



Programação em Linguagem C

Compilador Online GDB

Estrutura Básica de Programação

Plataforma de Programação

Online GDB

<https://www.onlinegdb.com/>

OnlineGDB beta
online compiler and debugger for c/c++

Welcome, **José WR Pereira**

Create New Project

My Projects

Classroom **new**

Learn Programming

Programming Questions

Logout

f t + 63.6K

Sustain Podcast #35 All About The Drupal Association
ads served ethically

About • FAQ • Blog • Terms of Use • Contact Us
• GDB Tutorial • Credits • Privacy
© 2016 - 2020 GDB Online

main.c

```
1 /*****  
2  
3 Welcome to GDB Online.  
4 GDB online is an online compiler and debugger tool for C, C++, Python, Java, PHP, Ruby, Perl,  
5 C#, VB, Swift, Pascal, Fortran, Haskell, Objective-C, Assembly, HTML, CSS, JS, SQLite, Prolog.  
6 Code, Compile, Run and Debug online from anywhere in world.  
7  
8 *****/  
9 #include <stdio.h>  
10  
11 int main()  
12 {  
13     printf("Hello World");  
14  
15     return 0;  
16 }  
17  
18
```

Language C

Run Debug Stop Share Save Beautify

input

Command line arguments:

Standard Input: ☒ Interactive Console ☐ Text

Implemente uma **estratégia** Red Hat Saiba mais

Plataforma de Programação

Online GDB

<https://www.onlinegdb.com/>

The screenshot shows the OnlineGDB website interface. On the left is a sidebar with navigation links: 'Create New Project', 'My Projects', 'Classroom' (marked 'new'), 'Learn Programming', 'Programming Questions', and 'Logout'. Below these are social media icons for Facebook and Twitter, and a '+ 63.6K' badge. The main content area displays a code editor with a C program. Above the code editor is a toolbar with buttons: 'New File' (document icon), 'Run' (play icon), 'Debug' (bug icon), 'Stop' (stop icon), 'Share' (share icon), 'Save' (floppy disk icon), 'Beautify' (code beautification icon), and 'Download' (download icon). A callout box points to the 'Beautify' button.

Botões de comandos

- New File:** Novo arquivo.
- Run:** Executar o programa.
- Debug:** Depurar o programa.
- Stop:** Parar a execução do prog.
- Share:** Compartilhar o código.
- Save:** Salvar o código.
- Beautify:** 'Embelezar' - Ajustar a indentação do código.
- Download:** Baixar o código.

Plataforma de Programação Online GDB

<https://www.onlinegdb.com/>

OnlineGDB beta
online compiler and debugger for c/c++

Welcome, **José WR Pereira**

Create New Project

My Projects

Classroom **new**

Learn Programming

Programming Questions

Logout

f t + 63.6K

Sustain Podcast #35 All About The Drupal Association
ads served ethically

About • FAQ • Blog • Terms of Use • Contact Us
• GDB Tutorial • Credits • Privacy
© 2016 - 2020 GDB Online

main.c

```
1 /*****  
2  
3 Welcome to GDB Online.  
4 GDB online is an online compiler and debugger tool for C, C++, Python, Java, PHP, Ruby, Perl,  
5 C#, VB, Swift, Pascal, Fortran, Haskell, Objective-C, Assembly, HTML, CSS, JS, SQLite, Prolog.  
6 Code, Compile, Run and Debug online from anywhere in world.  
7  
8 *****/  
9 #include <stdio.h>  
10  
11 int main()  
12 {  
13     printf("Hello World");  
14  
15     return 0;  
16 }  
17  
18
```

Run Debug Stop Share Save Beautify

Language C

Selecionar a Linguagem de compilação: C

input

Command line arguments:

Standard Input: ☒ Interactive Console ☐ Text

Implemente uma **estratégia** Red Hat Saiba mais

Plataforma de Programação

Online GDB

<https://www.onlinegdb.com/>

OnlineGDB beta
online compiler and debugger for c/c++

Welcome, **José WR Pereira**

Create New Project

My Projects

Classroom **new**

Learn Programming

Programming Questions

Logout

f t + 63.6K

Sustain Podcast #35 All About The Drupal Association
ads served ethically

About • FAQ • Blog • Terms of Use • Contact Us
• GDB Tutorial • Credits • Privacy
© 2016 - 2020 GDB Online

main.c

```
1 /*****  
2  
3 Welcome to GDB Online.  
4 GDB online is an online compiler and debugger tool for C, C++, Python, Java, PHP, Ruby, Perl,  
5 C#, VB, Swift, Pascal, Fortran, Haskell, Objective-C, Assembly, HTML, CSS, JS, SQLite, Prolog.  
6 Code, Compile, Run and Debug online from anywhere in world.  
7  
8 *****/  
9 #include <stdio.h>  
10  
11 int main()  
12 {  
13     printf("Hello World");  
14  
15     return 0;  
16 }  
17  
18
```

Run Debug Stop Share Save {} Beautify

Language C

Command line arguments:

Standard Input: ☒ Interactive Console ☐ Text

Implemente uma **estratégia**

Editor do código fonte

Plataforma de Programação

Online GDB

<https://www.onlinegdb.com/>

The screenshot displays the OnlineGDB web interface. On the left is a sidebar with navigation links: 'Create New Project', 'My Projects', 'Classroom' (marked 'new'), 'Learn Programming', 'Programming Questions', and 'Logout'. Below these are social media icons for Facebook and Twitter, and a '+ 63.6K' badge. The main area features a code editor with a file named 'main.c' containing the following C code:

```
1  /*****  
2  
3  Welcome to GDB Online.  
4  GDB online is an online compiler and debugger tool for C, C++, Python, Java, PHP, Ruby, Perl,  
5  C#, VB, Swift, Pascal, Fortran, Haskell, Objective-C, Assembly, HTML, CSS, JS, SQLite, Prolog.  
6  Code, Compile, Run and Debug online from anywhere in world.  
7  
8  *****/  
9  #include <stdio.h>  
10  
11  int main()  
12  {  
13      printf("Hello World");  
14  
15      return 0;  
16  }  
17  
18
```

Below the code editor is a terminal window. A yellow callout box with the text 'Terminal de execução do programa' points to the terminal. The terminal output shows 'Hello World' and '...Program finished with exit code 0'. The interface also includes a top bar with buttons for 'Run', 'Debug', 'Stop', 'Share', 'Save', 'Beautify', and a language dropdown set to 'C'.

Estrutura básica de um programa em linguagem C

```
main.c
1  /*****
2
3  Welcome to GDB Online.
4  GDB online is an online compiler and debugger tool for
5  C, C++, Python, Java, PHP, Ruby, Perl, C#, VB, Swift,
6  Pascal, Fortran, Haskell, Objective-C, Assembly, HTML,
7  CSS, JS, SQLite, Prolog.
8
9  *****/
10
11 // Code, Compile, Run and Debug online from anywhere in world.
12
13
14
15 #include <stdio.h>
16
17
18 int main()
19 {
20     printf("Hello World");
21
22     return 0;
23 }
24
```

Comentário em bloco

Comentário de linha

Inclusão de bibliotecas

Programa principal

Estilos de código

```
main.c
1 #include<stdio.h>
2 int
3 main
4 (
5 )
6 {
7 printf
8 (
9 "Hello World"
10 )
11 ;
12 return
13 0;
14 }
15
```

```
main.c
1 #include<stdio.h>
2
3 int main(){
4     printf("Hello World");
5     return 0;
6 }
7
```


```
main.c
1 #include<stdio.h>
2 int main(){ printf("Hello World");return 0;}
3
4
```

```
main.c
1 #include <stdio.h>
2
3 int
4 main ()
5 {
6     printf ("Hello World");
7
8     return 0;
9 }
10
```


```
main.c
1 #include <stdio.h>
2
3 int main()
4 {
5     printf("Hello World");
6
7     return 0;
8 }
9
```


Estilos de código


```
main.c
1 #include<stdio.h>
2 int
3 main
4 {
5     printf("Hello World");
6 }
7
8
9
10
11
12 return
13 0;
14 }
15
```



```
main.c
1 #include<stdio.h>
2
3 int main(){
4     printf("Hello World");
5     return 0;
6 }
7
```




```
main.c
1 #include <stdio.h>
2
3 int
4 main ()
5 {
6     printf ("Hello World");
7
8     return 0;
9 }
10
```



```
main.c
1 #include<stdio.h>
2 int main(){printf("Hello World");return 0;}
3
4
```



```
main.c
1 #include <stdio.h>
2
3 int main()
4 {
5     printf("Hello World");
6
7     return 0;
8 }
9
```



Biblioteca padrão

Entradas e Saídas

main.c

```
1 #include <stdio.h>
2
3 int main()
4 {
5     printf("Hello World");
6
7     return 0;
8 }
9
10
```

Inclusão de biblioteca **stdio.h**

(Standard Input Output)

Possui funções de manipulação da entrada e da saída padrão.

Exemplo:

```
printf()
scanf()
```

Estrutura básica função main()

main.c

```
1 #include <stdio.h>
2
3 int main()●
4 {
5     printf("Hello World");
6
7     return 0;
8 }
9
10
```

Programa principal (main)

Todo programa em linguagem C é estruturado em uma ou mais

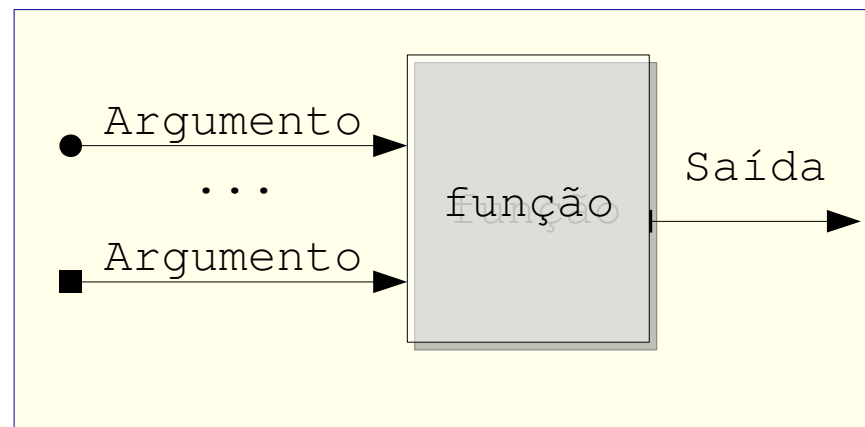
Funções.

O nome das funções é de escolha do programador, exceto a única função obrigatória, **main**.

Estrutura básica de uma função

main.c

```
1  #include <stdio.h>
2
3  int main()
4  {
5      printf("Hello World");
6
7      return 0;
8  }
9
10
```



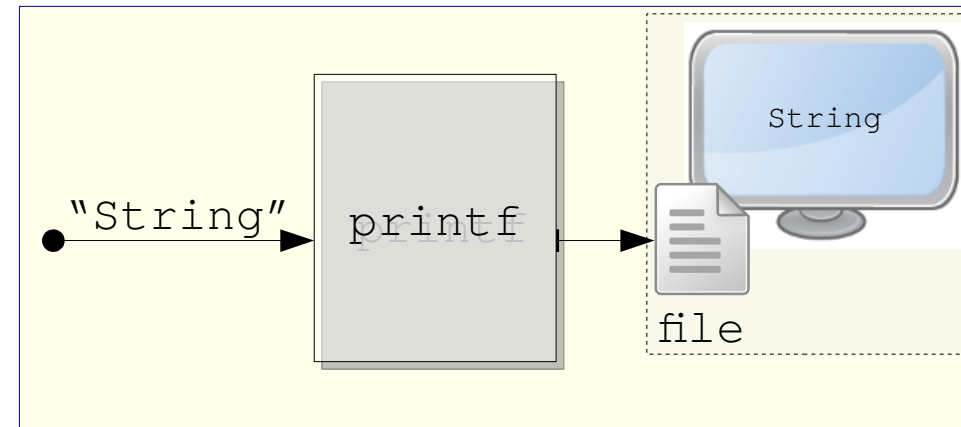
Estrutura de uma função

```
<tipo_saída> nome_da_funcao(<tipo_entrada> argumento,...)
{ início do corpo da função
  corpo da função com retorno de dado
} fim do corpo da função
```

Função printf()

source code

```
1 #include <stdio.h>
2
3 int main()
4 {
5     • printf("String");
6     return 0;
7 }
8
```



printf

Imprime uma cadeia de caracteres no arquivo de saída padrão.
A saída padrão, geralmente, é o monitor.

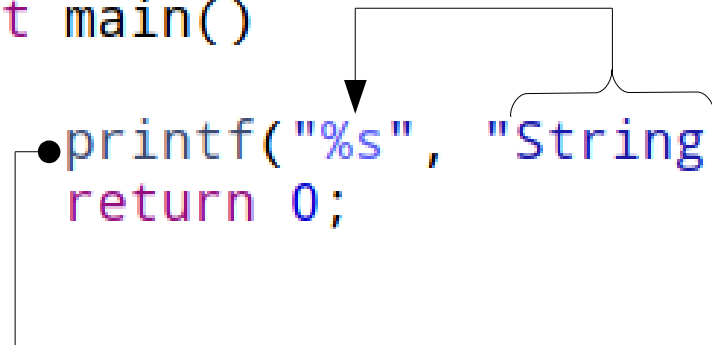
Memória

...	'S'	't'	'r'	'i'	'n'	'g'	\0	...
-----	-----	-----	-----	-----	-----	-----	----	-----

Função printf()

source code

```
1 #include <stdio.h>
2
3 int main()
4 {
5     printf("%s", "String");
6     return 0;
7 }
8
```



The diagram shows a bracket in the source code grouping the arguments of the printf call: the format string "%s" and the string "String". An arrow points from this bracket to the printf function definition box below. Another arrow points from the printf call line (line 5) to the same box.

Saída

String

printf

Imprime uma cadeia de caracteres no arquivo de saída padrão.
A saída padrão, geralmente, é o monitor.

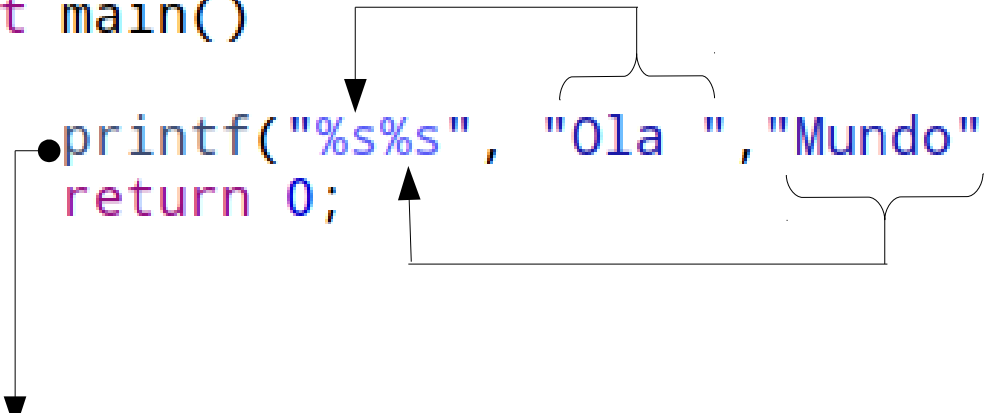
Memória

...	'S'	't'	'r'	'i'	'n'	'g'	\0	...
-----	-----	-----	-----	-----	-----	-----	----	-----

Função printf()

source code

```
1 #include <stdio.h>
2
3 int main()
4 {
5     printf("%s%s", "Ola ", "Mundo");
6     return 0;
7 }
8
```



Saída

Ola Mundo

Formato de impressão

%s: string - cadeia de caracteres

8 bits

1 byte

...	'O'	'l'	'a'	' '	'\0'	...	'M'	'u'	'n'	'd'	'o'	'\0'	...
-----	-----	-----	-----	-----	------	-----	-----	-----	-----	-----	-----	------	-----



Função printf()

main.c

```
1 #include <stdio.h>
2
3 int main()
4 {
5     int num = 0;
6     printf("Numero: %d\n", num );
7     return 0;
8 }
9
```

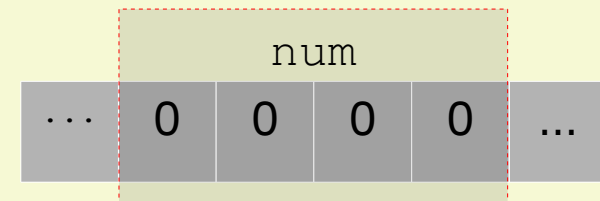
Formato de impressão

%d: número inteiro - 4 bytes

Saída

Numero: 0

Segmento de memória



Função printf()

```
main.c
1  #include <stdio.h>
2
3  int main()
4  {
5      int num = 0;
6      printf("Numero: %d\n", num );
7      printf("Numero: %3d\n", num );
8      return 0;
9  }
10
```

Formato de impressão

%d: número inteiro - 4 bytes

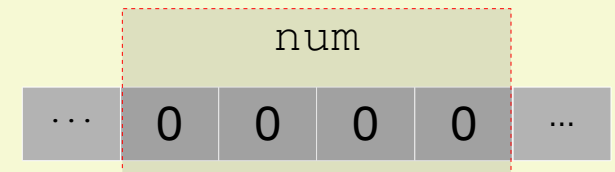
%nd: número de espaços reservados

Saída

Numero: 0

Numero: 0

Segmento da memória



Função printf()

```
main.c
1 #include <stdio.h>
2
3 int main()
4 {
5     int num = 74;
6     printf("Numero: %d\n", num );
7     printf("Numero: %3d\n", num );
8     return 0;
9 }
10
```

Formato de impressão

%d: número inteiro - 4 bytes

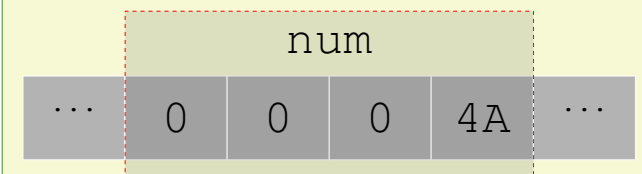
%nd: número de espaços reservados

Saída

Numero: 74

Numero: 74

Segmento da memória



74 = 0b01001010 = 0x4A

Função printf()

```
main.c
1  #include <stdio.h>
2
3  int main()
4  {
5      int num = 74;
6      printf("Numero: %d\n", num );
7      printf("Caractere: %c\n", num );
8      return 0;
9  }
10
```

Formato de impressão

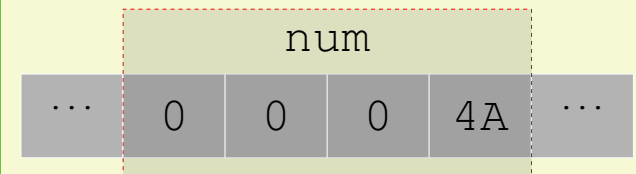
%c: caractere - símbolo gráfico convencional para:

- Dígitos Numéricos ('0', '1', '2', ...)
- Letras ('A', 'B', 'C', ... 'a', 'b', 'c', ...)
- Código de controle (\n, \t, ...)
- Símbolo especial

Saída

Numero: 74
Caractere: J

Segmento da memória



74 = 0b01001010 = 0x4A

Função scanf()

```
main.c
1  #include <stdio.h>
2
3  int main()
4  {
5      int num = 0;
6      printf("Numero: %d\n", num );
7      scanf("%d", &num );
8      printf("Numero: %d\n", num );
9      return 0;
10 }
11
```

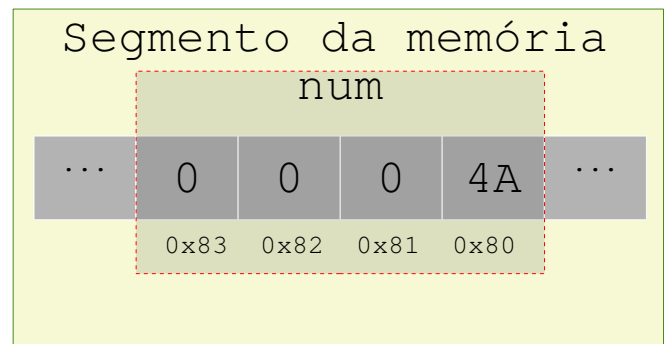
scanf()

Lê a entrada padrão, que geralmente, é o teclado

%d: tipo de dado de entrada: inteiro

&<var>: endereço da variável que receberá o dado lido.

Saída
Numero: 0
Entrada
74<Enter>
Saída
Numero: 74



74 = 0b01001010 = 0x4A

Função scanf()

main.c

```
1 #include <stdio.h>
2
3 int main()
4 {
5     char digito;
6     printf("Digite: \n");
7     scanf("%c", &digito);
8     printf("%c\n", digito);
9     return 0;
10 }
11
```

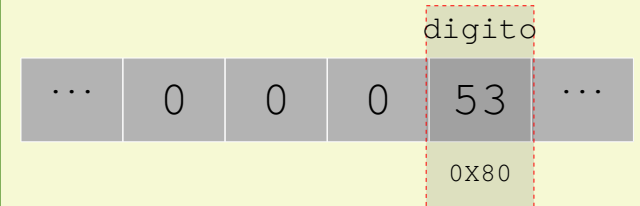
scanf()

%c: tipo de dado de entrada: caractere

Saída

Digite:
S<Enter>
S

Segmento da memória



83 = 0b01010011 = 0x53

Função scanf()

main.c

```
1 #include <stdio.h>
2
3 int main()
4 {
5     char digito;
6     printf("Digite: \n");
7     scanf("%c", &digito);
8     printf("%c\n", digito);
9     return 0;
10 }
11
```

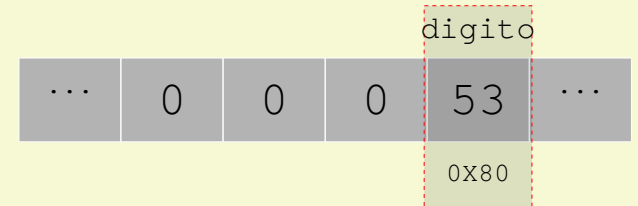
scanf()

%c: tipo de dado de entrada: caractere

Saída

Digite:
SENAI<Enter>
S

Segmento da memória



83 = 0b01010011 = 0x53

Função scanf()

main.c

```
1 #include <stdio.h>
2
3 int main()
4 {
5     char palavra[6];
6     printf("Digite: \n");
7     scanf("%s", palavra );
8     printf("%s\n", palavra );
9     return 0;
10 }
11
```

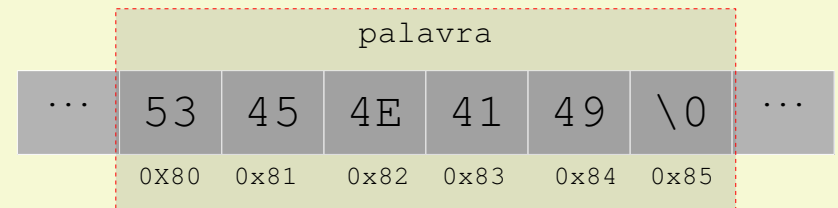
Saída

Digite:
SENAI<Enter>
SENAI

scanf()

%s:String - cadeia de caracteres

Segmento da memória



Função scanf()

main.c

```
1 #include <stdio.h>
2
3 int main()
4 {
5     char palavra[6];
6     printf("Digite: \n");
7     scanf("%s", palavra );
8     printf("%s\n", palavra );
9     return 0;
10 }
11
```

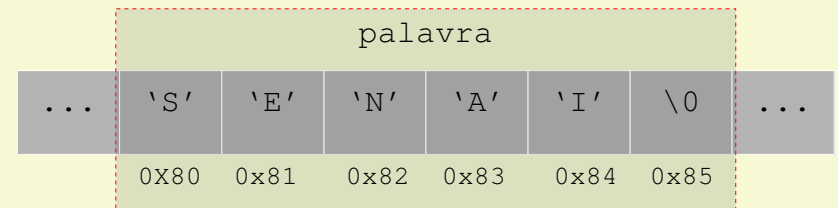
Saída

Digite:
SENAI<Enter>
SENAI

scanf()

%s:String - cadeia de caracteres

Segmento da memória

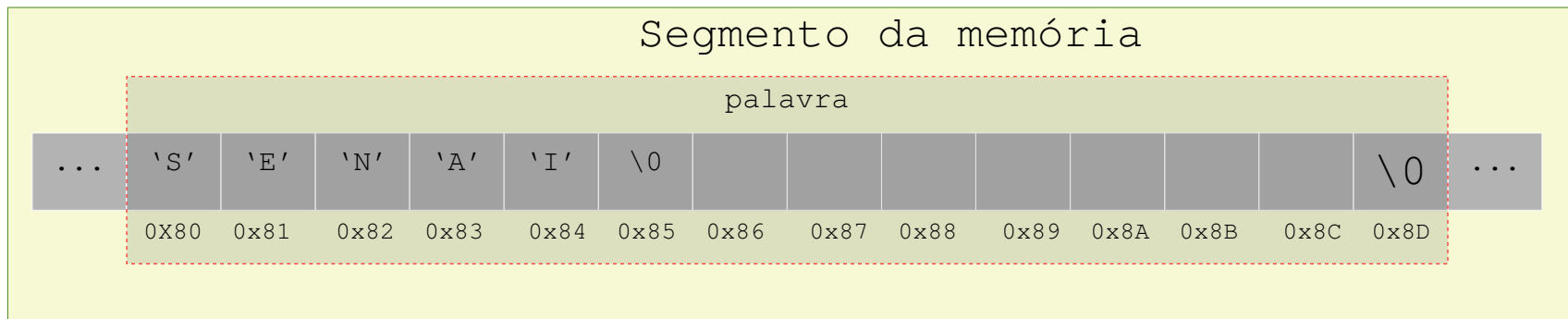


Função scanf()

```
main.c
1  #include <stdio.h>
2
3  int main()
4  {
5      char escola[14];
6      printf("Escola: \n");
7      scanf("%s", escola );
8      printf("%s\n", escola );
9      return 0;
10 }
11
```

Saída

```
Escola:
SENAI Jandira<Enter>
SENAI
```



Função scanf()

main.c

```
1 #include <stdio.h>
2
3 int main()
4 {
5     char escola[14];
6     printf("Escola: \n");
7     scanf("%13[^\n]", escola );
8     printf("%s\n", escola );
9     return 0;
10 }
11
```

Saída

```
Escola:
SENAI Jandira<Enter>
SENAI Jandira
```

Segmento da memória

palavra

...	'S'	'E'	'N'	'A'	'I'	' '	'J'	'a'	'n'	'd'	'i'	'r'	'a'	'\0'	...
	0x80	0x81	0x82	0x83	0x84	0x85	0x86	0x87	0x88	0x89	0x8A	0x8B	0x8C	0x8D	



Site: josewrpereira.github.io/ddp/
E-mail: josewrpereira@outlook.com