Brevíssima introdução à linguagem de programação C *

Lenimar Nunes de Andrade
UFPB - João Pessoa, PB
e-mail: lenimar@mat.ufpb.br

22 de setembro de 1999

1 Introdução

A linguagem de programação C foi criada no início dos anos 70 e é o resultado do desenvolvimento de uma linguagem mais antiga chamada BCPL. A BCPL deu origem a uma linguagem chamada B, que por sua vez deu origem à linguagem C. Posteriormente, a partir dos anos 80, extensões de C têm sido cada vez mais divulgadas como por exemplo as linguagens C++ e Java.

2 Comentários

Os comentários em C são delimitados por /* e */. Extensões de C também consideram como comentários tudo o que estiver à direita de // . Os comentários servem apenas para orientação do usuário e são ignorados pelo compilador.

```
/* Isto e' um exemplo  de um comentario em C */

//  ... e isto e' um comentario em C++ !
```

3 Palavras reservadas

São as palavras que o compilador reserva para fins específicos. O usuário não pode redefini-las. Por exemplo, não é possível a criação de uma variável de nome `long` em um programa pois essa palavra já é usada com outra finalidade.

Algumas das palavras reservadas de C ou de C++ são:

auto	break	case	char	class	const	continue	default	delete
double	else	enum	extern	far	float	for	friend	goto
if	inline	int	interrupt	long	operator	pascal	private	public
return	short	signed	static	struct	switch	template	this	typedef
unsigned	virtual	void	volatile	while	do	huge	register	union

4 Tipos de dados

Antes de ser usada, qualquer variável, constante ou função deve ser previamente declarada. Geralmente, isto pode ser feito usando-se um dos tipos de dados pré-definidos mostrados a seguir:

*Disponível em `ftp://mat.ufpb.br/pub/docs/cursos/brevissm.zip`

Tipo	Bytes	Domínio
unsigned char	1	0 a 255
char	1	-128 a 127
enum	2	-32.768 a 32.767
unsigned int	2	0 a 65.535
short int	2	-32.768 a 32.767
int	2	-32.768 a 32.767
unsigned long	4	0 a 4.294.967.295
long	4	-2.147.483.648 a 2.147.483.647
float	4	$3,4 \cdot 10^{-38}$ a $3,4 \cdot 10^{+38}$
double	8	$1,7 \cdot 10^{-308}$ a $1,7 \cdot 10^{+308}$
long double	10	$3,4 \cdot 10^{-4932}$ a $1,1 \cdot 10^{+4932}$

Os 8 primeiros tipos acima podem ser considerados inteiros, enquanto que os 3 últimos são tipos reais (admitem representação com ponto flutuante).

O tipo string de outras linguagens deve ser considerado em C como sendo uma seqüência de caracteres.

As constantes do tipo char são escritas entre apóstrofos e as do tipo string são escritas entre aspas:

```
char x = 'A', y = 'B';          /* caracteres              */
char s[] = "UFPB - CCEN - DM";  /* variavel do tipo string */
```

Todo comando em C deve encerrar com um ponto e vírgula (;).

Um tipo enumerado definido pelo usuário pode ser definido em C usando-se a palavra `enum`, por exemplo:

```
enum vogal = {'a', 'e', 'i', 'o', 'u'};
```

O C não tem um tipo booleano (como Pascal). No entanto, ele pode ser considerado como sendo o tipo inteiro, no qual o 0 equivale ao falso e qualquer valor diferente de 0 equivale ao verdadeiro.

Com o comando `typedef` é possível a definição de novos tipos:

```
typedef booleano int;
typedef real    float;
```

5 Constantes

As declarações de constantes em C podem ser feitas com a palavra reservada `const` escritas antes do tipo de dado:

```
const float e = 2.718;   /* Constante real    */
const int ano = 1999;    /* Constante inteira */
```

6 Variáveis

As variáveis em C devem ser declaradas antes de serem usadas. Para isso, basta colocar o nome do tipo antes do nome da variável:

```
float x, y, z;      /* Variaveis reais */
int i, j;           /* Variaveis inteiras */
char  X, Y;         /* Variaveis do tipo ``caracter'' */
```

O C é "sensível ao tipo das letras", o que quer dizer que é feita distinção entre letras minúsculas e maiúsculas. Por exemplo, as variáveis aux, Aux e AUX serão consideradas distintas; as funções SIN(x) ou Sin(x) não serão reconhecidas pelo compilador como sendo a função seno sin(x).

7 Operadores

C tem todos os operadores comuns à maioria das linguagens de programação, como os operadores de adição (+), subtração (-), multiplicação (*), divisão (/), maior (>), maior ou igual (>=), menor (<) e menor ou igual (<=). Cada operador tem seu nível de prioridade com relação aos demais. Por exemplo, o operador de multiplicação tem prioridade sobre o de adição.

Operador	Símbolo	Exemplo
atribuição	=	a = b
igualdade	==	a == b
desigualdade	!=	a != b
módulo (congruência)	%	a % b
AND lógico	&&	a && b
OR lógico	||	a || b
NOT lógico	!	!a
incremento	++	a++ ou ++a
decremento	−−	a−− ou −−a
endereço	&	&a
referência a endereço	*	*a

O C também admite declarações como "a OP= b" como uma forma abreviada da atribuição "a = a OP b" onde OP é um operador binário. Por exemplo, x += 1 é considerado como sendo o mesmo que x = x + 1.

O operador de atribuição em outras linguagens costuma ser representado por um ":=" ou por um "←".

8 Comandos compostos

Um comando composto ou bloco de comandos em C é um conjunto de comandos delimitado por "{" e "}":

```
{
  comando1;
  ...
  comandoN;
}
```

Onde o C admitir um comando simples, também admitirá um comando composto.

Em outras linguagens, os delimitadores de comandos compostos podem ser denotados com as palavras "begin" e "end".

9 Arquivos de inclusão

Arquivos-texto podem ser incluídos em programas em C através de um `#include` seguido do nome do arquivo entre < e > ou entre aspas. É muito comum a inclusão de arquivos-cabeçalho (de extensão .H) que contêm constantes e cabeçalhos de funções.

```
#include  "teste.inc"
#include  <stdio.h>
#include  <math.h>
```

10 Comandos de entrada e saída

O C possui muitos comandos para entrada e saída de dados. Talvez os mais comuns sejam o `scanf` e o `printf` (semelhantes ao READ e WRITE de outras linguagens).

A sintaxe do `scanf` é `scanf(string, &var1, &var2, ...)` onde string é uma string de formatação e `&var1, &var2, ...` são os endereços das variáveis que vão ser lidas. No string de formatação, é especificado o tipo de cada variável:

%d	decimal	%x	hexadecimal
%u	inteiro sem sinal	%f	real
%c	caracter simples	%s	string

Por exemplo, para se ler um inteiro positivo x e um real y, deve-se usar um `scanf("%u %f", &x, &y);` .

A saída com o `printf` tem uma sintaxe semelhante:
`printf(string, var1, var2, ...)`, onde string é a string de formatação que inclui a definição dos tipos, mensagens e caracteres de controle como "\n", usado para indicar uma nova linha.

```
printf("Valor de x = %f", x)
printf("%d * %d", a, b);
printf("%d * %d\n", a, b);
```

Para ler apenas um caracter, o C dispõe de outras funções como `getch()` ou `getche()`:

```
ch = getch();    /* Espera ser pressionada uma tecla */
if (ch == 0) {
  ch = getch();  /* Espera outra tecla */
  printf("Codigo estendido = %d", ch);
}
```

11 Estruturas de decisão

O comando `if` permite que determinado comando seja executado ou não, dependendo de determinada condição ser ou não satisfeita.

```
if (condicao)
  comando1;
else
  comando2;
```

Se a condição for verdadeira, será executado o `comando1`; senão, será executado o `comando2`. Somente o `comando1` é obrigatório em um `if`.

Observe que o `if` não tem um "then" utilizado por outras linguagens.

Exemplo 11.1 *No fragmento a seguir, se x for positivo, y será definido como sendo a raiz quadrada de x; caso contrário, y será definido como sendo $x^2 + x - 1$.*

```
if (x > 0)
   y = sqrt(x);
else
   y = x*x + x - 1;
```

O comando `switch` seleciona uma entre várias opções, baseando-se no valor de uma expressão fornecida como parâmetro:

```
switch (expr) {
  case val1: comando1; break;
  case val2: comando2; break;
  case val3: comando3; break;
     ...
  default:   comandoN;
}
```

Se o valor de `expr` for `val1`, então será executado o `comando1`, se for `val2`, será executado o `comando2` e assim por diante.

Exemplo 11.2 *Neste exemplo, se $x = 1, 2$ ou 3 então y será respectivamente igual a $\sin(x)$, $\cos(x)$ ou $\log(x)$; senão será executado o comando* `encerrar()`.

```
if (opcao == 1)
  switch (x) {
    case 1  : y = sin(x); break;
    case 2  : y = cos(x); break;
    case 3  : y = log(x); break;
    default : encerrar();
  }
```

O comando `switch` *será executado somente se* `opcao` *for igual a 1.*

12 Estruturas de repetição

O C possui três comandos que podem ser usados para repetir determinados trechos, os comandos `while, do` e `for`:

- O `while` repete um comando (simples ou composto) enquanto uma determinada condição for verdadeira. A condição é testada antes da execução do comando. Sua sintaxe é:

```
while (condicao) {
   comando1;
   ...
}
```

- O `do` também repete um comando enquanto determinada condição for verdadeira, mas condição é testada após a execução do comando.

```
do {
   comando1;
   ...
 } while (condicao);
```

- O comando `for` repete um comando enquanto determinada condição for verdadeira. Admite que variáveis sejam inicializadas e modificadas durante sua execução.

```
for (inicializacao; condicao; modificacao das variaveis) {
   comando1;
   ...
}
```

Exemplo 12.1 *Neste exemplo, o* `comando1` *e o* `comando2` *são executados para i assumindo os valores* $1, 2, \cdots, 99$.

```
for ( i = 1; i < 100; i++) {
  comando1;
  comando2;
}
```

Exemplo 12.2 *Cálculo de um somatório:*

```
/* Calculo da soma de 1000 termos da serie harmonica */
#include <stdio.h>
main() {
  int i, max = 1000;
  float soma = 0;
  for (i = 1; i <= max; i++) soma = soma + 1.0/i;
  printf("\nSoma dos %d termos = %f", max, soma);
}
```

13 Funções

As funções em C são definidas da seguinte forma:

(1) um tipo do valor de retorno é escrito antes do nome da função;

(2) o nome deve ser seguido pela relação de parâmetros (variáveis) da função entre parênteses; cada parâmetro deve ter seu tipo escrito antes de cada nome;

(3) definição da função entre chaves;

(4) o valor de retorno é definido com um comando "return valor".

Exemplo 13.1 *A função* `Max` *definida a seguir calcula o valor máximo entre os números reais x e y:*

```
float Max(float x, float y) {
  if (x < y)
    return y;
  else
    return x;
}
```

Uma função pode não retornar valor. Neste caso ela deve ser declarada como `void`. Além disso, se a função tiver que modificar os valores das variáveis, então deve haver um asterisco antes do nome dos valores a serem modificados pela função, conforme o exemplo a seguir que troca "a" por "b" e "b" por "a":

```
void troca(int *a, int *b) {
  int aux;
  aux = *a;
  *a = *b;
  *b = aux;
}
```

É permitida a recursividade das funções, ou seja, uma função pode chamar a si mesma na sua definição, conforme no exemplo da função S, a seguir, que calcula o valor do somatório

$$S(n) = \sum_{k=0}^{n} \frac{1}{k^2+1}$$

```
float S(int n)  {
  if (n == 0)
    return 0;
  else
    return S(n-1) + 1.0/(n*n + 1);
}
```

Outro exemplo de recursividade é a função Fatorial definida a seguir:

```
long Fatorial(int n) {
  if (n <= 1) return 1;
  else return n * Fatorial(n - 1);
}
```

Algumas funções pré-definidas (em math.h) são: `sqrt, fabs, pow, log, exp, sin, cos, tan, asin, acos, atan, sinh, cosh, tanh`.

14 A função principal

Todo programa em C é formado por um conjunto de funções, sendo que a primeira a ser executada será sempre a função `main`.

```
main() {
   comando1;
   comando2;
   ...
}
```

15 Apontadores

Os apontadores (ponteiros) do C são variáveis que guardam endereços de outras variáveis ou de determinadas regiões da memória. São declarados colocando-se um asterisco antes do nome da variável e refenciados através do operador "&".

```
float *x;  /* x e' o endereco de uma variavel float */
int y, *z;

*x = -3.7; /* O conteudo de x e' -3,7 */
y = 11;
z = &y;    /* z e' o endereco de y */
```

16 Vetores e matrizes

A declaração de um vetor em C é feita colocando-se o tipo antes do nome do vetor, seguido do seu tamanho entre colchetes. Uma matriz é declarada de forma análoga, colocando-se dois abre e fecha colchetes na frente do nome da matriz. O primeiro elemento de um vetor tem índice nulo.

```
float v[5];     /* Vetor com 5 numeros reais: v[0], ..., v[4] */

int mat[3][7];  /* Matriz 3 x 7 formada de numeros inteiros cujos
                   elementos sao mat[0][0], mat[0][1], ..., mat[2][6] */
```

Exemplo 16.1 *Neste exemplo é definida uma matriz identidade* $M_{10\times10}$*:*

```
main() {
  int M[10][10];
  int i, j;
  for (i = 0; i <= 9; i++)
    for (j = 0; j <= 9; j++)
      if (i == j)
        M[i][j] = 1;
      else
        M[i][j] = 0;
}
```

17 Estruturas

Diversos tipos de dados podem ser agrupados em um único tipo chamado "estrutura". São definidas através da palavra `struct` seguido do nome da estrutura e dos tipos que a formarão, como no exemplo a seguir onde é definido uma estrutura "aluno":

Exemplo 17.1

```
struct aluno {
  char nome[30];
  char matricula[8];
  int curso;
  int aprovado;
}
```

Cada variável que forma uma estrutura pode ser referenciada colocando-se o nome da estrutura seguido de um ponto e do nome da variável desejada. Por exemplo, se for uma estrutura do tipo aluno, então são válidas as seguintes atribuições:

```
strcpy(X.nome, "Olezinho");  /* Nome de X                  */
X.curso = 29;                /* Curso de X                 */
X.aprovado = 0;              /* Se X foi aprovado ou nao */
```

Exemplo 17.2 *A seguir definimos uma estrutura formada por 3 números reais chamada "PontoNoR3". Depois, uma variável P desse tipo é criada e atribuiu-se os valores {3, -2, 4} a P.*

```
typedef struct {
   float x, y, z;
 } PontoNoR3;
 ...
 PontoNoR3 P;
 ...
 P.x = 3, P.y = -2, P.z = 4;   /* P = (3, -2, 4 ) */
```

18 Parâmetros fornecidos ao programa

Parâmetros podem ser fornecidos a um programa em C através de identificadores (por exemplo, `argc` e `argv`) fornecidos à função `main`.

`argc` é o número de argumentos e `argv[]` é uma matriz de strings que correspondem aos parâmetros.

Exemplo 18.1

```
main(int argc, char *argv[]) {
  int n;
  char p1[80], p2[80];
  n = argc;
  strcpy(p1, argv[1]);
  strcpy(p2, argv[2]);
  ...
}
```

19 Arquivos

A abertura de arquivos do disco em C é feita com o comando `fopen`. O primeiro parâmetro do `fopen` é o nome do arquivo do disco e o segundo parâmetro especifica se o arquivo vai ser aberto para leitura ("r"), para gravação ("w"), além de outras possibilidades.

A leitura/gravação de dados pode ser feita com `getc, fgets, fread, fwrite, fputs, fprintf,` além de outros comandos. Todo arquivo em C é declarado como sendo um apontador do tipo `FILE` (escrito em maiúsculas). Para fechar arquivos, usa-se um `fclose`.

Exemplo 19.1 *Neste exemplo um arquivo é aberto para leitura e seus caracteres são mostrados na tela.*

```
#include <stdio.h>
main(int argc, char *argv[])
{
  FILE *arq;
  int ch;

  arq = fopen(argv[1], "r");
  do {
    ch = getc(arq);
    putchar(ch);
  } while (ch != EOF);
  fclose(arq);
}
```

20 Gráficos

Os comandos gráficos do C estão na biblioteca GRAPHICS e podem ser usados depois que for feita uma chamada à função `initgraph`. Alguns dos comandos gráficos são:

`putpixel(x, y, cor)`	Desenha ponto (x, y) com a cor especificada
`line(a, b, c, d)`	Segmento de reta (a, b) – (c, d)
`circle(a, b, R)`	Circunferência de centro (a, b) e raio R

Exemplo 20.1

```
/* Testando alguns comandos graficos  */
#include <stdio.h>
#include <graphics.h>

void IniciaModoGrafico(void) {

/* Coloca a tela no modo grafico */

  int drivergrafico, modografico;

  drivergrafico = DETECT;
  initgraph(&drivergrafico, &modografico, "");
  if (graphresult() < 0)
    printf("ERRO na mudanca para o modo grafico.\n");
}

void DesenhaFiguras(void) {

/* Marca um ponto, desenha uma reta e uma circunferencia */

  putpixel(400, 150, 4);   /* ponto (400, 150) na cor 4 */
  line(10, 10, 200, 300);  /* reta                      */
  circle(300, 300, 50);    /* circunferencia            */
}

main() {
  IniciaModoGrafico();
  DesenhaFiguras();
  delay(3000);  /* pausa de 3000 milissegundos */
  closegraph(); /* encerra o modo grafico      */
}
```