code4sp
coding for social promotion

Co-funded by the
Erasmus+ Programme
of the European Union

1.:

Pacote de Material de Formação Code4SP

WP3:

Materiais de Formação Code4SP

Preparado por:

CD
DR

Informação do Projeto

Acrónimo do projeto: Code4SP

Título do projeto: Coding for Social Promotion

Referência do projeto: 621417-EPP-1-2020-1-PT-EPPKA3-IPI-SOC-IN

Website do projeto: www.code4sp.eu

Parceiro autor: CodeDoor

Versão do documento: 1

Data de preparação: 25/02/2022

Histórico do Documento			
Data	Versão	Autor	Descrição
25/02/2022	1	CodeDoor	Esboço

Tabela de Conteúdos

Tópico:

1. Programação de Computadores e seus conceitos básicos

Pré-requisitos:

Literacia básica de computadores, software básico instalado e conhecimento básico de trabalhar com ficheiros.

Carga de trabalho:

5 horas.

Descrição:

Neste tópico, faremos uma breve introdução à programação de computadores e aos seus conceitos básicos. Trataremos de conceitos de hardware e software, bem como da forma como os computadores interpretam os dados. No final, enfatizaremos a forma como um programa funciona e que tipo de linguagens de programação existem.

Resultados da aprendizagem:

- Reconhecer o conceito de HyperText Markup Language (HTML) na família de linguagens de descrição de documentos.
- Distinguir entre estrutura, conteúdo e estilos de página.
- Usar HTML na construção de páginas para a Web

Material necessário:

- Computador ou portátil
- Ligação à Internet

Cenário de Aula:

O tempo total para este tópico é de 5 horas, e caberá ao formador decidir quanto tempo dedicar ao ensino de cada subtópico. A fim de aproveitar ao máximo todo o tempo disponível, propomos a utilização dos materiais de formação produzidos pelo projeto (apresentações PPT), que foram concebidos tendo em mente uma utilização eficaz do tempo. Estas apresentações são compostas pelos seguintes elementos:

- Desenvolvimento do subtópico e principais ideias a reter;
- Atividades/Exercícios propostos.

Dito isto, se o/a formador/a seguir a sequência lógica dos PPTs, certamente conseguirá completar a sessão dentro do prazo estipulado. Essas apresentações também podem ser disponibilizadas ao corpo discente para estudo individual.

Subtópicos:

1.1. Introdução aos computadores e à programação

1.2. Conceito de Hardware e Software

1.3. Como os computadores armazenam dados

1.4. Como funciona um programa

1.5. Programação e Linguagens de Programação

Recursos adicionais:

- Khan Academy: recursos úteis em várias línguas
- Marc Andreessen: https://future.a16z.com/software-is-eating-the-world/

1.1. Computadores e Programação – introdução

Os computadores são máquinas que podem ser programadas para executar uma sequência de instruções. A programação é o processo através do qual essas instruções se materializam. Os programas são construídos numa linguagem particular, que fornece uma estrutura para o programador e utiliza instruções específicas para controlar a sequência de operações que o computador executa. Citando Marc Andreesen, "*o software está a comer o mundo*", por isso é importante alertar a sua plateia discente para o facto de que, nos dias de hoje, quase tudo se encontra informatizado e programado (veja também esta página informativa acerca da Internet das Coisas).

Quais algumas formas de utilizar os computadores?

Permita que a turma faça um *brainstorming* ou que crie um mapa mental acerca dos vários propósitos de um computador. São várias as formas de utilizar o computador no nosso dia-a-dia. Algumas delas seguem na lista abaixo:

- Navegar na Internet
- Para verificar o correio eletrónico
- Fazer pesquisas
- Jogar videojogos
- Assistir a filmes ou a programas de TV
- Ouvir música
- Trabalhar num artigo ou folha de cálculo
- Ler as notícias

Que outros dispositivos são também computadores?

Os computadores não são apenas os dispositivos que se encontram nas nossas secretárias ou nos nossos bolsos. São também os dispositivos que fazem funcionar os

nossos carros, aviões e barcos. Estão nos nossos televisores, frigoríficos, e mesmo nos nossos relógios. Em suma, os computadores estão em todo o lado.

"Que software têm utilizado?"

Organize a turma de forma a levar a cabo uma sessão de *brainstorming* ou de desenho de um mapa mental sobre qual(is) o(s) software(s) que os seus integrantes já tenham, porventura, utilizado. Procure informar a turma que cada *software* foi desenvolvido por um determinado programador, seja o *software* que dá vida aos *smartphones*, máquinas de ATM ou até mesmo os televisores que ligam diariamente. Todos estes *softwares* foram desenvolvidos por um/a programador/a ou conjunto de programadores/as.

1.2. Conceito de *hardware* e *software*

Definição de *hardware* e *software*

Hardware refere-se ao conjunto de componentes físicos de um sistema informático, enquanto que o *software* se refere às instruções e dados que fazem o computador funcionar.

Os principais componentes do computador e as suas funções

Um computador tem quatro componentes principais: a unidade central de processamento (CPU), a memória, os dispositivos de entrada, e os dispositivos de saída. O CPU (*Central Processing Unit*) é a parte do computador que efetua os cálculos e controla as outras partes. A memória é o local onde o computador armazena os dados em que está a trabalhar. Os dispositivos de entrada são os dispositivos que o utilizador usa para introduzir dados no computador, tais como o teclado e o rato. Os dispositivos de saída são os dispositivos que o computador utiliza para exibir os resultados dos seus cálculos, como, por exemplo, o monitor e a impressora.

1.3 Como os computadores armazenam dados

O sistema binário – conceito

O sistema binário é uma forma de representar a informação usando dois símbolos: 0 (zero) e 1 (um). É utilizado em sistemas informáticos para armazenar e processar informação, sendo a forma mais simples de o conseguir.

Armazenamento de número e caracteres

ASCII

Os sistemas informáticos armazenam textos e números de várias maneiras, cada um com as suas próprias vantagens e desvantagens. A forma mais comum de armazenar

texto é através dos caracteres ASCII (*American Standard Code for Information Interchange*). No ASCII, cada caractere é representado por um número, de 0 (zero) a 127. Este número é chamado de código ASCII do caractere. Quando um computador armazena texto no modo ASCII, isto quer dizer que armazena os códigos ASCII para cada caractere no texto.

Unicode

Outra forma de armazenar texto é salvando-o como caracteres Unicode. Unicode é uma norma internacional que define um número único para cada carácter em cada língua. Quando um computador armazena texto como caracteres Unicode, armazena o código Unicode para cada caractere do texto.

UTF-8

UTF-8 é uma codificação de caracteres que consegue armazenar texto e números num único carácter Unicode. Esta codificação é amplamente utilizada na Internet porque pode codificar todos os caracteres numa variedade de línguas. O UTF-8 é uma codificação de comprimento variável, o que significa que pode codificar caracteres de diferentes tamanhos. A codificação mais pequena é de 1 *byte*, sendo a maior de 4 bytes. Esta codificação é retrocompatível com ASCII, o que significa que o texto ASCII será codificado em UTF-8 usando 1 byte.

Números

A forma mais comum de armazenar números é como números inteiros binários. Em binário, cada número é representado por uma sequência de 0s e 1s. Por exemplo, o número 12 pode ser representado como: 01001000 O número 12 também pode ser representado em hexadecimal, que é um sistema de numeração de base 16. Em hexadecimal, cada número é representado por uma cadeia de dígitos hexadecimais. Por exemplo, o número 12 pode ser representado como: C Quando um computador armazena um número em binário ou hexadecimal, armazena o valor inteiro do número.

Outros tipos de dados

Os computadores são frequentemente referidos como dispositivos digitais. O termo digital pode ser usado para descrever qualquer coisa que utilize números binários. Dados binários são dados que são armazenados em binário, e um dispositivo digital é qualquer dispositivo que funcione com dados binários. Nesta secção discutimos como os números e caracteres são armazenados em sistema binário, mas os computadores também funcionam com muitos outros tipos de dados digitais. Por exemplo, considerando as fotografias que tiramos com a máquina fotográfica digital, estas imagens são compostas por pequenos pontos de cor conhecidos como *pixels* (o termo *pixel* significa elemento de imagem). Cada *pixel* de uma imagem é convertido para um código numérico que representa a cor do pixel. O código numérico é armazenado na memória como um número binário. A música que toca no seu leitor de CD, iPod ou leitor de MP3 é também digital. Uma canção digital é composta por pequenas peças de música chamadas *samples*. Cada amostra é transformada num número binário, que pode ser armazenado na memória de um computador. Quanto mais amostras uma canção tiver, soará de forma mais parecida em relação à música original quando reproduzida. Uma canção com qualidade de CD tem mais de 44.000 *samples* por segundo!

1.4. Como funciona um programa

Existem muitos tipos diferentes de programas de computador, mas todos eles têm os mesmos componentes básicos: uma interface de utilizador, um processador e memória. A interface de utilizador permite que este introduza informações e instruções no programa, o processador executa as instruções, e a memória armazena o programa e os dados que este último processa. A maioria dos programas de computador são escritos numa linguagem de programação de alto nível, uma linguagem concebida para ser de fácil leitura e escrita pelo ser humano. Contudo, o processador só pode compreender o código da máquina, que é uma série de uns e zeros. Portanto, antes de um programa poder ser executado, deve ser convertido em código de máquina. Ou seja, terá de ser feito por um programa chamado *compilador*. O compilador lê o programa e converte-o em código de máquina. Depois armazena o código da máquina num ficheiro chamado *executável*. Quando o utilizador executa o programa, o executável é carregado na memória e o processador executa as instruções.

O ciclo *fetch-decode-execute*

O ciclo *fetch-decode-execute* é o processo básico que um computador utiliza para executar instruções. O ciclo começa quando o computador vai buscar uma instrução à memória. Em seguida, descodifica a instrução para determinar o que é suposto fazer. Finalmente, executa a instrução. O ciclo repete-se então, recuperando a instrução seguinte da memória.

Da linguagem de máquina à linguagem de montagem

Como a programação em linguagem de máquina, que consiste apenas em código binário, é demasiado complexa para um ser humano, foi criada uma linguagem de montagem. A linguagem de montagem (ou *assembly*) trata-se de uma linguagem de programação de baixo nível para um computador, microprocessador, ou outro dispositivo programável, na qual o programador utiliza as instruções da linguagem

assembly para controlar o funcionamento do dispositivo. A linguagem de montagem é específica para um determinado microprocessador ou família de microprocessadores. Consiste numa série de códigos mnemónicos, nomes simbólicos para as operações que o microprocessador pode realizar, e os operandos (dados) sobre os quais estas operações devem ser realizadas. A linguagem de montagem é convertida em código de máquina, uma forma de código binário específico de um determinado tipo de computador e que pode ser compreendido pelo processador do computador. O código da máquina é a única forma de código que o processador pode executar diretamente.

Mesmo sendo mais fácil do que a programação em linguagem de máquina, a programação em linguagem de montagem não é suficientemente útil para produzir código fonte rápido e fácil de ler. Por conseguinte, as linguagens de programação de alto nível (tais como *C#* ou *python*) foram criadas.

Atualmente, existem muitas linguagens de programação de alto nível. Algumas das mais comuns são Java, Python e Ruby. As linguagens de programação de alto nível são mais fáceis de utilizar do que as linguagens de programação de baixo nível. Permitem que se concentre na tarefa em mãos, em vez de se concentrar nos detalhes do computador. Isto torna-as ideais para a criação de aplicações e programas. As linguagens de programação de alto nível também tendem a ser mais tolerantes do que as linguagens de programação de baixo nível. Se cometer um erro ao escrever código numa linguagem de alto nível, o compilador será normalmente capaz de o corrigir por si. Isto pode poupar-lhe muito tempo e frustração ao "*bater código*".

Palavras-chave, Operadores, e Sintaxe: uma visão geral

Há muitas linguagens de programação de alto nível disponíveis hoje em dia. Cada uma tem o seu conjunto único de palavras-chave, operadores e sintaxe. Para ser eficaz com uma linguagem de programação de alto nível, é importante estar familiarizado com as palavras-chave, operadores, e sintaxe específicos utilizados por essa linguagem. Algumas das palavras-chave mais comuns utilizadas nas linguagens de programação

de alto nível incluem: *if, then, else, while, for, do, break, continue*. Estas palavras-chave são utilizadas para controlar o fluxo da execução do programa. Os operadores são símbolos que representam operações que podem ser realizadas sobre valores. Os operadores mais comuns incluem: + *(addition), - (subtraction), * (multiplication), / (division), e % (modulus)*. Estes operadores podem ser utilizados para calcular os resultados das expressões. A sintaxe de uma linguagem de programação é o conjunto de regras que dirigem a forma como o código deve ser escrito para ser interpretado pelo compilador ou pelo intérprete. A sintaxe de uma linguagem de programação de alto nível é tipicamente mais tolerante do que a sintaxe de uma linguagem de nível inferior. Isto pode facilitar aos principiantes a aprendizagem da programação.

Compiladores e Intérpretes

Os compiladores e intérpretes de computador são ferramentas importantes para os programadores de *software*. Um compilador pega no código escrito numa língua e converte-o em código que pode ser executado numa máquina diferente. Um intérprete pega no código escrito numa língua e executa-o tal como está, sem o compilar primeiro. Os compiladores são tipicamente utilizados para linguagens que têm muita estrutura, como *C* ou *Java*. Os intérpretes são tipicamente utilizados para linguagens que são mais flexíveis, como Python ou Ruby. Os compiladores produzem normalmente códigos mais rápidos do que os intérpretes. No entanto, os intérpretes são tipicamente mais *portáteis*, o que significa que podem funcionar em mais tipos de máquinas. O instrumento a utilizar depende da situação. Se a velocidade é importante, um compilador é uma melhor escolha. Se a portabilidade é importante, um intérprete é uma melhor alternativa.

1.5. Programas e linguagens de programação

As linguagens de programação são usadas para criar programas, que são usados para controlar o comportamento de uma máquina, tipicamente um computador. Uma linguagem de programação fornece uma estrutura para o programador dar instruções à máquina, e uma forma de comunicar essas instruções a outros programadores. Existem muitas linguagens de programação em uso atualmente. As mais populares são C, Java, Python, e JavaScript.

Tipos de linguagens

Existem dezenas de linguagens de programação em uso hoje em dia, podendo ser sumariamente classificadas em cinco categorias:

- **Linguagens de programação de baixo nível:** Estas linguagens de programação são muito próximas do *hardware* e são utilizadas para programar microprocessadores e outros dispositivos de baixo nível. Não são fáceis de aprender e não são populares para a programação de uso geral. Exemplos: Linguagem de montagem, linguagem de programação C, e linguagem de montagem de baixo nível.
- **Linguagens de programação de alto nível:** Estas linguagens de programação foram concebidas para serem fáceis de aprender e utilizar. São populares para a programação de uso geral. Exemplos: Java, C++ e Python.
- **Linguagens de *scripting*:** são concebidos para serem fáceis de usar e são populares para de *scripting*. Exemplos: Python, Ruby e JavaScript.
- **Línguas específicas do domínio:** são concebidas para uma tarefa ou indústria específica. Não são fáceis de aprender e não são populares para programação de uso geral. Exemplos: MATLAB, SQL, and FORTRAN.
- **Linguagens de programação orientadas a objetos:** são baseadas no paradigma da programação orientada para objectos. Exemplos: Java, C++, and Python.

Do programa de alto nível ao ficheiro executável

Quando um programa de computador é escrito numa língua de alto nível, é primeiramente traduzido para uma língua de nível inferior, que é mais facilmente compreendida por máquinas. A língua de nível inferior é então compilada num ficheiro executável, que pode ser executado num computador.

IDEs (*Integrated Development Environments*)

Um Ambiente Integrado de Desenvolvimento (IDE) é uma aplicação de software que proporciona instalações abrangentes aos programadores de computadores para o desenvolvimento de *software*. Um IDE consiste tipicamente num editor de código fonte, ferramentas de automação de construção e um depurador. O editor de código fonte permite ao programador escrever o código, enquanto as ferramentas de automatização da construção automatizam o processo de compilação desse código numa forma que o computador pode executar. O depurador permite ao programador percorrer o código, examinando o estado do programa em cada ponto da sua execução. As IDEs são muitas vezes utilizadas em conjunto com um sistema de controlo de versões, o que permite a diferentes programadores que trabalham no mesmo projeto partilhar e fundir as suas alterações sem problemas.

Os elementos mais comuns em linguagens de programação

As linguagens de programação informática partilham uma série de elementos comuns, apesar das suas diferenças. Todas as linguagens de programação têm uma forma de representar instruções para o computador de uma forma que o computador possa compreender. Isto é normalmente chamado código, ou código fonte. Os programadores utilizam o código para criar programas e aplicações de *software*. Todas as linguagens de programação têm também uma forma de organizar instruções para que possam ser reutilizadas, modificadas ou partilhadas com outros programadores. A isto chamamos normalmente “biblioteca” ou “módulo”. As bibliotecas e os módulos permitem aos

programadores criar programas complexos, baseando-se no trabalho de outros programadores. Por fim, todas as linguagens de programação têm uma forma de transmitir informação ao utilizador sobre o que o programa está a fazer e como está a funcionar. Isto é normalmente chamado de informação de saída ou *debug*. A informação de saída e *debug* ajuda os programadores a compreender e corrigir problemas dos seus programas.

Programação processual e orientada a objetos

Há dois tipos principais de programação: processual e orientada para objetos. A programação processual envolve um processo passo-a-passo, enquanto a programação orientada aos objetos envolve a criação de objetos que interagem uns com os outros. A programação processual é muitas vezes vista como mais simples do que a programação orientada a objetos. É fácil aprender os passos necessários para completar uma tarefa, e é fácil alterar a ordem desses passos sem afetar o resultado. Contudo, a programação de procedimentos pode ser menos eficiente porque pode ser difícil reutilizar o código que foi escrito para uma tarefa específica. A programação orientada a objetos é mais complexa que a programação processual, mas permite maior flexibilidade e reutilização do código. Os objetos podem ser criados para tarefas específicas e depois reutilizados conforme necessário. Além disso, o código orientado para objetos é muitas vezes mais fácil de ler e compreender do que o código processual. Contudo, a programação orientada aos objetos pode ser mais difícil de aprender e pode ser menos eficiente do que a programação processual.