Strings

ECM404

Caracteres – Char

- **Char** é um tipo de dado utilizado para armazenar um único caractere. Os caracteres armazenados na variável são delimitados por **aspas simples**.

```
int main (int argc, char *argv[]){

  char c = 'a';

  return 0;
}
```

- Exemplos:

```
'A'   'a'   '1'   '?'   '+'   '%%'   '\n'   '\"'
```

Caracteres – Char

- A leitura de um **char** é igual ao que fazemos com valores inteiros ou reais.

```c
int main (int argc, char *argv[]){
  char c;

  printf("Digite um char: ");
  scanf("%c", &c);
  printf("O char digitado foi: %c\n", c);

  return 0;
}
```

```
Digite um char: a
O char digitado foi: a
```

Caracteres – Char

- E quando fazemos a leitura após outra leitura?

```c
int main (int argc, char *argv[]){
  int valor;
  char c;

  printf("Digite um numero inteiro: ");
  scanf("%i", &valor);
  printf("O numero digitado foi: %i\n", valor);

  printf("Digite um char: ");
  scanf("%c", &c);
  printf("O char digitado foi: %c\n", c);

  return 0;
}
```

```
Digite um numero inteiro: 1
O numero digitado foi: 1
Digite um char: O char digitado foi:
```

Caracteres – Char

- É necessário limpar o ***buffer de entrada*** antes de ler o **char**

```c
int main (int argc, char *argv[]){
  int valor;
  char c;

  printf("Digite um numero inteiro: ");
  scanf("%i", &valor);
  printf("O numero digitado foi: %i\n", valor);
  setbuf(stdin, 0);
  printf("Digite um char: ");
  scanf("%c", &c);
  printf("O char digitado foi: %c\n", c);

  return 0;
}
```

```
Digite um numero inteiro: 1
O numero digitado foi: 1
Digite um char: a
O char digitado foi: a
```

String – Definição

- String:
  - Sequência de caracteres que pode representar uma palavra ou uma frase;
  - Na linguagem C, as strings são representadas como um array do tipo **char** e delimitadas por **aspas duplas**.

```
int main (int argc, char *argv[]){

  char str_1[6];

  char str_2[] = "eu sou uma string";

  return 0;
}
```

String – Definição

- Toda **String** é terminada com o caractere **'\0'**. Esse caractere também deve ser considerado na declaração do tamanho da string.

```
char s[] = "OLA Mundo";
```

s[0]	s[1]	s[2]	s[3]	s[4]	s[5]	s[6]	s[7]	s[8]	s[9]
'O'	'L'	'A'	' '	'M'	'u'	'n'	'd'	'o'	'\0'

```
char t[10] = "Piccolo";
```

t[0]	t[1]	t[2]	t[3]	t[4]	t[5]	t[6]	t[7]	t[8]	t[9]
'P'	'i'	'c'	'c'	'o'	'l'	'o'	'\0'	?	?

String – Exibição

- Podemos exibir uma string utilizando o especificador de formatação

  %s ou apenas um caractere desta string utilizando **%c**.

```c
int main (int argc, char *argv[]){
  char str[] = "eu sou uma string";

  printf("String completa: %s\n",str);
  printf("Um caractere: %c\n", str[0]);

  return 0;
}
```

```
String completa: eu sou uma string
Um caractere: e
```

String – Leitura

- Para a leitura de strings, será necessária uma função específica por conta do comando **scanf** ler apenas palavras (strings sem espaços).

```c
int main (int argc, char *argv[]){
  char str[50];

  printf("Digite alguma coisa: ");
  scanf("%s",str);

  printf("String lida: %s\n",str);

  return 0;
}
```

```
Digite alguma coisa: alguma coisa
String lida: alguma
```

String – Leitura

- Para realizar a leitura de strings, utilizaremos a função **fgets**.
  - A função **fgets** recebe três parâmetros de entrada:
    - Variável onde será armazenada a string que será lida;
    - Limite máximo de caracteres a serem lidos;
    - Local de onde a string será lida.

String – Leitura

- Exemplo de leitura:

```c
int main (int argc, char *argv[]){
  char str[50];

  printf("Digite alguma coisa: ");
  fgets(str,50,stdin);

  printf("String lida: %s\n",str);

  return 0;
}
```

```
Digite alguma coisa: alguma coisa
String lida: alguma coisa
```

String – Leitura

- Exemplo de leitura após outra leitura:

```c
int main (int argc, char *argv[]){
  char str[50];
  int valor;

  printf("Digite um valor: ");
  scanf("%i", &valor);
  printf("Valor lido: %i\n",valor);

  printf("Digite alguma coisa: ");
  fgets(str,50,stdin);
  printf("String lida: %s\n",str);

  return 0;
}
```

```
Digite um valor: 10
Valor lido: 10
Digite alguma coisa: String lida:
```

String – Leitura

- Exemplo de leitura após outra leitura:

```
int main (int argc, char *argv[]){
  char str[50];
  int valor;

  printf("Digite um valor: ");
  scanf("%i", &valor);
  printf("Valor lido: %i\n",valor);
  setbuf(stdin, 0);
  printf("Digite alguma coisa: ");
  fgets(str,50,stdin);
  printf("String lida: %s\n",str);

  return 0;
}
```

```
Digite um valor: 10
Valor lido: 10
Digite alguma coisa: alguma coisa
String lida: alguma coisa
```

String – Leitura

- **CUIDADO:** Ao utilizar a função **fgets**, o último caractere da string **pode ser** o **'\n'**. Exemplo:

```
char s[10];
//Usuário digita "Goku"
fgets(s,10,stdin);
```

s[0]	s[1]	s[2]	s[3]	s[4]	s[5]	s[6]	s[7]	s[8]	s[9]
'G'	'o'	'k'	'u'	'\n'	'\0'	?	?	?	?

String – Leitura

- Exemplo de leitura tratando os problemas:

```c
#include <stdio.h>
#include <string.h>

#define MAX 100

void lerString(char s[]){
  setbuf(stdin,0);
  fgets(s, MAX, stdin);
  if(s[strlen(s) - 1] == '\n'){
    s[strlen(s) - 1] = '\0';
  }
}
```

A função **strlen** retorna a quantidade de caracteres dentro de uma string antes do caractere '\0'.

String – Cópia

- Exemplo de como criar uma cópia de uma string:

```
int main (int argc, char *argv[]){
  char str_1[] = "Teste de copia";
  char str_2[MAX], str_3[MAX];
  int i;

  for(i = 0; i < strlen(str_1); i++ ){
    str_2[i] = str_1[i];
  }

  strcpy(str_3,str_1);

  printf("str_1: %s\n", str_1);
  printf("str_2: %s\n", str_2);
  printf("str_3: %s\n", str_3);

  return 0;
}
```

strcpy(char destino[], **char** origem[])

A função **strcpy** copia o conteúdo da string origem para a string destino. A string destino deve ser longa o bastante para caber o conteúdo.

String – Concatenação

- Exemplo de como concatenar duas strings:

```
int main (int argc, char *argv[]){
  char str_1[MAX] = "Power ";
  char str_2[MAX] = "Ranger";

  strcat(str_1,str_2);

  printf("str_1: %s\n", str_1);
  printf("str_2: %s\n", str_2);

  return 0;
}
```

strcat(char destino[], **char** origem[])

A função **strcat** copia o conteúdo da string origem para o final da string destino. A string destino deve ser longa o bastante para caber o conteúdo.

String – Comparação

- Exemplo de comparação entre duas strings:

```
int main (int argc, char *argv[]){
  char str_1[MAX] = "A";
  char str_2[MAX] = "A";
  int valor;

  valor = strcmp(str_1,str_2);

  printf("valor: %i\n", valor);

  return 0;
}
```

strcmp(char s1[], **char** s2[])

A função **strcmp** compara duas strings:

- s1 < s2 -> retorna -1;
- s1 = s2 -> retorna 0;
- s1 > s2 -> retorna 1;

String – Conversão Numérica

- Exemplo de conversão de uma string para um número inteiro:

```c
#include <stdio.h>
#include <stdlib.h>

int main (int argc, char *argv[]){
  char str_1[MAX] = "1";
  int valor;

  valor = atoi(str_1) + 1;

  printf("valor: %i\n", valor);

  return 0;
}
```

atoi(**char** s1[])

A função **atoi** converte a s1 para um valor inteiro, quando possível. Caso nenhum valor válido é encontrado, retorna 0.

String – Conversão Numérica

- Exemplo de conversão de uma string para um número real:

```c
#include <stdio.h>
#include <stdlib.h>

int main (int argc, char *argv[]){
  char str_1[MAX] = "1.23";
  float valor;

  valor = atof(str_1) + 1.11;

  printf("valor: %f\n", valor);

  return 0;
}
```

atof(**char** s1[])

A função **atof** converte a s1 para um valor real, quando possível. Caso nenhum valor válido é encontrado, retorna 0.0.

String – Conversão Numérica

- Exemplo de conversão de um número para uma string:

```
int main (int argc, char *argv[]){
  char str_1[MAX];
  float valor = 1.2345;

  sprintf(str_1, "%.2f",valor);
  printf("str_1: %s\n", str_1);

  return 0;
}
```

sprintf(**char** s1[], **char** s2[])

A função **sprintf** ao invés de “imprimir” o texto (s2) na saída padrão, “imprime” em s1.

Atividade

- Abra a pasta Downloads no VS Code
  - Utilize o comando git clone https://github.com/ECM404/lista5.git --recursive
  - Entre na pasta utilizando o comando cd lista5
  - Inicialize o diretório com o comando make install