

Funções de Biblioteca e Funções para Manipulação de *Strings*

Prof^a.Dr^a.Thatyana de Faria Piola Seraphim (ECO)
Prof.Dr.Enzo Seraphim (ECO)

Universidade Federal de Itajubá

thatyana@unifei.edu.br
seraphim@unifei.edu.br

Biblioteca Padrão C

- ▶ **Bibliotecas** são conjuntos de funções que foram desenvolvidas e que podem ser usadas por outros programas sem que o programador se preocupe com o código dessas funções.
- ▶ Vantagens no uso de bibliotecas:
 - ▶ Organização do código.
 - ▶ Podem ser utilizadas em vários programas sem a necessidade de copiar grandes trechos de código.
- ▶ A biblioteca padrão do C é também conhecida como **libc**.
 - ▶ É uma biblioteca de rotinas padronizada da linguagem de programação C.
 - ▶ Contém operações comuns como tratamento de entrada/saída e cadeia de caracteres.
- ▶ Para muitos compiladores C, a biblioteca padrão está contida em um arquivo.
- ▶ Algumas implementações têm as funções relacionadas agrupadas em suas próprias bibliotecas por eficiência ou restrição de tamanho.

Funções de Bibliotecas

stdio.h

- ▶ A biblioteca **stdio** é um cabeçalho da biblioteca padrão do C.
- ▶ Seu nome vem da expressão inglesa *standard input-output header*, que significa cabeçalho padrão de entrada/saída.
- ▶ Possui definições de subrotinas referentes às operações de entrada/saída.
 - ▶ Como leitura de dados digitados no teclado .
 - ▶ Exibição de informações na tela do programa de computador.
- ▶ Possui numerosas definições de constantes, variáveis e tipos.
- ▶ É um dos cabeçalhos mais populares da linguagem de programação C, intensivamente utilizado tanto por programadores iniciantes quanto experientes.

#include <stdio.h>

Funções de Bibliotecas

`stdlib.h`

- ▶ **Stdlib.h** é um arquivo cabeçalho da biblioteca de propósito geral padrão da linguagem de programação C.
- ▶ O nome `stdlib` vem de *standard library*.
- ▶ Possui funções que são classificadas em:
 - ▶ Conversão de tipo.
 - ▶ Geração de sequências aleatórias.
 - ▶ Alocação e liberação de memória.
 - ▶ Controle de processos.
 - ▶ Matemática.

Funções de Bibliotecas

stdlib.h

```
double atof(const char *str);
```

- ▶ Converte uma *string* em um valor **float**.
- ▶ *String* deve conter um número de ponto flutuante válido.
- ▶ Se a função **atof()** for chamada com **100.00Hello**, o valor **100.00** será devolvido.

```
int atoi(const char *str);
```

- ▶ Converte uma *string* em um valor **inteiro**.
- ▶ A *string* deve conter um inteiro válido.
- ▶ Se a função **atoi()** for chamada com **123.45**, o valor inteiro **123** é devolvido e 0.45 é ignorado.

Funções de Bibliotecas

stdlib.h

```
int atol(const char *str);
```

- ▶ Converte uma *string* em um valor **long int**.
- ▶ A *string* deve conter um inteiro longo válido.
- ▶ Se a função **atol()** for chamada com **123.45**, o valor inteiro **123** é devolvido e 0.45 é ignorado.

```
int abs(int num);
```

- ▶ Devolve o valor absoluto de seu argumento inteiro.

abs(—)

Funções de Bibliotecas

stdlib.h

```
int rand(void);
```

- ▶ Gera um número aleatório.
- ▶ Cada vez que a função **rand()** é chamada, é devolvido um número entre zero e RAND_MAX.

```
void srand(unsigned int seed);
```

- ▶ Estabelece um ponto de partida para a sequência gerada por **rand()**.
- ▶ É usada para permitir que programas gerem sequências diferentes de números aleatórios a cada execução.

Exemplo 01 - Sorteio da face do dado

Faça um programa que recebe como parâmetro a quantidade máxima de faces para um sorteio aleatório.

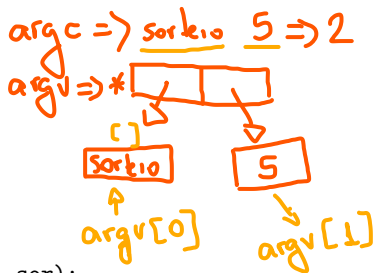


Funções de Bibliotecas

stdlib.h

Exemplo 01

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <time.h>
4  int main(int argc, char *argv[]) {
5      int faces, sor;
6      srand(time(NULL));
7      if((argc == 2)) {
8          faces = abs(atoi(argv[1]));
9          if(faces > 9) {
10             faces = 10;
11         } //end if
12         sor = rand() % faces;
13         printf("Sorteado a face: %d\n", sor);
14     } else {
15         printf("Digite: nomeExecutavel QuatFaces\n");
16     } //end else
17     return 0;
18 } //end main
```



Funções de Bibliotecas

stdlib.h

```
div_t div(int num, int den);
```

- ▶ Devolve o quociente e o resto da operação num/den em uma estrutura do tipo `div_t`.
- ▶ O tipo de estrutura `div_t` tem dois campos: `int quot, rem`.

```
int system(const char *command);
```

system ("Pause");

- ▶ Passa uma *string* como um comando para o processador de comandos do sistema operacional.
- ▶ O valor retornado pela função `system()` é definido pela implementação.

Funções de Bibliotecas

stdlib.h

```
void *calloc(size_t num, size_t size);
```

- ▶ Aloca uma quantidade de memória igual a $num * size$.
- ▶ Aloca memória suficiente para um vetor de num objetos de tamanho $size$.

```
void free(void *ptr);
```

- ▶ Libera a memória alocada, tornando a memória disponível para alocação futura.

```
void *malloc(size_t size);
```

- ▶ Retorna um ponteiro para o primeiro *byte* de uma região de memória de tamanho $size$ que foi alocada.

Funções de Bibliotecas

`math.h`

```
double cos(double num);
```

- ▶ Retorna o coseno de *num* que deve estar em radianos ($\text{rad} = (\text{PI}/180) \cdot \text{grau}$).

```
double log(double num);
```

- ▶ Retorna o logaritmo natural de um número.

```
double pow(double base, double exp);
```

- ▶ Retorna a base elevada à potência
 $\text{pow}(\text{base}, \text{exp}) = (\text{base}^{\text{exp}})$.

Funções de Bibliotecas

math.h

```
double sin(double num);
```

- ▶ Retorna o seno de *num* que deve estar em radianos ($\text{rad} = (\text{PI}/180) \cdot \text{grau}$).

```
double sqrt(double num);
```

- ▶ Retorna a raiz quadrada de um número.

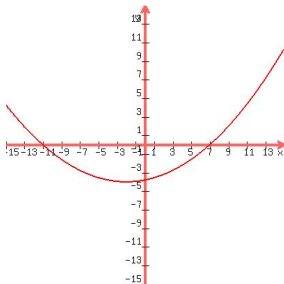
```
double tan(double num);
```

- ▶ Retorna a tangente de *num* que deve estar em radianos ($\text{rad} = (\text{PI}/180) \cdot \text{grau}$).

Exemplo 02 - Eq. 2º Grau

Faça um programa que recebe como parâmetros os coeficientes a, b e c de uma equação de segundo grau. Calcule e imprime as raízes da equação, segundo a fórmula abaixo.

$$X_{1,2} = \frac{-b \pm \sqrt{b^2 - 4 * a * c}}{2 * a}$$



Funções de Bibliotecas

math.h

Exemplo 02

```
1  #include <stdio.h>
2  #include <math.h>
3  int main(int argc, char *argv[]){
4      float a,b,c,delta,x1,x2;
5      if(argc == 4){
6          a = atof(argv[1]);
7          b = atof(argv[2]);
8          c = atof(argv[3]);
9          printf("%fx2 + %fx + %f.\n", a,b,c);
10         delta = pow(b,2) - (4*a*c);
11         if(delta >= 0){
12             x1 = (-b+sqrt(delta))/(2*a);
13             x2 = (-b-sqrt(delta))/(2*a);
14             printf("Raizes:%f e %f.\n",x1,x2);
```

argc
segundo 1 3 5
argv \Rightarrow [0] [1] [2] [3]

Exemplo 02 (Cont...)

```
15     }//end if(delta>=0)
16     else{
17         delta = sqrt(-delta)/(2*a);
18         x1 = -b/(2*a);
19         x2 = -b/(2*a);
20         printf("Raizes sao (%f+%fi) e (%f-%fi).\n",
21             x1, delta, x2, delta);
22     }//end else
23 }else{
24     printf("Insira os coeficientes na forma: a b c\n");
25 }//end else
26 return 0;
27 }//end main
```


Funções de Bibliotecas

`time.h`

```
int clock(void);
```

- ▶ Retorna um valor aproximado do tempo de execução do programa que a chama.

```
double difftime(double t1, double t2);
```

- ▶ Retorna a diferença em segundos, entre *t1* e *t2*.

```
double time(double *time);
```

- ▶ Retorna o horário atual do calendário do sistema.

CLOCKS_PER_SEC: contém o número de *clock* em um segundo.

Exemplo 03 - Contagem de Tempo

Faça um programa que recebe como parâmetro um tempo em segundos. O programa deve ficar este tempo parado sem realizar nenhuma operação.

Funções de Bibliotecas

time.h

Exemplo 03

```
1  #include <stdio.h>
2  #include <time.h>
3  int main(int argc, char *argv[]){
4      double inicio, fim, dif;
5      int atraso;
6      inicio = clock();
7      fim = clock();
8      if(argc == 2){
9          atraso = abs(atoi(argv[1]));
10         if(atraso > 60){
11             atraso=60;
12         } //end if
13         printf("Aguarde \n");
14         dif = (fim - inicio) / CLOCKS_PER_SEC;
```

Funções de Bibliotecas

time.h

Exemplo 03 – cont...

```
15     while(dif < atraso){
16         fim = clock();
17         dif = (fim - inicio) / CLOCKS_PER_SEC;
18     }//end while
19 }//end if
20 else{
21     printf("Sintaxe: atraso TEMPOSEGUNDOS \n");
22 }//end else
23 return 0;
24 }//end main
```

a = getchar() (d)
getchar();

Funções para Manipulação de *Strings*

Funções de Manipulação de *String*

- ▶ A biblioteca `<string.h>` fornece um conjunto de funções para manipulação de *strings*.
- ▶ Estas funções são úteis, devido ao fato de que em C toda *string* é um vetor de caracteres.
 - ▶ Por ser um vetor de caracteres as *strings* herdam uma série de limitações pertinentes ao vetores.
 - ▶ Isto se deve ao fato de a linguagem C não suportar operações com vetores utilizando diretamente os operadores aritméticos e relacionais.
- ▶ Todo programa que manipular *strings* deve tomar cuidado com as seguintes limitações:
 1. *Strings* não podem ser atribuídas com o operador de atribuição (`=`), embora possam ser inicializadas.
 2. *Strings* não podem ser comparadas com os operadores relacionais (`==`, `!=`, `>`, `>=`, `<`, `<=`).

Funções de Manipulação de *String*

- ▶ Vetores e ponteiros são muito usados em C e estão de certo modo relacionados às *strings*.
- ▶ Como o nome de um vetor é na verdade um **ponteiro**, existe uma outra maneira de definir uma variável *strings*, ou seja, ela pode ser um **ponteiro** para o tipo *char*.

- ▶ Por exemplo:

```
char nome[20];
```



```
char *profissao;
```

- ▶ A variável *nome* na primeira declaração guarda o endereço do primeiro *byte* de um local na memória que pode armazenar até 19 caracteres. O último caracter é sempre *null*, ou seja, apenas indica o final do nome.

Funções de Manipulação de *String*

Diferença

A diferença entre definir uma variável *string* como vetor e uma variável como ponteiro **está na memória**.

- ▶ **Definição como vetor:** a memória é automaticamente reservada, como para qualquer vetor.
- ▶ **Definição como ponteiro:** não existe memória reservada.
- ▶ Quando uma *string* é escrita entre aspas, o compilador cria a *string* seguida por um *null* em algum lugar no programa objeto.
- ▶ Por exemplo:

```
char *nome="Jose Pereira";
```


Funções de Manipulação de *String*

```
char *nome="Jose Pereira";
```

- ▶ Essa forma de atribuição se aplica bem à variáveis *strings* definidas como ponteiros para **char**.
- ▶ Para vetores, existe uma série de funções de biblioteca prontas.
- ▶ As funções para manipulação de *strings* são definidas no arquivo **string.h**.

Funções de Manipulação de *String*

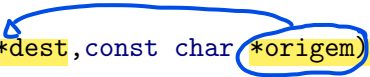
string.h

char a[20], b[50]

`int strlen(char *str);`

- ▶ Conta o número de caracteres armazenados em uma *string*, antes do `'\0'`.
- ▶ Retorna o número de caracteres da *string*.
- ▶ **str**: a *string* que terá seu tamanho calculado.

`char strcpy(char *dest, const char *origem);`



- ▶ Copia o conteúdo de uma *string* para outra. Deve ser utilizada para atribuição de *strings*, no lugar do operador de atribuição.
- ▶ Retorna o endereço do primeiro caracter da *string* dest.
- ▶ **dest**: *string* que vai receber o conteúdo.
- ▶ **origem**: *string* cujo conteúdo será copiado.

Funções de Manipulação de *String*

string.h

```
char *strncpy(char *dest, char *origem, int n);
```

- ▶ Copia no máximo **n** caracteres de uma *string* para a outra.
- ▶ Não coloca '\0' no final de dest, a não ser que tenha atingido o final da *string* origem.
- ▶ Retorna o endereço do primeiro caracter da *string* dest.
- ▶ **dest**: *string* que receberá o conteúdo.
- ▶ **origem**: *string* cujo conteúdo será copiado.
- ▶ **n**: número máximo de caracteres a ser copiado.

Funções de Manipulação de *String*

string.h

```
char *strcmp(char *s1, char *s2);
```

- ▶ Compara o conteúdo de duas *strings*. Esta função deve ser utilizada em substituição aos operadores relacionais no caso de uso com *strings*.
- ▶ Retorna:
 - ▶ 0 (zero) se o conteúdo das duas *strings* são iguais.
 - ▶ Um valor maior que 0, se o conteúdo de s1 for maior que s2.
 - ▶ Um valor menor que 0, se o conteúdo de s1 é menor que s2.
- ▶ Maior e menor não se refere ao tamanho da *string*, mas a posição, quando ordenadas de forma ascendente.
- ▶ s1 e s2: as duas *strings* a serem comparadas.

Funções de Manipulação de *String*

string.h

```
char *strncmp(char *s1, char *s2, int n);
```

- ▶ Compara apenas um trecho do início de duas *strings* com tamanho especificado.
- ▶ Retorna:
 - ▶ 0 (zero) se o conteúdo das duas *strings* são iguais.
 - ▶ Um valor maior que 0, se o conteúdo de *s1* for maior que *s2*.
 - ▶ Um valor menor que 0, se o conteúdo de *s1* é menor que *s2*.
- ▶ Maior e menor não se refere ao tamanho da *string*, mas a posição, quando ordenadas de forma ascendente.
- ▶ *s1* e *s2*: as duas *strings* a serem comparadas.
- ▶ *n*: número de caracteres a serem comparados.

Funções de Manipulação de *String*

string.h

Exemplo 04 - Conversão maiúsculo–minúsculo e vice-versa

Faça um programa que recebe como parâmetro uma *string* e faz a conversão da *string* em maiúsculo ou minúsculo, segundo os valores da tabela ASCII.

[32]=	[48]=0	[64]=@	[80]=P	[96]=`	[112]=p	[128]=	[144]=	[160]=	[176]=°	[192]=À	[208]=Ð	[224]=à	[240]=ð
[33]=!	[49]=1	[65]=A	[81]=Q	[97]=a	[113]=q	[129]=	[145]=	[161]=i	[177]=±	[193]=Á	[209]=Ñ	[225]=á	[241]=ñ
[34]="	[50]=2	[66]=B	[82]=R	[98]=b	[114]=r	[130]=	[146]=	[162]=ç	[178]=²	[194]=Â	[210]=Ò	[226]=â	[242]=ò
[35]=#	[51]=3	[67]=C	[83]=S	[99]=c	[115]=s	[131]=	[147]=	[163]=f	[179]=³	[195]=Ã	[211]=Ó	[227]=ã	[243]=ó
[36]=\$	[52]=4	[68]=D	[84]=T	[100]=d	[116]=t	[132]=	[148]=	[164]=m	[180]='	[196]=Ä	[212]=Ô	[228]=ä	[244]=ô
[37]=%	[53]=5	[69]=E	[85]=U	[101]=e	[117]=u	[133]=	[149]=	[165]=¥	[181]=µ	[197]=Å	[213]=Õ	[229]=å	[245]=õ
[38]=&	[54]=6	[70]=F	[86]=V	[102]=f	[118]=v	[134]=	[150]=	[166]=!	[182]=¶	[198]=Æ	[214]=Ö	[230]=æ	[246]=ö
[39]='	[55]=7	[71]=G	[87]=W	[103]=g	[119]=w	[135]=	[151]=	[167]=§	[183]=·	[199]=Ç	[215]=×	[231]=ç	[247]=÷
[40]=([56]=8	[72]=H	[88]=X	[104]=h	[120]=x	[136]=	[152]=	[168]="	[184]=,	[200]=È	[216]=Ø	[232]=è	[248]=ø
[41]=)	[57]=9	[73]=I	[89]=Y	[105]=i	[121]=y	[137]=	[153]=	[169]=©	[185]=¹	[201]=É	[217]=Ù	[233]=é	[249]=ù
[42]=*	[58]=:	[74]=J	[90]=Z	[106]=j	[122]=z	[138]=	[154]=	[170]=ª	[186]=º	[202]=Ê	[218]=Ú	[234]=ê	[250]=ú
[43]=+	[59]=;	[75]=K	[91]=[[107]=k	[123]={	[139]=	[155]=	[171]=«	[187]=»	[203]=Ë	[219]=Û	[235]=ë	[251]=û
[44]=,	[60]<	[76]=L	[92]=\	[108]=l	[124]=	[140]=	[156]=	[172]=¬	[188]=¼	[204]=Ì	[220]=Ü	[236]=ì	[252]=ü
[45]=-	[61]=	[77]=M	[93]=]	[109]=m	[125]=}	[141]=	[157]=	[173]=	[189]=½	[205]=Í	[221]=Ý	[237]=í	[253]=ý
[46]=.	[62]=>	[78]=N	[94]=^	[110]=n	[126]=~	[142]=	[158]=	[174]=®	[190]=¾	[206]=Î	[222]=Þ	[238]=î	[254]=þ
[47]=/	[63]=?	[79]=0	[95]=_	[111]=o	[127]=	[143]=	[159]=	[175]=	[191]=¿	[207]=Ï	[223]=ß	[239]=ï	[255]=ÿ

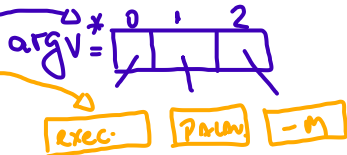
Funções de Manipulação de *String*

string.h

Exemplo 04

```
1 #include <stdio.h>
2 #include <string.h>
3 int main(int argc, char *argv[]) {
4     int i, tam, conv=32, inf=65, sup=90;
5     char pal[100];
6     if((argc == 3)){
7         strcpy(pal, argv[1]);
8         tam = strlen(pal);
9         if((strcmp(argv[2], "-M")==0) ||
10            (strcmp(argv[2], "-m")==0)){
11             // palavra em Maiusculo
12             if(strcmp(argv[2], "-M")==0){
13                 inf=97;
```

$argc \Rightarrow$ executavel **palavra** **(-M)** $\Rightarrow 3$



Funções de Manipulação de *String*

string.h

Exemplo 04 – cont...

```
14         sup=122;
15         conv=-32;
16     }//end if(strcmp(argv[2],"-M")==0)
17     else{
18         // palavra em Minusculo
19         inf=65;
20         sup=90;
21         conv=32;
22     }//end else
23     for(i=0; i<tam; i++){
24         if((pal[i] >= inf) && (pal[i] <= sup)){
25             pal[i] += conv;
26         }//end if
27     }//end for
```


Funções de Manipulação de *String*

string.h

Exemplo 04 – cont...

```
28     printf("Palavra: %s\n", pal);
29 } //end if((strcmp(argv[2], "-M")==0) ||
30 //      (strcmp(argv[2], "-m")==0))
31 else{
32     //nomeExecutavel: nome do arquivo executavel
33     //palavra: palavra digitada a ser convertida
34     //ABC-abc: -M (maiusculo) -m (minusculo)
35     printf("Sintaxe: nomeExecutavel palavra ABC-abc");
36 } //end else
37 }else{
38     printf("Sintaxe: nomeExecutavel palavra ABC-abc");
39 } //end else
40 return 0;
41 } //end main
```

Funções de Manipulação de *String*

string.h

Exemplo 05 – Gerador de senha aleatório

Faça um programa que recebe **dois parâmetros**: a quantidade de **caracteres** que formarão uma senha que será gerada aleatoriamente e um outro parâmetro que indica se a senha gerada deve ser maiúscula ou minúscula, segundo os valores da tabela ASCII.

[32]=	[48]=0	[64]=@	[80]=P	[96]=`	[112]=p	[128]=	[144]=	[160]=	[176]=°	[192]=À	[208]=Ð	[224]=à	[240]=ð
[33]=!	[49]=1	[65]=A	[81]=Q	[97]=a	[113]=q	[129]=	[145]=	[161]=i	[177]=±	[193]=Á	[209]=Ñ	[225]=á	[241]=ñ
[34]="	[50]=2	[66]=B	[82]=R	[98]=b	[114]=r	[130]=	[146]=	[162]=ç	[178]=²	[194]=Â	[210]=Ò	[226]=â	[242]=ò
[35]=#	[51]=3	[67]=C	[83]=S	[99]=c	[115]=s	[131]=	[147]=	[163]=é	[179]=³	[195]=Ã	[211]=Ó	[227]=ã	[243]=ó
[36]=\$	[52]=4	[68]=D	[84]=T	[100]=d	[116]=t	[132]=	[148]=	[164]=¸	[180]=´	[196]=Ä	[212]=Ô	[228]=ä	[244]=ô
[37]=%	[53]=5	[69]=E	[85]=U	[101]=e	[117]=u	[133]=	[149]=	[165]=¥	[181]=µ	[197]=Å	[213]=Õ	[229]=å	[245]=õ
[38]=&	[54]=6	[70]=F	[86]=V	[102]=f	[118]=v	[134]=	[150]=	[166]=¡	[182]=¶	[198]=Æ	[214]=Ö	[230]=æ	[246]=ö
[39]='	[55]=7	[71]=G	[87]=W	[103]=g	[119]=w	[135]=	[151]=	[167]=§	[183]=·	[199]=Ç	[215]=×	[231]=ç	[247]=÷
[40]=([56]=8	[72]=H	[88]=X	[104]=h	[120]=x	[136]=	[152]=	[168]=¨	[184]=,	[200]=È	[216]=Ø	[232]=è	[248]=ø
[41]=)	[57]=9	[73]=I	[89]=Y	[105]=i	[121]=y	[137]=	[153]=	[169]=©	[185]=¹	[201]=É	[217]=Ù	[233]=é	[249]=ù
[42]=*	[58]=:	[74]=J	[90]=Z	[106]=j	[122]=z	[138]=	[154]=	[170]=ª	[186]=º	[202]=Ê	[218]=Ú	[234]=ê	[250]=ú
[43]=+	[59]=;	[75]=K	[91]=[[107]=k	[123]={	[139]=	[155]=	[171]=«	[187]=»	[203]=Ë	[219]=Û	[235]=ë	[251]=û
[44]=,	[60]<	[76]=L	[92]=\	[108]=l	[124]=	[140]=	[156]=	[172]=¬	[188]=¼	[204]=Ì	[220]=Ü	[236]=ì	[252]=ü
[45]=-	[61]=	[77]=M	[93]=]	[109]=m	[125]=}	[141]=	[157]=	[173]=	[189]=½	[205]=Í	[221]=Ý	[237]=í	[253]=ý
[46]=.	[62]=>	[78]=N	[94]=^	[110]=n	[126]=~	[142]=	[158]=	[174]=®	[190]=¾	[206]=Î	[222]=Þ	[238]=î	[254]=þ
[47]=/	[63]=?	[79]=O	[95]=_	[111]=o	[127]=	[143]=	[159]=	[175]=¯	[191]=¸	[207]=Ï	[223]=ß	[239]=ï	[255]=ÿ

Funções de Manipulação de *String*

string.h

Exemplo 05

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <string.h>
4  #include <time.h>
5  int main(int argc, char *argv[]){
6      int i, digit = 6, letra = 97;
7      char senha[100];
8      srand(time(NULL));
9      if(argc == 3){
10         digit = abs(atoi(argv[1]));
11         if(digit >= 100){
12             digit = 99;
13         } //end if
```

Funções de Manipulação de *String*

string.h

Exemplo 05 – cont...

```
14     if((strcmp(argv[2], "-M")==0) ||
15         (strcmp(argv[2], "-m")==0)){
16         if(strcmp(argv[2], "-M")==0){ //letras Maiusculas
17             letra = 65;
18         }//end if(strcmp(argv[2], "-M")==0)
19         else{
20             // letras Minusculas
21             letra = 97;
22         }//end else
23         for(i=0; i<digit; i++){
24             senha[i]=letra+rand()%26;
25         }//end for
26         senha[i] = '\0';
```

Funções de Manipulação de *String*

string.h

Exemplo 05 – cont...

```
27     printf("Senha:%s\n", senha);
28 } //end if((strcmp(argv[2], "-M")==0)...
29 else{
30     //nomeExecutavel: nome do arquivo executavel
31     //qtLetrasSenha: quantidade de letras na senha
32     //ABC-abc: -M (maiusculo) -m (minusculo)
33     printf("Sintaxe:nomeExecutavel qtLetrasSenha ABC-abc");
34 } //end else
35 } //end if(argc == 3)
36 else{
37     printf("Sintaxe:nomeExecutavel qtLetrasSenha ABC-abc");
38 } //end else
39 return 0;
40 } //end main
```

Funções de Manipulação de *String*

string.h

```
char *strcat(char *dest, char *origem);
```

- ▶ Concatena o conteúdo da *string* de **origem** ao final da *string* **dest**, preservando o valor de **dest**.
- ▶ Retorna o endereço do primeiro caracter da *string* **dest**.
- ▶ **dest**: *string* que irá receber, no seu final, o conteúdo. Deve ter tamanho suficiente para armazenar o conteúdo atual mais o novo.
- ▶ **origem**: *string* cujo conteúdo será adicionado ao final da outra.

Funções de Manipulação de *String*

string.h

```
char *strncat(char *dest, char *origem, int n);
```

- ▶ Concatena não mais que **n** caracteres da *string*.
- ▶ Retorna o endereço do primeiro caracter da *string* dest.
- ▶ **dest**: *string* que irá receber, no seu final, o conteúdo. Deve ter tamanho suficiente para armazenar o conteúdo atual mais o novo.
- ▶ **origem**: *string* cujo conteúdo será adicionado ao final da outra.
- ▶ **n**: número de caracteres a ser concatenado.

Funções de Manipulação de *String*

string.h

```
char *strstr(char *s1, char *s2);
```

- ▶ Compara se uma *string* está contida dentro de outra *string*.
- ▶ Retorna o ponteiro para a primeira ocorrência de s2 em s1, ou *null* se não encontrou.
- ▶ s1: *string* a ser comparada.
- ▶ s2: *string* que contém a sequência de caracteres a ser comparada.

Funções de Manipulação de *String*

string.h

```
char *strchr(char *str, int n);
```

- ▶ Localiza a primeira ocorrência de **n** (convertida para **char**) na *string* **str**.
- ▶ Retorna o ponteiro para a primeira ocorrência do caracter **n** em **str**.
- ▶ **str**: *string*.
- ▶ **n**: caracter a ser localizado.

Funções de Manipulação de *String*

Uma função muito útil para manipulação de *string* está na biblioteca `stdio.h`.

print("Bom dia");
sprint(a, "Bom dia");

`int sprintf(char *str, char *format, ...);`

- ▶ É idêntica à função **printf()** exceto que a saída é colocada na variável **str**.
- ▶ Retorna o número total de caracteres escritos em **str**
- ▶ **str**: vetor de caracteres que armazena a *string* resultante.
- ▶ **format**: *string* que contém o texto a ser armazenado em **str**.

Funções de Manipulação de *String*

$\%d = \&$
 $\%f = \&$
 $\%c = \&$

Exemplo usando `sprintf()`

```
1  #include <stdio.h>
2  int main(int argc, char *argv[]){
3      char nome[30];
4      char msg[100];
5      printf("Digite seu nome: ");
6      scanf("%s", nome);
7      → sprintf(msg, "%s seja bem vindo(a) a Unifei!", nome);
8      printf("%s \n", msg);
9      return 0;
10 }
```

Funções de Manipulação de *String*

scanf ("%s", end); {Rua A, ...}
char a[50]; gets (a) {RUA A, BAIRRO, CIDADE}

`char *gets(char *str);`

- ▶ Os caracteres são lidos até que seja recebido um caractere de nova linha.
- ▶ Não há limite de caracteres que a função **gets** irá ler.
- ▶ Retorna o valor *null* se houve algum problema.
- ▶ **str**: vetor de caracteres que armazena a *string*.

Exemplo usando gets()

```
1  #include <stdio.h>
2  int main(int argc, char *argv[]){
3      char str[256];
4      printf("Digite o endereço completo: ");
5      gets(str);
6      printf("Endereço = %s \n",str);
7      return 0;
8  }//end main
```