cisti.2017.7975788 (<https://dx.doi.org/10.23919%2Fcisti.2017.7975788>)
15. Piteira, Martinha; Costa, Carlos J. (junho de 2017). «Gamification: Conceptual framework to online courses of learning computer programming» (<https://ieeexplore.ieee.org/document/7975695/?reload=true>). IEEE. *2017 12th Iberian Conference on Information Systems and Technologies (CISTI)* (em inglês). ISBN 9789899843479. doi:10.23919/cisti.2017.7975695 (<https://dx.doi.org/10.23919%2Fcisti.2017.7975695>)

Ligações externas

- «A História da Programação de Computadores» (<http://www.superempreendedores.com/internet/a-historia-da-programacao-de-computadores>)

Obtida de "https://pt.wikipedia.org/w/index.php?title=Programação_de_computadores&oldid=68708885"