Conceitos de tipos de dados

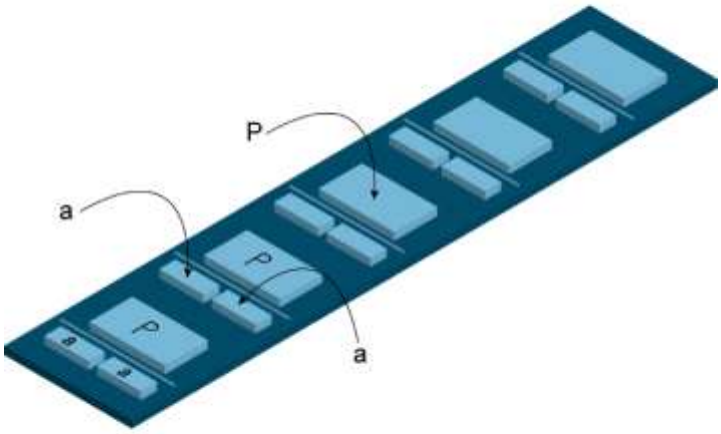
Dados

Tipos de dados

Você sabe como podemos representar a solução de um problema da vida real na linguagem de programação? É possível fazer isso por meio de uma sequência finita de passos conhecida como algoritmo.

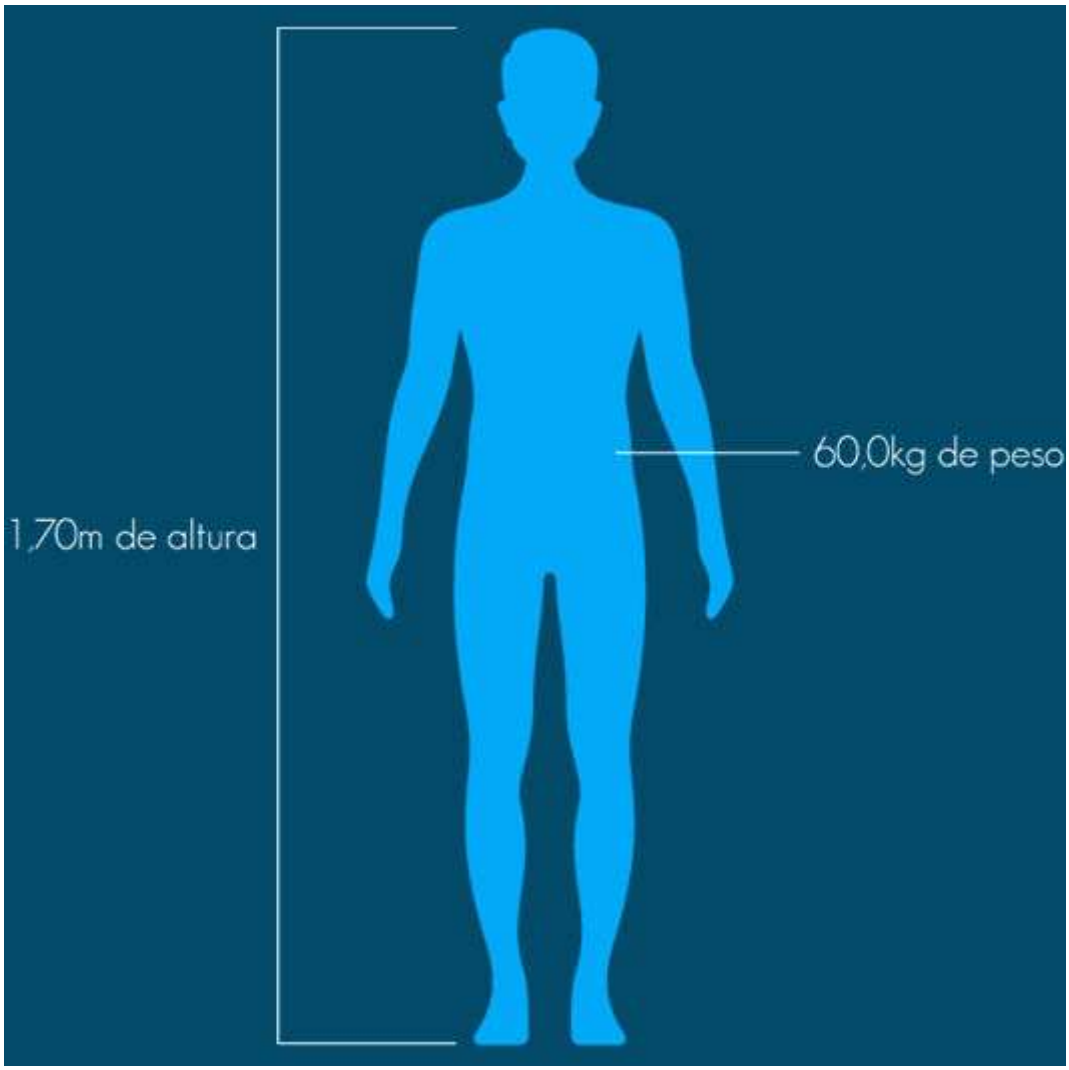
Um algoritmo pode ser entendido como uma linha de produção fabril semelhante à do Fordismo, uma vez que transformamos os dados de entrada para alcançarmos ou calcularmos determinado valor.

Veja na imagem ao lado a representação do carregamento do código na linguagem C como uma linha de produção. Os espaços de memória recebem inputs (entradas) para transformá-los em códigos (saídas). Com a execução do código, esses espaços são preenchidos por variáveis que dão origem à linguagem de programação.



Representação do carregamento do código.

Para o cálculo do índice de massa corpórea (IMC) de uma pessoa, medimos sua altura e seu peso. Normalmente, ela é medida em metros e possui valores com casas decimais. Da mesma forma, ele o é em quilogramas, apresentando casas decimais. Suponhamos que essas medidas apresentem os seguintes números apresentados na imagem a seguir.



A representação em casas decimais de números tão comuns do nosso dia a dia também pode ser feita nas linguagens de programação.

Para desenvolver um algoritmo, precisamos:

1. Identificar quais dados de entrada serão utilizados e como representá-los em nossa linguagem de programação.
2. Fazer com esses dados já identificados as transformações necessárias para modificar ou realizar cálculos.

Da mesma forma que, para montar um carro, Ford iniciava o processo com uma carroceria a fim de poder agregar seus demais componentes, como portas, sistema de suspensão e motor, nós o começamos com uma região de memória na qual serão acrescentados os dados necessários para a realização do nosso cálculo.

Desde sua concepção, a linguagem C possui quatro tipos de dados básicos:

1. *char*;
2. *int*;
3. *float*;
4. *double*.

Por meio deles e de suas manipulações, é possível representar qualquer tipo de informação do mundo real.

Dica

Ainda existe nessa linguagem uma forma de identificar a ausência de valores. Tal situação será vivida quando posteriormente forem tratados os casos de modularização de códigos (funções e procedimentos). Neste caso, é usada a palavra reservada *void*.

Char

O tipo *char* representa um caractere (podendo ser uma letra, um número ou um símbolo) e ocupa um *byte* na memória. Em computação, ele é representado pela tabela ASCII (Sigla para American Standard Code for Information Interchange) com seus 256 símbolos. Observe-a a seguir:

ASCII control characters			ASCII printable characters						Extended ASCII characters							
00	NULL	(Null character)	32	space	64	@	96	`	128	Ç	160	á	192	└	224	Ó
01	SOH	(Start of Header)	33	!	65	A	97	a	129	ü	161	í	193	┴	225	ß
02	STX	(Start of Text)	34	"	66	B	98	b	130	é	162	ó	194	┬	226	Ô
03	ETX	(End of Text)	35	#	67	C	99	c	131	â	163	ú	195	├	227	Ò
04	EOT	(End of Trans.)	36	$	68	D	100	d	132	ä	164	ñ	196	─	228	õ
05	ENQ	(Enquiry)	37	%	69	E	101	e	133	à	165	Ñ	197	┼	229	Õ
06	ACK	(Acknowledgement)	38	&	70	F	102	f	134	å	166	ª	198	ã	230	µ
07	BEL	(Bell)	39	'	71	G	103	g	135	ç	167	º	199	Ã	231	þ
08	BS	(Backspace)	40	(	72	H	104	h	136	ê	168	¿	200	╚	232	Þ
09	HT	(Horizontal Tab)	41	)	73	I	105	i	137	ë	169	®	201	╔	233	Ú
10	LF	(Line feed)	42	*	74	J	106	j	138	è	170	¬	202	╩	234	Û
11	VT	(Vertical Tab)	43	+	75	K	107	k	139	ï	171	½	203	╦	235	Ù
12	FF	(Form feed)	44	,	76	L	108	l	140	î	172	¼	204	╠	236	ý
13	CR	(Carriage return)	45	-	77	M	109	m	141	ì	173	¡	205	═	237	Ý
14	SO	(Shift Out)	46	.	78	N	110	n	142	Ä	174	«	206	╬	238	¯
15	SI	(Shift In)	47	/	79	O	111	o	143	Å	175	»	207	¤	239	´
16	DLE	(Data link escape)	48	0	80	P	112	p	144	É	176	░	208	ð	240	≡
17	DC1	(Device control 1)	49	1	81	Q	113	q	145	æ	177	▒	209	Ð	241	±
18	DC2	(Device control 2)	50	2	82	R	114	r	146	Æ	178	▓	210	Ê	242	‗
19	DC3	(Device control 3)	51	3	83	S	115	s	147	ô	179	│	211	Ë	243	¾
20	DC4	(Device control 4)	52	4	84	T	116	t	148	ö	180	┤	212	È	244	¶
21	NAK	(Negative acknowl.)	53	5	85	U	117	u	149	ò	181	Á	213	ı	245	§
22	SYN	(Synchronous idle)	54	6	86	V	118	v	150	û	182	Â	214	Í	246	÷
23	ETB	(End of trans. block)	55	7	87	W	119	w	151	ù	183	À	215	Î	247	¸
24	CAN	(Cancel)	56	8	88	X	120	x	152	ÿ	184	©	216	Ï	248	°
25	EM	(End of medium)	57	9	89	Y	121	y	153	Ö	185	╣	217	┘	249	¨
26	SUB	(Substitute)	58	:	90	Z	122	z	154	Ü	186	║	218	┌	250	·
27	ESC	(Escape)	59	;	91	[	123	{	155	ø	187	╗	219	█	251	¹
28	FS	(File separator)	60	<	92	\	124	|	156	£	188	╝	220	▄	252	³
29	GS	(Group separator)	61	=	93	]	125	}	157	Ø	189	¢	221	¦	253	²
30	RS	(Record separator)	62	>	94	^	126	~	158	×	190	¥	222	Ì	254	■
31	US	(Unit separator)	63	?	95	_			159	ƒ	191	┐	223	▀	255	nbsp
127	DEL	(Delete)														

ASCII control characters / ASCII printable characters / Extended ASCII characters.

Nos 256 símbolos listados, ocorre a seguinte divisão:

Do 0 ao 31

Do 32 ao 127

Do 128 ao 255

Os 32 iniciais são símbolos de controle.

Esses caracteres podem ser usados de diversas formas.

Exemplo

A representação dos termos masculino e feminino em um cadastro é feita pelos caracteres *M* e *F*. Utilizam-se aspas simples para a sua representação quando ambos forem mostrados em uma implementação. Desse modo, o caractere **M** de **masculino** é representado por '*M*' e o **F** de **feminino**, por '*F*'.

Observemos que a tabela ASCII representa os caracteres minúsculos e maiúsculos de forma distinta:

m ≠ M

Assim, conforme pode ser visto, o ‘*m*’ (**m** minúsculo) é diferente de ‘*M*’ (**M** maiúsculo).

Notemos também que, para cada caractere da tabela, existe um índice representado em decimal ou hexadecimal.

Manipulação de variáveis e constantes

Já sabemos como representar os dados do mundo real na linguagem de programação C. Agora precisamos entender como eles podem ser manipulados. Para fazer isso, a linguagem trata os dados como variáveis e constantes.

Conceito

A variável é um tipo de espaço de memória que pode ser alterado a qualquer tempo. A constante, por sua vez, não pode. As duas formas permitem a referenciação deles em um espaço de memória. Esses espaços são identificados por meio de rótulos. Chamados de identificadores, eles possibilitam, a partir de seu uso, o acesso ao conteúdo armazenado em memória.

Exemplo

Caixas de correio que ficam em frente às residências.

Definição das variáveis

Formalmente, um espaço de memória é rotulado por intermédio de um identificador quando as variáveis são definidas. Veja uma ilustração de processo na próxima imagem.

Para criar uma variável, utiliza-se a seguinte notação:

O tipo do dado pode ser qualquer um dos quatro tipos já abordados: *char*, *int*, *float* e *double*.

De acordo com o que vimos, como estaria descrita a representação dos valores de peso e altura? Veja no próximo recurso.

LINGUAGEM C

```
float peso;

float altura;
```

Já sabemos que a linguagem C é considerada sensível a um contexto. Assim, ao escrevermos uma aplicação nessa linguagem, os identificadores serão diferentes. Veja um exemplo a seguir:

peso - Peso - PESO

Além disso, o próprio tipo de dado utilizado possui a mesma regra. Desse modo:

Float (todas as letras são minúsculas) = ao tipo de dado, constituindo uma palavra reservada da linguagem.

Quaisquer representações diferentes não correspondem a ele, podendo, dessa forma, ser utilizadas como identificadores, a exemplo de *Float* ou *FLOAT*.

Recomendação

Não constitui uma boa prática de programação usar identificadores que sejam variantes em minúsculas ou maiúsculas de palavras reservadas. Por exemplo, não é recomendável o uso de identificadores como *Float* ou *FLOAT*.

Ainda podemos definir as variáveis com outro formato:

LINGUAGEM C

```
TIPO NOME_DO_IDENTIFICADOR_1,
NOME_DO_IDENTIFICADOR_2;
```

Como estaria descrita, portanto, a representação dos valores de peso e altura? Vamos conferir!

LINGUAGEM C

```
float peso, altura;

//OU

float altura, peso;
```

Também é possível estabelecer uma quantidade maior de variáveis separando-as sempre das demais pelo uso de vírgula, enquanto a última deve conter um ponto e vírgula para finalizar.

Atenção!

Não é recomendável definir uma quantidade muito grande de variáveis de uma só vez, pois isso dificulta o entendimento do código-fonte da aplicação.

Recomendamos a definição de poucas variáveis por vez. Caso haja algum tipo de relação entre elas, essa identificação deve ser feita por meio de comentários.

Seguindo o exemplo do caso de peso e altura, faríamos assim:

LINGUAGEM C

```
float altura, peso;
//Valores de altura e peso do usuário,
//medidos em metros e quilograma,
//respectivamente.
```

Outro ponto importante é que uma variável sempre deve ser definida antes de seu uso. Assim, quando formos usar determinada variável, sua definição deverá ocorrer previamente.

Coloquialmente conhecidos como **nomes de variáveis**, os identificadores podem ter até 32 caracteres formados por:

1. Letras do alfabeto (maiúsculas e minúsculas);
2. Dígitos (0-9);
3. Símbolo de underscore _ .

O primeiro caractere deve ser uma letra do alfabeto ou o underscore.

Não usamos caracteres acentuados ao definirmos um identificador.

Saiba mais

Pesquise na internet sobre a notação húngara criada por Charles Simonyi.

Além das variáveis, há situações em que é necessário usar valores fixos em toda a aplicação. Conhecidos como constantes, esses valores são definidos por intermédio da palavra reservada “const” antes do tipo de acordo com o seguinte formato:

const TIPO NOME_DO_IDENTIFICADOR;

Nele, o NOME_DO_IDENTIFICADOR segue as mesmas regras relativas ao identificador descritas anteriormente. Por exemplo, é possível definir o valor π usando o seguinte exemplo:

const float pi = 3.141592;

Conceitos

Aplicação dos conceitos apresentados

Da teoria da informação, surgem os conceitos de:

Dados

Considerado um valor sem contextualização.

Informação

Quando é contextualizado, o dado transforma-se em informação.

Hoje em dia, a informação é o principal fator de destaque em empresas vencedoras. A partir dos dados contextualizados, é possível compreender tudo à nossa volta. Afinal, eles dão origem a áreas que estão revolucionando o mercado nos últimos anos.

Exemplo

Big data, ciência de dados e inteligência artificial.

Só será possível analisar os dados, entendendo suas correlações e regras de formação, se eles forem tratados da melhor forma à medida que estiverem sendo capturados no mundo real.