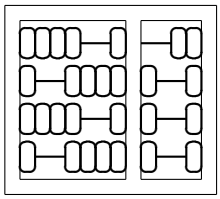

MC-102 Algoritmos e Programação de Computadores
IC-UNICAMP

Aula 03 - Variáveis

Por: Luís Augusto Angelotti Meira
(Sala IC-71) 1S2005

1 Objetivos

Conceituar constantes e variáveis, definir variáveis em C, atribuição, tipos de variáveis, `int, float, double, long, char,` inteiros com e sem sinal, inicializar variáveis, nome de variáveis, palavras chaves.

2 Motivação

Variáveis são importantes para facilitar a programação. Comandos repetitivos ficam mais sucintos com variáveis. Atribuição é uma ferramenta fundamental da programação.

3 Aula e Exemplos

A aula que se segue baseou-se em [1, 3, 2].

3.1 Constantes

Exemplos de constantes:

- `'c'`
- 8
- `"Primeiro Programa"`

Programas:

```
#include <stdio.h>

int main(void){
  printf("O programa A imprime o número 2\n");
  printf("O programa %c imprime o número 2\n",'A');
  printf("O programa A imprime o número %d\n",2);
  printf("O programa %c imprime o número %d\n",'A',2);
  printf("O programa %c %s %d\n",'A',"imprime o número",2);

  return 1;

}
```

A saída deste programa será

```
O programa A imprime o número 2
O programa A imprime o número 2
O programa A imprime o número 2
O programa A imprime o número 2
O programa A imprime o número 2
```

3.2 Variáveis

Usar o exemplo da caixinha de sapato e fósforo. Um variável inteira é uma caixinha de fósforo. O número que ela representa é um valor que fica guardado dentro da caixinha, como, por exemplo, número de fósforos nela. Existe um limite para o número que ela representa. Uma variável `float` é uma caixa de sapato e o valor que ela representa é, por exemplo, o comprimento do sapato. Não se pode misturar o conteúdo entre tipos diferentes.

Declarando uma variável em C:

```
int num;
```

Atribuindo um valor a uma variável:

```
num = 3;
```

Programa que usa variável:

```
#include <stdio.h>

int main(void){
  int num;
  num = 10;
  printf("O programa A imprime o número %d\n",num);
  return 1;

}
```

A saída deste programa será:

```
O programa A imprime o número 10
```

3.3 Tipos de Variáveis

Em C temos os seguintes tipos *básicos* de variáveis:

- `char`: Guarda um caracter;
- `int`: Guarda um número inteiro;
- `float`: Guarda um número real com certa precisão;
- `double`: Guarda um número real com precisão maior que float;
- `void`: Tipo vazio.

Temos algumas variações destes tipos, que são:

- `unsigned char`: caracter sem sinal;
- `long int`: número inteiro, com domínio estendido;
- `unsigned int`: numero inteiro positivo;
- `unsigned long int`: numero inteiro positivo com domínio estendido.
- `short int`: inteiro com domínio reduzido.
- `unsigned short int`: inteiro positivo com domínio reduzido.

Características dos tipos. Os tipos variam com a arquitetura. O tipo `int` tem tamanho igual ao do barramento do processador. Antigamente este valor era de 16b e hoje é de 32b. Os tamanhos dos tipos abaixo foram extraídos de um pentium IV.

Tipo	Num de bits	Formato i/o	Início	Fim
char	8	%c	-128	127
unsigned char	8	%c	0	255
int	32	%d	-2.147.483.648	2.147.483.647
unsigned int	32	%u	0	4.294.967.295
long int	32	%li	-2.147.483.648	2.147.483.647
unsigned long int	32	%lu	0	4.294.967.295
short int	16	%hi	-32.768	32.767
unsigned short int	16	%hu	0	65.535
float	32	%f	$(+/-)10^{-38}$	$(+/-)10^{38}$
double	64	%lf	$(+/-)10^{-308}$	$(+/-)10^{308}$
long double	96			

	bits	mantissa	exponent	sign
character	8	7	0	1
long integer	32	31	0	1
float	32	23	8	1
double	64	52	11	1
long double	96			

O programa

```
#include <stdio.h>

int main(void){
      float a = 1000.43;
      float b = 1000.0;
      printf("%f\n",a - b);
  return 1;
}
```

Em uma implementação qualquer de C, o código acima imprime 0.429993.

Arrredondamentos, truncamentos e aproximações não são realmente um problema do C; são um problema da ciência da computação.

Ponto flutuante é uma aproximação. O padrão IEEE para 32 bits suporta um bit para sinal, 8 bits para o expoente e 23 bits para a mantissa. Devido ao fato de que uma mantissa, representada em sistema binário, tem a forma 1.xxxxx... o primeiro dígito um é suprimido e você tem efetivamente 24 bits para mantissa. O número 1000.43 não é representado exatamente em ponto flutuante ou formato duplo. 1000.43 é, na realidade, representado pelo seguinte padrão de bits

O s indica a posição do bit de sinal, e's indicam as posições do expoente, e os m's indicam as posições da mantissa):

```
seeeeeeeemmmmmmmmmmmmmmmmmmmmmmm
00000100111110100001101110000101
++ 9     111.1010.0001.1011.1000.0101
```

```
2^9*(1),111.1010.0001.1011.1000.0101

    1.111.101.000,01101110000101
```

A número é 1111101000.01101110000101 ou 1000.429992675781. Com 24 bits de mantissa tem-se uma precisão de 1 parte em 16M para ponto flutuante. O tipo double propicia uma maior precisão porque tem 53 bits de mantissa.

3.4 Inicializando uma Variável

Uma variável pode ser inicializada através de uma atribuição, como no programa abaixo:

```
#include <stdio.h>

int main(void){
  int evento ;
  char corrida;
  float tempo;
  evento = 5;
  corrida = 'C';
  tempo = 27.25;
  printf("O tempo vitorioso na eliminatória %c",corrida);
  printf("\nda competição %d foi %f.", evento, tempo);
  return 1;
}
```

Ou diretamente em sua declaração:

```
#include <stdio.h>

int main(void){
  int evento = 5 ;
  char corrida = 'C';
  float tempo = 27.25;
  printf("O tempo vitorioso na eliminatória %c",corrida);
  printf("\nda competição %d foi %f.", evento, tempo);
  return 1;
}
```

Nos dois casos, a saída do programa será:

```
O tempo vitorioso na eliminatória C
da competição 5 foi 27.250000.
```

3.5 Nome das Variáveis

- O nome das variáveis pode ser **qualque palavra que não seja uma palavra chave da linguagem.**
- É possivel conter um número na palavra:`Casa1`
- Não é aceitável iniciar com um número: `1casa` (errado)
- É possivel utilizar subscrito: Casa_da_ana
- Não pode-se utilizar:

```
{ ( + - * / ; . , ?
```

As seguintes palavras já tem um significado na linguagem C e por esse motivo não podem ser utilizadas como nome de variáveis:

```
auto      double  int       struct
break     enum    register  typedef
char      extern  return    union
const     float   short     unsigned
continue  for     signed    void
default   goto    sizeof    volatile
do        if      static    while
```

4 Exercícios

1) Corrija o seguinte programa:

```
#include <stdio.h>
int main{void}(
   printf(Existem %d semanas no ano., 56)
   return1;

)
```

2) Execute o seguinte programa e veja as mensagem de erro geradas pelo compilador e corrija-o:

```
#include <stdio.h>
int Main(void){
  int a=1; b=2; c=3;
  printf("Os números são: %d, %d e %d\n,a,b,c,d)
}
```

3) Qual será a saída do programa abaixo?

```
#include <stdio.h>
int main(void){
  printf("%s\n%s\n%s","um","dois","três");
}
```

4) O que é uma variável em C?

5) O que é uma constante em C? De exemplos.

6) Quais nomes de variáveis são aceitas pelo compilador C?

- `3ab`
- `ab3`
- `a3b`
- `FIM`
- `_sim`
- `int`
- `\meu`
- `____A`
- `n_a_o`
- `A123`
- `papel-branco`
- `a*`
- `c++`
- `*nova_variavel`

7) Quais das seguintes instrucções são corretas?

- `int a;`
- `float b;`
- `double float c;`
- `unsigned char d;`
- `long float e;`

Referências

[1] Henrique José dos Santos. Curso de linguagem c, ufmg. Universidade Federal de Minas Gerais.

[2] Flávio Keidi Miyazawa. Notas de aula de algoritmos e programação de computadores. Colaboradores : Cid Carvalho de Souza e Tomasz Kowaltowski.

[3] Victorine, Viviane, and Mizrahi. Treinamento em linguagem c, curso completo, módulo 1.