LÓGICA DE PROGRAMAÇÃO

Resumo da Linguagem C

Sérgio Carlos Portari Júnior

Tipos de dados mais usados em C

Inteiro	int	2 bytes
Real	float	4 bytes
Caractere	char	1 byte
String	não existe*	xxxxxxx
Booleano	não existe	xxxxxxx

- * para utilizarmos uma string em C faremos uso de um vetor de caracteres preparado para responder como string, que será visto mais adiante. Seu tamanho será do número de caracteres reservados para o vetor em bytes.

Operadores matemáticos em C

operador	função	ex	res
=	atribuição	a=3	xxx
+	soma	21+4	25
-	subtração	21-4	17
*	multiplicação	21*4	84
/	divisão	21/4	5,25
%	resto divisão	21%4	1

(equivale ao mod no algoritmo)

Curiosidades da divisão

- Sendo a e b duas variáveis inteiras, onde a=7 e b=2.
- Se usarmos printf para ver o resultado, observem:
  - printf("a / b = %f: ",a/b); //resposta: 0.000
  - printf("a / b = %d: ",a/b);//resposta: 3
  - printf("a / b = %d: ",a%b);//resposta: 1
- Isto porque inteiro / inteiro dá inteiro, não podendo usar %f para mostrar o resultado (lembrem-se: int=2bytes e float=4bytes)

Função scanf (leia)

- A função scanf é uma das funções responsáveis por receber dados do usuário, entrados por exemplo pelo teclado, para que ele seja processado no programa. Ela equivale ao leia do algoritmo.
- A sintexa da função scanf é similar à sintaxe da função printf:
  - scanf(“expressão” ,lista de endereços);

Função scanf (leia)

- scanf(“expressão” ,lista de endereços);
- Expressão: contém formatos de leitura, como os utilizados na função printf, para indicar o tipo de dados que será lido (%d, %f, %c,...)
- Lista de endereços: contém os endereços de memória que receberão os dados inseridos pelo usuário.
  - Mas como eu vou colocar os endereços de memória????

Função scanf (leia)

- Simplesmente adicionando o caractere & antes do nome de uma variável.
  - Por exemplo:

```
#include <stdio.h>
int main()
{  int a;
   printf("Digite um valor para a: ");
  scanf("%d",&a);  //equivale a leia(a)
   printf("\nO valor lido para a foi %d.",a);
return 0;
}
```

Função scanf (leia)

- Se o programa fosse ler um caractere ao invés de um inteiro, ele ficaria da seguinte forma:

```
#include <stdio.h>
int main()
{  char a;
   printf("Digite um valor para a: ");
  scanf("%c",&a);  //equivale a leia(a)
   printf("\nO valor lido para a foi %c.",a);
return 0;
}
```

Função scanf (leia)

- Podemos usar a leitura formatada de mais de uma variável num mesmo scanf.
  - scanf("%d%d",&a,&b);

Digite o valor para a, ENTER, digite o valor para b e ENTER novamente para ler os dois valores no scanf acima.

Funções getch, getche, getchar

- Em se tratando de caractere, C possui várias funções para fazer sua leitura.
- A função getchar (biblioteca stdio.h) serve para receber um caractere único e atribuí-lo se necessário a uma variável.

```
#include <stdio.h>
int main()
{  char a;
   printf(“Digite um valor para a: ”);
   a=getchar();  // dispensa o endereço de memória
   printf(“\nO valor lido para a foi %c.”,a);
return 0;
}
```

Funções getch, getche, getchar

- A função getch e getche (biblioteca conio.h) servem para receber um caractere único e atribuí-lo se necessário a uma variável, mas getch oculta a leitura e getche mostra (idêntico ao getchar). As duas incluem o Enter automaticamente no fim da leitura.

```
#include <stdio.h>
#include <conio.h>
int main()
{  char a;
   printf("Digite um valor para a: ");
   a=getch();  // dispensa o endereço de memória
   printf("\nO valor lido para a foi %c.",a);
return 0;
}
```

Funções getch, getche, getchar

```
#include <stdio.h>
#include <conio.h>
int main()
{  char a;
   printf(“Digite um valor para a: ”);
   a=getche();  // dispensa o endereço de memória
   printf(“\nO valor lido para a foi %c.”,a);
return 0;
}
```

Problema com caracteres

- Agora, tente no mesmo programa utilizar mais de uma leitura de caracteres seguida (pode ser por scanf, getchar, getch, getche, diferentes ou iguais...)
- Veja o exemplo:

Problema com caracteres

```
#include <stdio.h>
#include <conio.h>
int main()
{  char a,b;
   printf(“Digite um valor para a: ”);
   a=getche();
   b=getchar();
   printf(“\nO valor lido para a foi %c e para b %c.”, a,b);
return 0;
}
```

Problema com caracteres

- Existe uma razão para que isso aconteça. A leitura de valores através da maioria das funções utiliza o *buffer* do teclado como repositório temporário dos caracteres que escrevemos.
- A utilização do buffer é visível pelo fato de podermos alterar o conteúdo antes de enviarmos ao processador (apertando backspace e alterando a letra). Enquanto não apertarmos Enter o conteúdo não é enviado para a variável.

Problema com caracteres

- Assim, sempre mandamos ao programa dois caracteres (um com a letra digitada e outro com o comando de encerramento de edição (ENTER), que como vimos na tabela ASC, cada tecla representa um número para o computador (o enter é o 13, assim como o A é o 65, B é 66, etc.).
- Quando a função que receberá o caractere recebe os dados, ela utiliza o primeiro caractere, sobrando o último (ENTER) no buffer do teclado.

Problema com caracteres

- Então, quando a segunda função que receberá o próximo caractere solicita ao buffer que aguarde chegar o enter (na tabela asc o 13), ele já está lá e “entende” que o usuário já digitou o caractere.
- Uma forma para prevenir essa situação é a inclusão de uma função que limpe o buffer antes da leitura dos caracteres.
- fflush(stdin);
- Veja o exemplo agora.

Problema com caracteres

```
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>
int main()
{  char a,b;
   printf("Digite um valor para a: ");
   a=getche();  //inclui o enter automaticamente
   fflush(stdin); //logo após digitar um caractere você já está
   b=getchar(); //digitando o b, que precisa do enter.
   printf("\nO valor lido para a foi %c e para b %c.\n",a,b);
   system("pause");
return 0;
}
```

Função putchar

- A função putchar é o complemento da função getchar, serve para escrever um caractere na tela.

```
#include <stdio.h>
int main()
{  char a;
   printf("Digite um valor para a: ");
   a=getchar();
   putchar(a);
return 0;
}
```

Exercícios

- Arquivos 08 e 09 PDF - WEbgiz

- Complementares:

Livro Fundamentos da programação de Computadores – Ana Fernanda Gomes Ascencio – Página 25 a 38 (resolvidos em algoritmo) e 38 a 40.