INSTITUTO FEDERAL
Paraná
Campus Paranaguá

Ministério da Educação

ENSINO MÉDIO INTEGRADO - INFORMÁTICA
Disciplina de Linguagem de Programação
Aula 01: Introdução a Linguagem C

Gil Eduardo de Andrade

Conceitos Preliminares

(https://docs.microsoft.com/pt-br/cpp/c-language/?view=msvc-170)

Introdução

A linguagem C é uma linguagem estruturada, desenvolvida na década de 70, e que permitiu a criação do sistema UNIX. A linguagem C pode ser considerada uma linguagem mãe para o surgimento de outras tecnologias, visto que serviu como base para criação de novas linguagens, tais como C++ e Java.

Nota: a programação estruturada é uma técnica que tem como objetivo a construção de programas claros, legíveis, eficientes e de fácil manutenção, sendo baseada em três estruturas de controle para a construção da lógica do programa: ***a sequência***, ***a seleção*** e ***a repetição***.

Principais Características

A linguagem C possui uma estrutura de codificação relativa simples, com geração de códigos executáveis pequenos e rápidos, via processo de compilação. Além disso, a linguagem C é portável, ou seja, programas escritos em C para uma plataforma (Linux) podem ser facilmente adaptados para outra (Windows). Por fim, é considerada uma linguagem de nível relativamente baixo, possuindo elementos Assembly. Por se tratar de uma linguagem estruturada, a linguagem C permite que a codificação seja dividida em módulos, que o programador possa definir e incluir várias rotinas e que o mesmo construa suas bibliotecas de modo customizado.

Ferramentas de Programação Online

- Online GDB
  https://www.onlinegdb.com/online_c_compiler

- Replit
  https://replit.com/languages/c

Interfaces de Desenvolvimento (Windows)

- DevC++
  https://sourceforge.net/projects/orwelldevcpp/

- CodeBlocks
  https://sourceforge.net/projects/codeblocks/files/Binaries/20.03/Windows/codeblocks-20.03-setup.exe/download

- Visual Studio
  https://visualstudio.microsoft.com/pt-br/thank-you-downloading-visual-studio/?sku=Community&channel=Release&version=VS2022&source=VSLandingPage&cid=2030&passive=false

Conceitos Iniciais

- Bibliotecas do Sistema: possuem um conjunto específico de funções, já implementadas, e que podem ser utilizadas pelo programador. A biblioteca (standard in/out) é um exemplo, possui funções que permitem utilizar a entrada (teclado) e a saída (monitor) padrão do computador.

- Função main(): todo arquivo **“.c”** deve, obrigatoriamente, possuir uma função **“main()”**. A execução do código se inicia nela. As chaves **“{“** e **“}”** servem para indicar onde a codificação da função **“main()”** inicia e acaba, respectivamente.

- Variáveis: são espaços reservados na memória RAM, para que valores possam ser armazenados e resgatados posteriormente. A linguagem C

possui dois tipos de variáveis, locais (nesta aula) e globais (futuramente), e é dita fortemente tipada, pelo fato de suas variáveis possuírem tipos (int, char, float).

- Tipos Básicos:
  - **int** → armazena um valor inteiro, exemplo: **0, 2, -56, 200**;
  - **char** → armazena um caractere, exemplo: **a, z, D, X, q**;
  - **float** → armazena um valor com vírgula (ponto flutuante), exemplo: **3.14, -12.6, 0, 100.56**;

- Declaração de Variáveis: a declaração das variáveis em C deve ser feita no início do código, logo abaixo da função main(). Todas as variáveis que serão utilizadas ao longo da codificação devem, obrigatoriamente, ser declaradas. No momento da declaração também é obrigatório especificar o tipo de cada variável (int, char), assim como o seu nome. O nome das variáveis não podem conter espaço, acento, cedilha ou começar com um número.

- Final de Linha: na linguagem C é necessário indicar ao seu compilador o término de uma linha de comando, para tal existe o caractere reservado **";"**. Sendo assim, após declarar uma variável, por exemplo, utiliza-se o ponto-vírgula;

- Atribuição de Valores "=": como descrito, as variáveis são utilizadas para armazenar valores, e para tal foi definido um operador, o caractere reservado **"="**. Exemplo: **x = 12**;

- Operadores Matemáticos:
  - **"+"**: permite efetuar a soma de valores, sejam eles estáticos ou armazenados em variáveis. Exemplo: **soma = x + y**;
  - **"-"**: permite efetuar a subtração de valores, sejam eles estáticos ou armazenados em variáveis. Exemplo: **sub = x + y**;

- **“*”**: permite efetuar a multiplicação de valores, sejam eles estáticos ou armazenados em variáveis. Exemplo: **mult = x * y**;
- **“/”**: permite efetuar a divisão de valores, sejam eles estáticos ou armazenados em variáveis. Exemplo: **div = x / y**;
- **“%”**: permite obter o resto da divisão entre dois valores, que devem ser obrigatoriamente inteiros, sejam eles estáticos ou armazenados em variáveis. Exemplo: **resto = x % y**;

Comando de Saída

Os comandos de saída permitem que informações sejam enviadas para dispositivos computacionais (monitores, impressoras, etc). O comando de saída mais importante na linguagem C é o **printf()**, disponível na biblioteca **<stdio.h>**.
O comando **printf()** permite que textos e valores contidos em variáveis sejam apresentados ao usuário. O texto que deseja-se apresentar deve ser escrito entre aspas duplas, dentro dos parênteses do comando.
Exemplo: **printf(“Gil Eduardo de Andrade”);**

- **Caracteres Especiais:**
  - **“\n”:** permite que quebras de linha sejam efetuadas na utilização do comando **printf()**. Exemplo: **printf(“Gil\nEduardo\nAndrade\n”);**

  - **“\t”:** permite que tabulações sejam criadas na utilização do comando **printf()**. Exemplo: **printf(“\tGil\n\t\tEduardo\n\t\t\tAndrade”);**

- **Exibição de Conteúdo das Variáveis:**
  - **“%i” ou “%d”:** permite exibir o conteúdo das variáveis do tipo **int** na utilização do **printf()**. Exemplo: **printf(“%i”, soma);**

  - **“%c”:** permite exibir o conteúdo das variáveis do tipo **char** na utilização do **printf()**. Exemplo: **printf(“%c”, letra);**

- **“%f”:** permite exibir o conteúdo das variáveis do tipo **float** na utilização do **printf()**. Exemplo: **printf(“%f”, pi);**

Codificação – Linguagem de Programação C

Estrutura básica de um programa em C

```c
#include <stdio.h>

int main() {

   return 0;

}
```

```
                        Estrutura Básica - Linguagem C
          ____________________________________________________________
      [#include] - utilizado para incluir, no programa, bibliotecas do sistema
   [<stdio.h>] - biblioteca que contém funções que permitem acesso a entrada e saída
                        padrão do sistema (teclado e monitor)
          [main()] - função principal do programa, indica onde ele inicia
                   [return 0] - finaliza a execução do programa
          ____________________________________________________________
                  (Arquivo-fonte: 01 - Introdução C/referencia.c)
```

Declaração de Variáveis / Atribuição de Valores

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

```c
int main() {

   int a1, a2;

   float pi;

   char c;

   a1 = 10;

   pi = 3.1415;

   c = 'c';

   return 0;

}
```

Declaração de Variáveis / Atribuição de Valores

[int a1, a2] - declara duas variáveis do tipo “int” com nomes “a1” e “a2”

[float pi] - declara uma variável do tipo “float” com nome “pi”

[char c] - declara uma variável do tipo “char” com nome “c”

[a1 = 10] - atribui/armazena o valor “10” na variável “a”

[pi = 3.1415] - atribui/armazena o valor “3.1415” na variável “pi”

[c = ‘c’] - atribui/armazena o valor “c” na variável “c”

(Arquivo-fonte: 01 - Introdução C/variavel.c)

Operadores Matemáticos

```c
#include <stdio.h>

int main() {

   int v1, v2, soma, mult, div, sub, resto;

   v1 = 10;

   v2 = 20;

   soma = v1 + v2;
```

```
   mult = v1 * v2;

   div = v2 / v1;

   sub = v2 - v1;

   v1 = 5;

   v2 = 2;

   resto = v1 % v2;

   . . .

   return 0;

}
```

Operadores Matemáticos

[soma = v1+v2] - soma o valores das variáveis “v1” e “v2” e armazena em “soma”

[mult = v1*v2] - multiplica o valor da variável “v1” por “v2” e armazena em “mult”

[div = v2/v1] - divide o valor da variável “v2” por “v1” e armazena em “div”

[sub = v2-v1] - subtrai o valor da variável “v2” por “v1” e armazena em “soma”

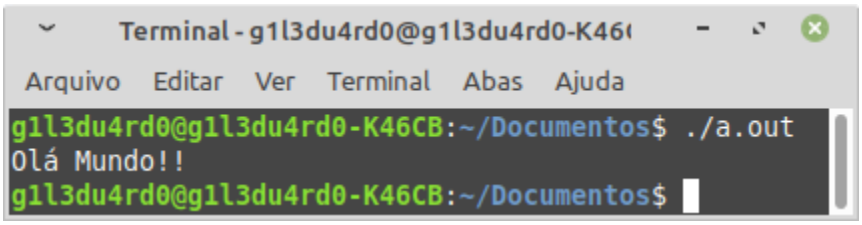
[resto = v1%v2] - calcula o resto da divisão da variável “v1” por “v2” e armazena em “resto”

(Arquivo-fonte: 01 - Introdução C/variavel_int.c)

Comando de Saída - printf(“”);

```
#include <stdio.h>

int main() {

   printf("Olá Mundo!!");

   return 0;

}
```

```
Comando de Saída - printf("");

_____________________________________________________________

[printf("Olá Mundo");] - exibe o texto "Olá Mundo" no monitor/terminal

_____________________________________________________________

(Arquivo-fonte: 01 - Introdução C/printf.c)
```

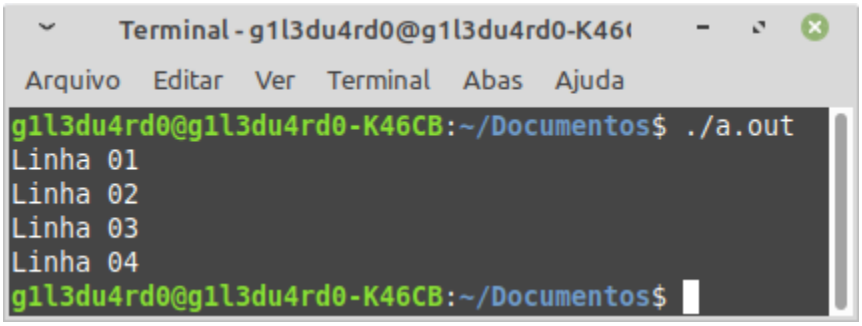


Resultado da execução do código anterior

Comando de Saída - Caracteres Especiais "\n"

```c
#include <stdio.h>

int main() {

   printf("Linha 01\nLinha 02\nLinha 03\nLinha 04");

   return 0;

}
```

```
Comando de Saída - "\n"

_____________________________________________________________

[printf("\n");] - permite quebra linha dentro do texto inserido no printf();

_____________________________________________________________

(Arquivo-fonte: 01 - Introdução C/printf_barran.c)
```



Resultado da execução do código anterior

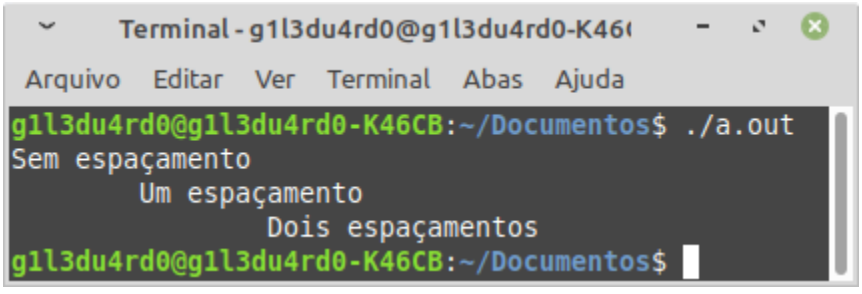
Comando de Saída - Caracteres Especiais “\t”

```c
#include <stdio.h>

int main() {

   printf("Sem espaçamento\n\tUm espaçamento\n\t\tDois espaçamentos");

   return 0;

}
```

Comando de Saída - “\t”

[printf(“\n”);] - permite adicionar tabulações dentro do texto inserido no printf();

(Arquivo-fonte: 01 - Introdução C/printf_barrat.c)



Resultado da execução do código anterior

Comando de Saída - Exibindo o Conteúdo das Variáveis

```c
#include <stdio.h>

int main() {

   int a = 10;

   float b = 3.1415;

   char c = 'G';

   printf("Valor da variável a: %i (int)", a);

   printf("\nValor da variável b: %f (float)", b);

   printf("\nValor da variável c: %c (char)", c);

   return 0;

}
```

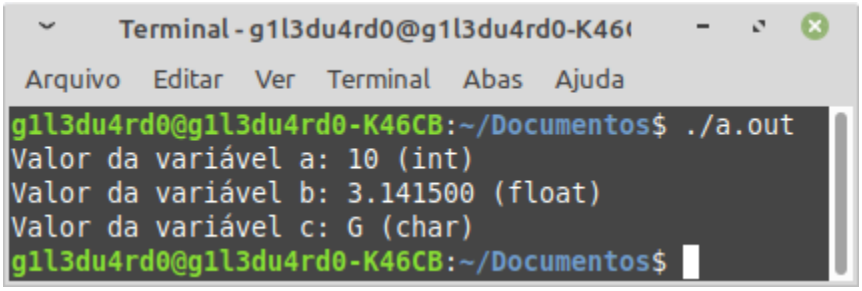
Comando de Saída - “\t”

[printf(“%i”, a);] - exibe o conteúdo da variável “a” do tipo “int”

[printf(“%f”, b);] - exibe o conteúdo da variável “b” do tipo “float”

[printf(“%c”, c);] - exibe o conteúdo da variável “c” do tipo “char”

(Arquivo-fonte: 01 - Introdução C/printf_barrat.c)



Resultado da execução do código anterior

COMANDOS DE ENTRADA E SAÍDA

Conceitos Preliminares

(https://docs.microsoft.com/pt-br/cpp/c-language/?view=msvc-170)

Introdução

Os comandos de entrada e saída permitem ao programador interagir com o usuário, possibilitando que informações sejam enviadas e recebidas do usuário.

Comando de Saída

O principal e mais utilizado comando de saída, na linguagem C, é o **printf()** - visto na aula anterior. O **printf()** permite o envio de textos estáticos e também o valor de variáveis ao monitor do computador (saída padrão). Exemplo: **printf("Gil Eduardo de Andrade");**

- Caracteres Especiais:
  - **"\n":** permite que quebras de linha sejam efetuadas na utilização do comando **printf()**. Exemplo: **printf("Gil\nEduardo\nAndrade\n");**
  - **"\t":** permite que tabulações sejam criadas na utilização do comando **printf()**. Exemplo: **printf("\tGil\n\t\tEduardo\n\t\t\tAndrade");**
- Exibição de Conteúdo das Variáveis:
  - **"%i" ou "%d":** permite exibir o conteúdo das variáveis do tipo **int** na utilização do **printf()**. Exemplo: **printf("%i", soma);**
  - **"%c":** permite exibir o conteúdo das variáveis do tipo **char** na utilização do **printf()**. Exemplo: **printf("%c", letra);**

- **“%f”:** permite exibir o conteúdo das variáveis do tipo **float** na utilização do **printf()**. Exemplo: **printf(“%f”, pi);**

Comando de Entrada

os comandos de entrada permitem que informações especificadas pelo usuário sejam recebidas pelo programa codificado. O principal e mais utilizado comando de entrada, na linguagem C, é o **scanf()**. O **scanf()** permite ao programador ler informações fornecidas pelo usuário em tempo de execução, através do teclado (entrada padrão). Exemplo: **scanf(“%i”, &valor);**

- **“%i”:** para obter um valor inteiro;
- **“%c”:** para obter um caractere;
- **“%f”:** para obter um valor float;
- **“%d”:** para obter um valor decimal.

Lendo vários valores com apenas um scanf()

Também é possível efetuar a leitura de vários valores através da utilização de apenas um comando **scanf()**. Os múltiplos valores de entrada podem ou não ser do mesmo tipo. Exemplo: **scanf(“%i %c %f”, &inteiro, &letra, &flutuante);**

Comando getchar()

O comando **getchar()** permite efetuar a leitura de caracteres de forma análoga ao comando scanf(“%c”). Entretanto, sua sintaxe é mais simples, pois é preciso apenas atribuí-lo a variável que receberá o caractere digitado. Exemplo: **letra = getchar();**

Codificação – Linguagem de Programação C

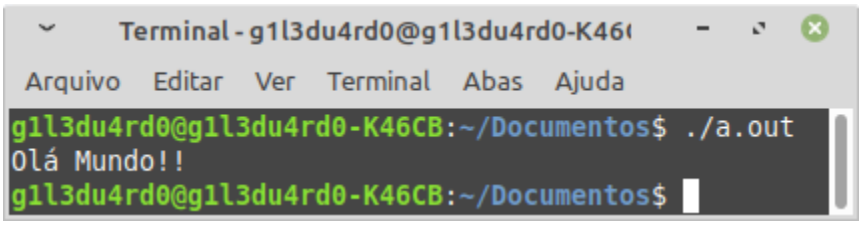
Comando de Saída - printf(“”);

```c
#include <stdio.h>

int main() {

   printf("Olá Mundo!!");

   return 0;

}
```

Comando de Saída - printf(“”);

[printf(“Olá Mundo”);] - exibe o texto “Olá Mundo” no monitor/terminal

(Arquivo-fonte: 01 - Introdução C/printf.c)



Resultado da execução do código anterior

Comando de Saída - Caracteres Especiais “\n”

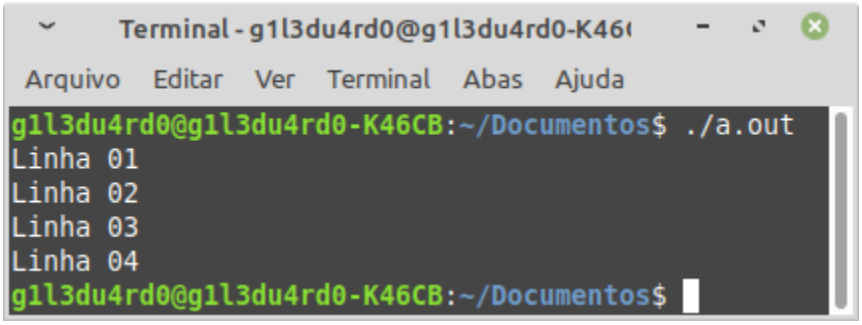
```c
#include <stdio.h>
```

```c
int main() {

   printf("Linha 01\nLinha 02\nLinha 03\nLinha 04");

   return 0;

}
```

Comando de Saída - “\n”

[printf(“\n”);] - permite quebra linha dentro do texto inserido no printf();

(Arquivo-fonte: 01 - Introdução C/printf_barran.c)



Resultado da execução do código anterior

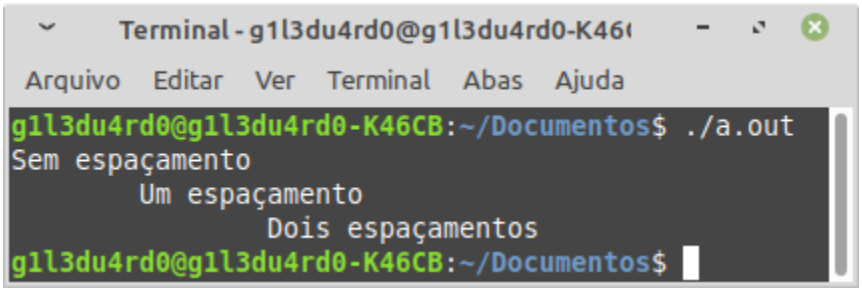
Comando de Saída - Caracteres Especiais “\t”

```c
#include <stdio.h>

int main() {

   printf("Sem espaçamento\n\tUm espaçamento\n\t\tDois espaçamentos");

   return 0;

}
```

Comando de Saída - “\t”

[printf(“\n”);] - permite adicionar tabulações dentro do texto inserido no printf();

(Arquivo-fonte: 01 - Introdução C/printf_barrat.c)



Resultado da execução do código anterior

Comando de Saída - Exibindo o Conteúdo das Variáveis

```c
#include <stdio.h>

int main() {

   int a = 10;

   float b = 3.1415;

   char c = 'G';

   printf("Valor da variável a: %i (int)", a);

   printf("\nValor da variável b: %f (float)", b);

   printf("\nValor da variável c: %c (char)", c);

   return 0;

}
```

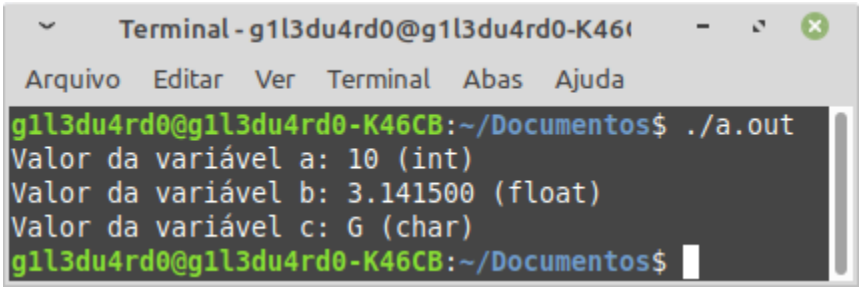
Comando de Saída - “\t”

[printf(“%i”, a);] - exibe o conteúdo da variável “a” do tipo “int”

[printf(“%f”, b);] - exibe o conteúdo da variável “b” do tipo “float”

[printf(“%c”, c);] - exibe o conteúdo da variável “c” do tipo “char”

```
(Arquivo-fonte: 01 - Introdução C/printf_barrat.c)
```



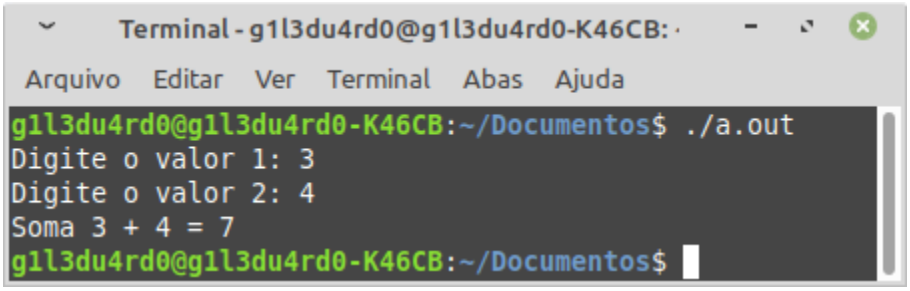
Resultado da execução do código anterior

Comando de Entrada “scanf()” - Inteiros

```c
#include <stdio.h>

int main() {

   int v1, v2, soma;

   printf("Digite o valor 1: ");

   scanf("%i", &v1);

   printf("Digite o valor 2: ");

   scanf("%i", &v2);

   soma = v1 + v2;

   printf("Soma %i + %i = %i", v1, v2, soma);

   printf("\n");

   return 0;

}
```

```
                    Comando de Entrada - “scanf(“%i”)”
          ____________________________________________________________
    [scanf(“%i”, &v1);] - obtém o valor inteiro digitado e armazena na variável v1
    [scanf(“%i”, &v2);] - obtém o valor inteiro digitado e armazena na variável v2
```

(Arquivo-fonte: 02 - Entrada e Saída/es_scanf_int.c)



Resultado da execução do código anterior

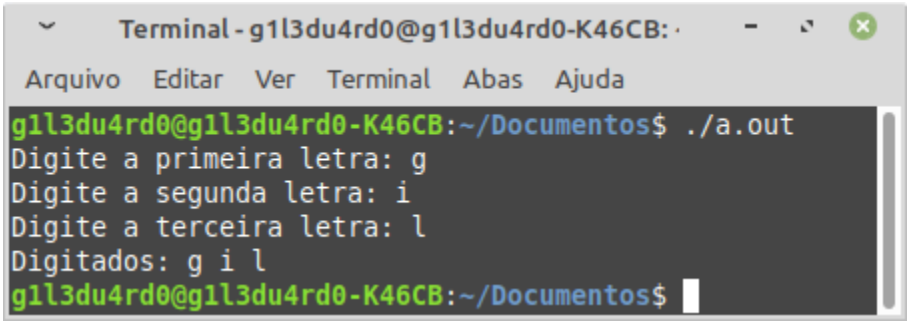
Comando de Entrada “scanf()” - Caracteres / Espaço

```c
#include <stdio.h>

int main() {

   char c1, c2, c3;

   printf("Digite a primeira letra: ");

   scanf(" %c", &c1);

   printf("Digite a segunda letra: ");

   scanf(" %c", &c2);

   printf("Digite a terceira letra: ");

   scanf(" %c", &c3);

   printf("Digitados: %c %c %c", c1, c2, c3);

   printf("\n");

   return 0;

}
```

Comando de Entrada - “scanf(“%c”)”

[scanf(“%c”, &c1);] - obtém o caractere digitado e armazena na variável c1

```
[scanf("%c", &c2);] - obtém o caractere digitado e armazena na variável c2
[scanf("%c", &c3);] - obtém o caractere digitado e armazena na variável c2

(Arquivo-fonte: 02 - Entrada e Saída/es_scanf_char_espaco.c)
```



Resultado da execução do código anterior

Comando de Entrada "scanf()" - Caracteres / __fpurge(stdin)

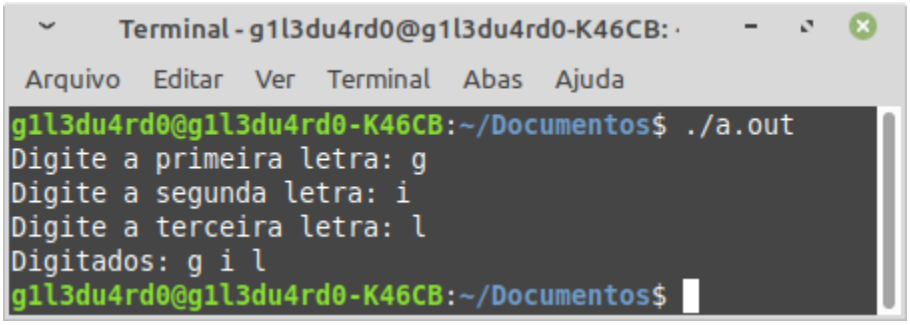
```c
#include <stdio.h>

#include <stdio_ext.h>

int main() {

   char c1, c2, c3;

   printf("Digite a primeira letra: ");

   __fpurge(stdin); // fflush(stdin);

   scanf("%c", &c1);

   printf("Digite a segunda letra: ");

   __fpurge(stdin);

   scanf("%c", &c2);

   printf("Digite a terceira letra: ");

   __fpurge(stdin);

   scanf("%c", &c3);
```

```
   printf("Digitados: %c %c %c", c1, c2, c3);

   printf("\n");

   return 0;

}
```

Comando de Entrada - “scanf(“%c”)” + “__fpurge(stdin)”

[__fpurge(stdin);] - limpa o buffer e garante a execução adequada do próximo “scanf()”

(Arquivo-fonte: 02 - Entrada e Saída/es_scanf_char_fpurge.c)



Resultado da execução do código anterior

Comando de Entrada “scanf()” - Caracteres / Juntos

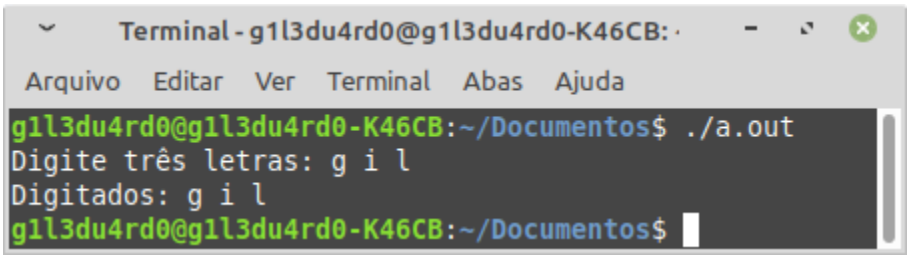
```
#include <stdio.h>

int main() {

   char c1, c2, c3;

   printf("Digite três letras: ");

   scanf("%c %c %c", &c1, &c2, &c3);

   printf("Digitados: %c %c %c", c1, c2, c3);

   printf("\n");
```

```
    return 0;

}
```

Comando de Entrada - “scanf(“%c %c %c”)”

[scanf(“%c %c %c”, &c1, &c2, &c3);] - obtém os caracteres digitados e armazena nas variáveis c1, c2 e c3

(Arquivo-fonte: 02 - Entrada e Saída/es_scanf_char_junto.c)

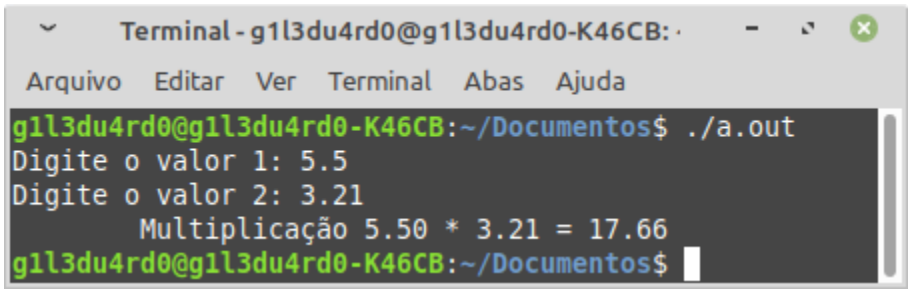


Resultado da execução do código anterior

Comando de Entrada “scanf()” - Ponto Flutuante

```c
#include <stdio.h>

int main() {

   float v1, v2, mult;

   printf("Digite o valor 1: ");

   scanf("%f", &v1);

   printf("Digite o valor 2: ");

   scanf("%f", &v2);

   mult = v1 * v2;

   printf("\tMultiplicação %.2f * %.2f = %.2f", v1, v2, mult);

   printf("\n");
```

```
    return 0;

}

                    Comando de Entrada - “scanf(“%f”)”

        ____________________________________________________________

    [scanf(“%f”, &v1);] - obtém o valor digitado e armazena na variável v1

    [scanf(“%f”, &v2);] - obtém o valor digitado e armazena na variável v2

        ____________________________________________________________

            (Arquivo-fonte: 02 - Entrada e Saída/es_scanf_float.c)
```



Resultado da execução do código anterior

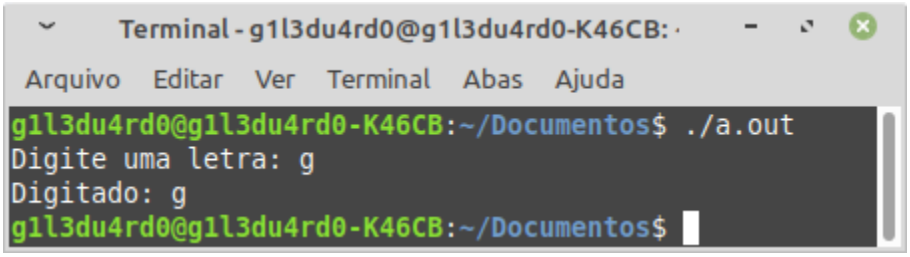
Comando de Entrada “getchar()” - Variável

```c
#include <stdio.h>

int main() {

   char c1;

   printf("Digite uma letra: ");

   c1 = getchar();

   printf("Digitado: %c", c1);

   printf("\n");

   return 0;

}
```

Comando de Entrada - “getchar()” / Variável

[c1 = getchar();] - obtém o caractere digitado e armazena na variável c1

(Arquivo-fonte: 02 - Entrada e Saída/es_getchar_variavel.c)



Resultado da execução do código anterior

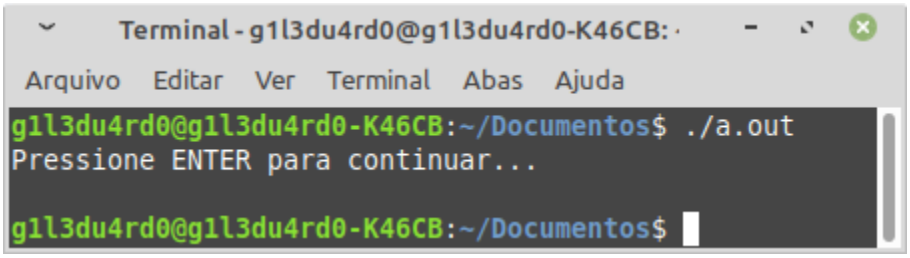
Comando de Entrada “getchar()” - Enter

```c
#include <stdio.h>

int main() {

   printf("Pressione ENTER para continuar...");

   getchar();

   printf("\n");

   return 0;

}
```

Comando de Entrada - “getchar()” / Enter

[getchar();] - Aguarda a tecla ENTER ser pressionada

(Arquivo-fonte: 02 - Entrada e Saída/es_getchar_enter.c)



Resultado da execução do código anterior