ENTRADA E SAÍDA

Objetivos

- Entrada e saída formatada
- Uso de arquivos
  - Escrita e leitura de blocos de dados
  - Movimentação em arquivos

I/O em C

- Formalmente, rotinas de entrada e saída não fazem parte da linguagem, e sim de bibliotecas que acompanham os compiladores
  - Felizmente, são padronizadas
  - Exige-se a linha #include <stdio.h> para usá-las

I/O em C

- `int` **`printf`**`(const char *format, ...);`
  - Retorna o número de caracteres impressos
  - O string contém uma "máscara" (template) com lacunas reservadas para os valores que serão impressos
  - Pode não existir lacuna

```
printf("O valor de x eh %d", x);
printf("Area: %f\n", PI*d*d/4);
printf("Nome: %s", nomeAluno);
```

printf

- Especificadores de formato
  - %c (char)
  - %s (string)
  - %d (int)
  - %ld (long int)
  - %f (float)
  - %lf (double)
  - %e (float notação científica)
  - %g (e ou f, ou seja, notação científica se necessário)

printf

- %[flags][width][.precision]

format	output
printf("%08X",32 bit var);	00ABCD
printf("%d",16 bit var);	4567
printf("%'d",32 bit var);	43,981
printf("%10s","string");	string
printf("%*s",10,"string");	string
printf("%.3s","string");	str
printf("%-10.10s","truncate looong");	truncate

Caracteres de escape

- Acrescentados à máscara para provocar reposicionamento do cursor
  - \n: nova linha
  - \t: tabulação
  - \r: backspace
  - \\: caractere da barra invertida
    - (Mostrar no código!!)

printf

```
int main() {
int ret;

    ret = printf("Hello world!\n");
    printf("ret: %d\n", ret);

    printf("%p %p\n", &ret, (int *) ret);

    return 0;
}
```

Entrada

- Com máscara: scanf(string, *var [, *var, ...]);
  - Mesmos especificadores de formato do printf
  - A função precisa receber o endereço da variável à qual o valor digitado será atribuído

```
scanf("%d", &num)
scanf("%c%d", &letra, &num);
scanf("%c", &ch);
scanf("%s", s); // string
scanf("%13c", s);  //le 13 caracteres
scanf(" %c", &ch); //pula brancos
```

Entrada

Caracter de controlo	Descrição
%c	um caracter (exemplo: 'a')
%i	um número inteiro
%d	um número inteiro em decimal (exemplo: 1)
%e ou *%f* ou *%g*	um float (exemplo: 5.9785)
%o	um número em octal
%s	uma simples string
%x	um número em hexadecimal
%p	um ponteiro
%n	um inteiro equivalente ao número de caracteres escritos até ali naquela iteração
%u	um unsigned integer
%[]	**um conjunto de caracteres**
%%	o símbolo %

Entrada

- Observe que `scanf` interrompe a leitura de um string quando encontra um branco, se usado com %s
- Uso de %[]:

- %[aeiou]: apenas as vogais são permitidas
- %[^aeiou]: as vogais não são permitidas
- %[^\n]: interrompe quando recebe um [enter]
- %60[^\n]: admite até 60 caracteres, e para quando encontra um enter

Entrada

- Linhas inteiras:

```
char *gets(char *);
```

  - Lê uma linha inteira, excluindo \n, e coloca \0 no final
  - Com limite de tamanho:
    - `fgets(string, tamMax, stdin)`

Entrada

```
int main() {
char str[10];
char str2[10];
char str3[10];

    gets(str);
    gets(str2);
    fgets(str3, 10, stdin);

    printf("str: %s\nstr2: %s\nstr3: %s", str, str2,str3);

    return 0;
}
```

Entrada

- Caracteres individuais:

```
int getchar(void);
```

  - O caractere digitado é o valor de retorno da função

```
char ret = getchar();
printf("entrada: %c %d\n", ret, ret);

> 1
> entrada: 1 49
```

Exemplo (POSCOMP 2009)

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

int main (void)
{
     char texto[]= "foi muito facil";
     int i;

     for (i = 0; i < strlen(texto); i++){
          if (texto[i] == ' ') break;
     }
     i++;
     for ( ; i < strlen(texto); i++)
          printf("%c", texto[i]);
     return 0;
}
```

O que será impresso quando o programa for executado?

I/O para arquivos

- A entrada do teclado e saída para o monitor são realizadas internamente considerando três dispositivos virtuais: stdin, stdout e stderr
  - Como são dispositivos padrão, referências a stdin e stdout eles são omitidas dos comandos
  - `fgets(string, tamMax, stdin);`
- I/O para arquivos é muito semelhante à operação com stdin e stdout, mas um *handle* ao arquivo tem que ser fornecido

I/O para arquivos

```
// abre ou cria um arquivo
// retorno: handle para arquivo criado
FILE *fopen(const char *name, const char *type);


// fecha um arquivo
// retorna 0 se executado com sucesso
int    fclose(FILE *);


// remove dados do buffer e envia para o arquivo
// retorna 0 se executado com sucesso
int    fflush(FILE *);


// testa final de arquivo
// Retorna diferente de zero se fim do arquivo foi atingido
int feof(FILE *);
```

I/O para arquivos

- fopen: modos de abertura

Mode	Meaning
"r"	Open a text file for reading
"w"	Create a text file for writing
"a"	Append to a text file
"rb"	Open a binary file for reading
"wb"	Create a binary file for writing
"ab"	Append to a binary file
"r+"	Open a text file for read/write
"w+"	Create a text file for read/write
"a+"	Open a text file for read/write
"rb+"	Open a binary file for read/write
"wb+"	Create a binary file for read/write
"ab+"	Open a binary file for read/write

I/O para arquivos

- O handle é obtido no momento da abertura do arquivo

```
FILE *inFile; // variável handle
FILE *outFile;
…
//abre o arquivo para leitura (r) ou gravacao (w)
inFile = fopen(“arquivo.txt”, “r”);
outFile = fopen(“saida.txt”, “w”);
...
fscanf(inFile, “%d”, &num);
...
fprintf(outFile, “O valor lido eh %8.2d\n”, num);
...
fclose(inFile);
fclose(outFile);
```

I/O para arquivos

- Se o *handle* retornar nulo do comando fopen, então ocorreu erro (arquivo não encontrado, arquivo travado contra gravação, permissão negada pelo sistema operacional, etc.)
- Testar:

```
if ((inFile = fopen("arquivo.txt", "r")) == NULL)
{
    printf("Nao consegui abrir.\n");
    exit(1);
}
```

I/O para arquivos

- Equivalências
  - gets() → fgets(arq, tamMax, string);
  - getchar() → fgetc(arq);
  - putc(ch) → fputc(arq, ch)
  - printf → fprintf(arq, string, valor)
  - scanf → fscanf(arq, string, endereço)

I/O para arquivos

```
#include <stdio.h>
#include <stdlib.h>
int main() {
FILE *f;
char ch;

    f = fopen(“arquivo.txt","r");

    while ((ch = fgetc(f)) != EOF)
        printf("%c", ch);

    fclose(f);

    return 0;
}
```

I/O para arquivos (variação)

```
#include <stdio.h>
#include <stdlib.h>
int main() {
FILE *f;
char ch;

    f = fopen("main.c","r");
    while (!feof(f)) {
        ch = fgetc(f);
        printf("%c", ch);
    }
    fclose(f);

    return 0;
}
```

I/O para arquivos

- Movimentação em arquivos

```
// retorna a posição atual do arquivo
long ftell(FILE *f);

// seta posição no arquivo, com relação à variável origin
// retorna 0 se bem sucedido (offset em bytes).
int fseek(FILE *f, long offset, int origin);
```

origin		
Macro	**Valor**	**deslocamento relativo**
SEEK_SET	0	Início do arquivo
SEEK_CUR	1	Posição atual
SEEK_END	2	Fim do arquivo

I/O para arquivos

```
int main() {
FILE *f;
char arquivo[] = "main.c";
long int pos;
char str[512];

    f = fopen(arquivo,"r");

    pos = ftell(f);
    fgets(str, 512, f);
    printf("[posicao: %ld]
%s\n", pos, str);

    fseek(f, 10, SEEK_SET);
    pos = ftell(f);
    fgets(str, 512, f);
    printf("[posicao: %ld]
%s\n", pos, str);

    fseek(f, -15, SEEK_END);
    pos = ftell(f);
    fgets(str, 512, f);
    printf("[posicao: %ld] %s\n",
pos, str);

    fclose(f);

    return 0;
}
```

Numerando linhas

```
int main() {
FILE *f;
char ch;
int nlinha = 1;
    f = fopen("main.c","r");

    printf("%d: ", nlinha++);
    while (!feof(f)) {
        ch = fgetc(f);
        printf("%c", ch);
        if (ch == '\n')
            printf("%d: ", nlinha++);
    }
    fclose(f);

    return 0;
}
```

Entrada da Linha de Comando

- Comunicação pela linha de comando

```
int main(int argc, char *argv[]) {
int i, vezes;
   vezes = atoi(argv[2]);
   for (i = 0; i < vezes; i++)
      printf("%s\n", argv[1]);
   return 0;
}

./exec teste 10
```

I/O para arquivos

- Leitura e escrita de blocos de dados

```
// lê n objetos com tamanho de size bytes cada
// retorno: número de objetos lidos
size_t fread(void *ptr, size_t size, size_t n, FIE *f);


// escreve n objetos com tamanho de siz bytes cada
// retorno: número de objetos escritos
size_t fwrite(void *ptr, size_t size, size_t n, FILE *f);
```

I/O para arquivos

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


typedef struct {
   string nome;
   int matricula;
   char conceito;
} TipoAluno;
```

```
main() {
TipoAluno al;
FILE *f;
int c;

      al.nome = "Pedro"
      al.matricula = 200712;
      al.conceito = 'A';

      f = fopen("arquivo.dat",
"w");
      c = fwrite(&al,

sizeof(TipoAluno), 1, f);
      fclose(f);

}
```

I/O para arquivos

- Outras funções úteis

```
// cria um arquivo temporário (removido quando fechado)
// retorno: handle para arquivo criado
FILE *tmpfile(void);

// gera um nome único que pode ser usado com nome de
   arquivo
char *tmpnam(char *);
```