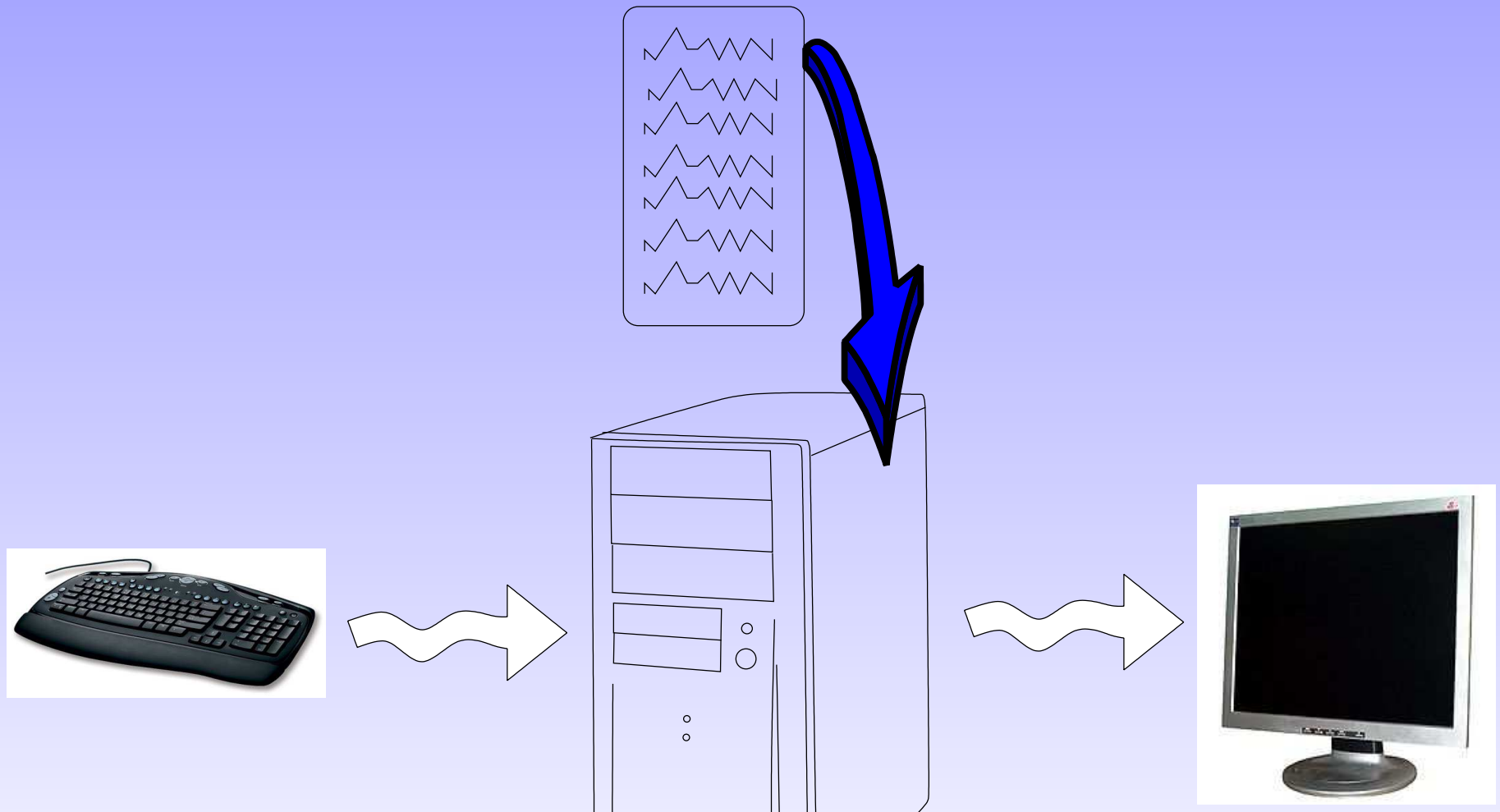
SSC 0301 – IC para Engenharia Ambiental

Arquivos

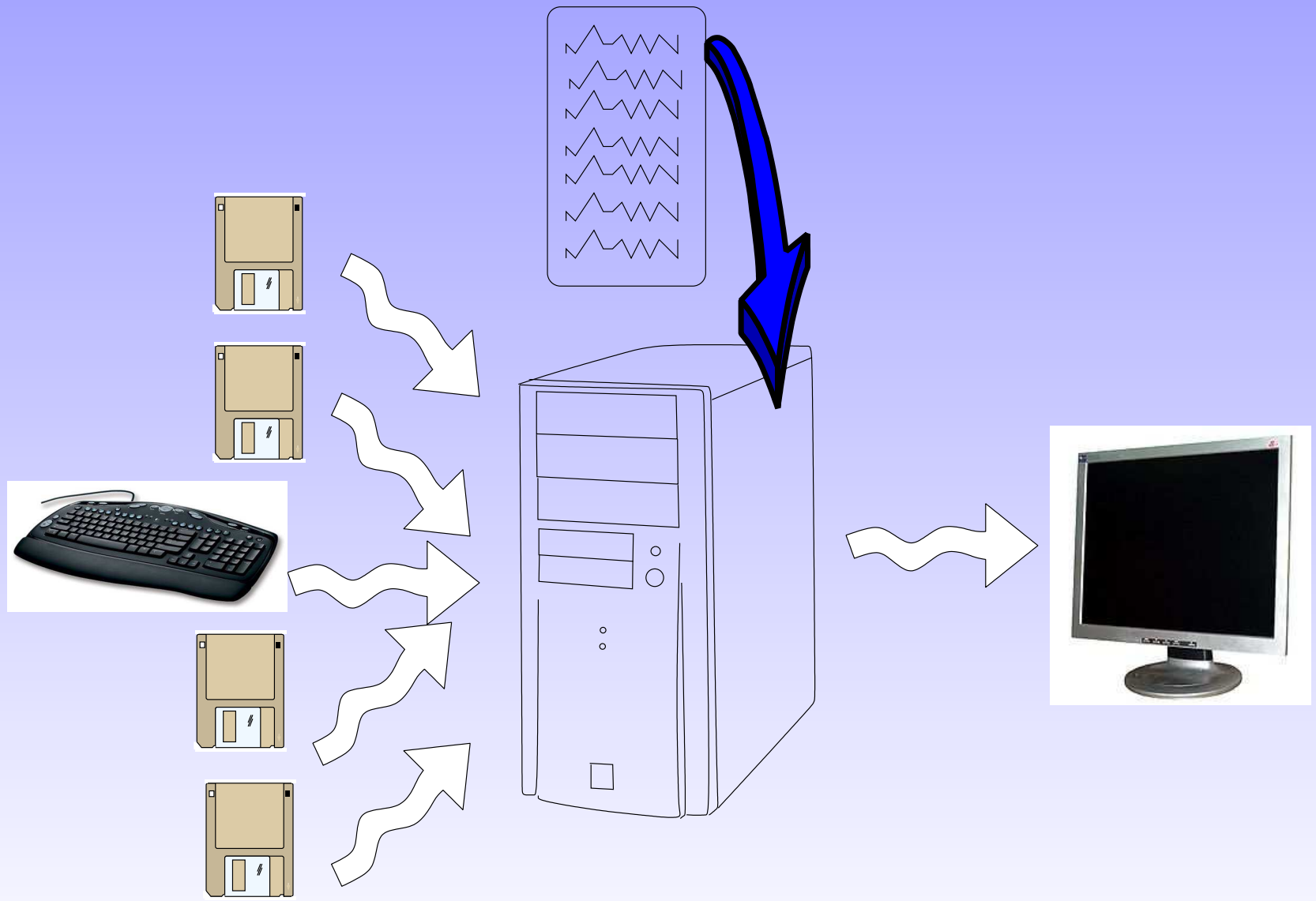
Prof. Marcio E. Delamaro

delamaro@icmc.usp.br

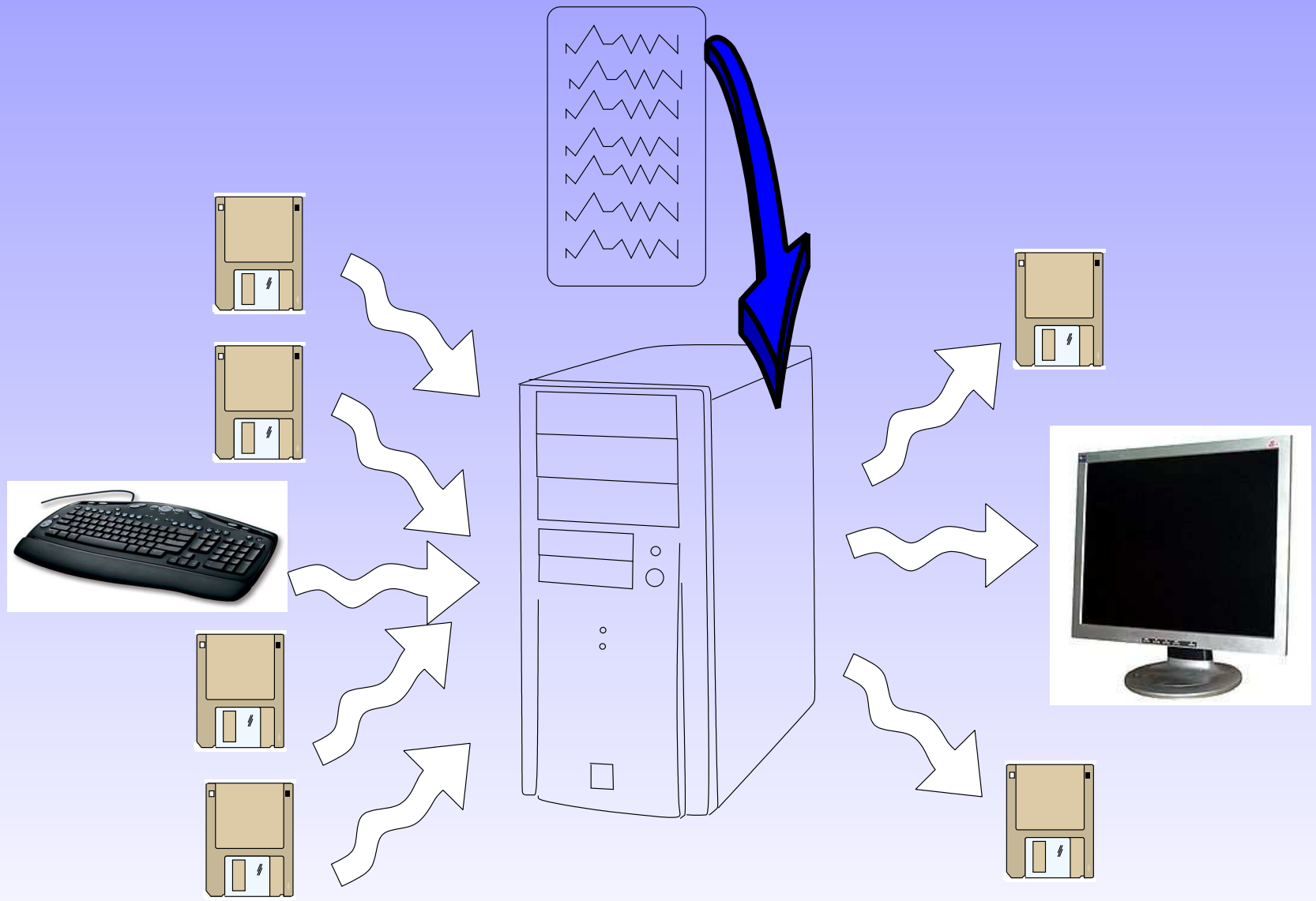
O ambiente de execução



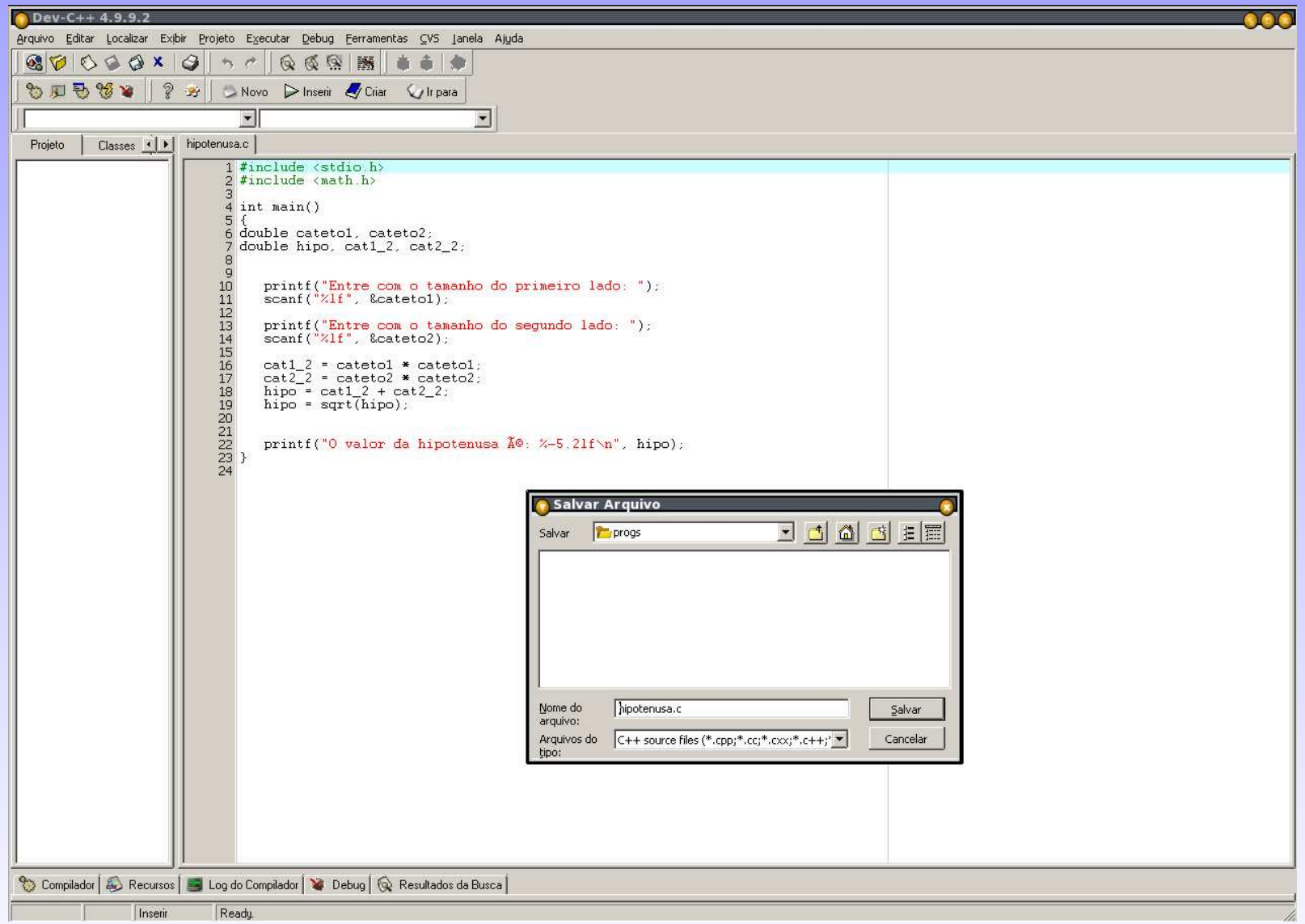
O ambiente de execução



O ambiente de execução

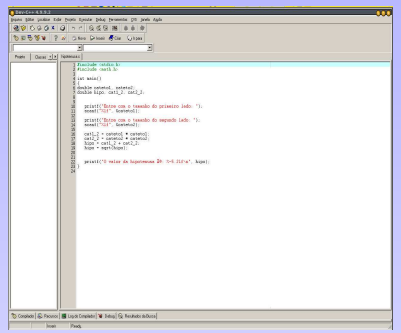


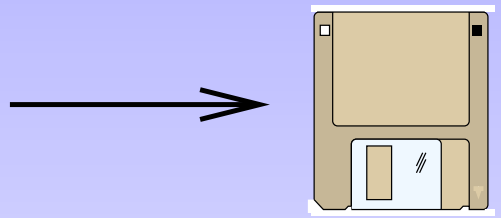
Exemplo: DEV C++



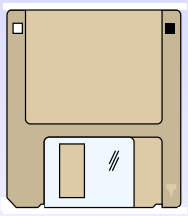
Exemplo: DEV C++

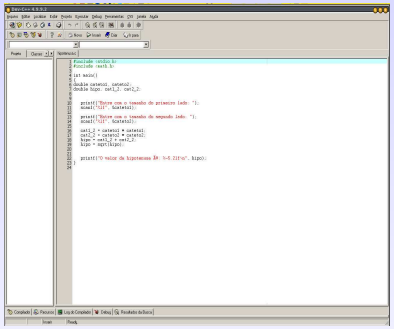
SALVAR



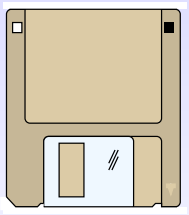


.c





.exe



COMPILAR

O que é um arquivo

- É apenas um sequência de bytes (números).
- O que esses números significam, depende do tipo de programa que vai manipular o arquivo
- Um arquivo que contém apenas números no espaço "legível" é chamado de um arquivo texto
  - Um arquivo criado pelo editor do DEVC++ ou pelo Notepad do windows é um arquivo texto
- Um arquivo que tem valores que não representam texto é chamado de arquivo binário
  - Um arquivo criado pelas versões mais antigas do word é um arquivo binário

Arquivo texto

- “Marcio Eduardo Delamaro” em um arquivo texto seria representado como?
-
```
4d 61 72 63 69 6f 20
45 64 75 61 72 64 6f 20
44 65 6c 61 6d 61 72 6f
```

Nosso exemplo

- Escreva um programa que lê números inteiros do teclado até que um número negativo seja digitado. À medida que vai lendo, ele deve escrever em um arquivo chamado “impares.txt” os números ímpares e em um arquivo “pares.txt” os que são pares.
- Escreva um programa que lê números inteiros de um arquivo chamado “numeros.txt”. À medida que vai lendo, ele deve escrever em um arquivo chamado “impares.txt” os números ímpares e em um arquivo “pares.txt” os que são pares.

Nosso exemplo - V0

```
#include <stdio.h>

int main()
{
int k = 0;

    while ( k >= 0 )
    {
        printf("Digite um numero inteiro =
        scanf("%d", &k);
        if ( k >= 0 )
        {
            printf("%d ", k);
        }
    }
```

Usar arquivos

- São precisas apenas 3 coisas
  - Abrir (criar) o arquivo ou arquivos que se deseja escrever
  - Na hora de escrever, dizer qual é o arquivo
  - Fechar o arquivo

Abrir o arquivo

- Significa que você vai associar um arquivo a uma variável
- Assim, cada vez que você referenciar essa variável, está referenciando o arquivo
- Comando: `fopen`
- `fopen(<nome do arquivo> , <modo>)`

Nosso exemplo – v0.1

```
#include <stdio.h>

int main()
{
int k = 0;
FILE *arq_impar, *arq_par;

    // cria os dois arquivos
    arq_impar = fopen("impares.txt", "w");
    arq_par = fopen("pares.txt", "w");

    while ( k >= 0 )
    {
        printf("Digite um numero inteiro ==> ");
        scanf("%d", &k);
        if ( k >= 0 )
        {
            printf("%d ", k);
        }
    }
}
```

Fechar o arquivo

- Indica que o arquivo não vai ser mais usado
- Grava os dados que faltam
- A partir desse ponto, arquivo não pode ser mais utilizado
- Função `fclose`
- `fclose(<variavel arquivo>);`

Nosso exemplo – v0.2

```
#include <stdio.h>
int main()
{
int k = 0;
FILE *arq_impar, *arq_par;
   // cria os dois arquivos
   arq_impar = fopen("impares.txt", "w");
   arq_par = fopen("pares.txt", "w");
   while ( k >= 0 )
   {
       printf("Digite um numero inteiro ==> ");
       scanf("%d", &k);
       if ( k >= 0 )
       {
           printf("%d ", k);
       }
   }
   fclose(arq_impar);
   fclose(arq_par);
}
```

Escrever no arquivo

- Usa-se a mesma idéia de escrever na tela, mas deve-se indicar qual é o arquivo
- Usa-se para isso uma outra função, parecida com o `printf`
- Função `fprintf`
- `fprintf(<variável arquivo>, <string formato>, <valores>)`

Nosso exemplo – v1.0

```
int main()
{
int k = 0;
FILE *arq_impar, *arq_par;
   arq_impar = fopen("impares.txt", "w");
   arq_par = fopen("pares.txt", "w");
   while ( k >= 0 )
   {
       printf("Digite um numero inteiro ==> ");
       scanf("%d", &k);
       if ( k >= 0 )
       {
           if ( k % 2 == 0)
               fprintf(arq_par, "%d ", k);
            else
               fprintf(arq_impar, "%d ", k);
       }
   }
   fclose(arq_impar);
   fclose(arq_par);
}
```

Modo de abertura

- Indica como deve ser feito o acesso ao arquivo
  - `"r"` – Open a file for reading. The file must exist.
  - `"w"` – Create an empty file for writing. If the file exists its content is erased.
  - `"a"` – Append to a file. Writing operations append data at the end of the file. The file is created if it does not exist.
  - `"r+"` – Open a file for reading and writing. The file must exist.
  - `"w+"` – Create an empty file for both reading and writing. If the file exists its content is erased.
  - `"a+"` – Open a file for reading and appending. All writing operations are performed at the end of the file, protecting the previous content to be overwritten. You can reposition (fseek, rewind) the internal pointer to anywhere in the file for reading, but writing operations will move it back to the end of file. The file is created if it does not exist.

Nosso exemplo – V1.1

- O programa sempre “zera” os arquivos no início da execução
- Para mudar isso, deve-se alterar a forma de abrir os arquivos
-
```
arq_impar = fopen("impares.txt", "a");
arq_par = fopen("pares.txt", "a");
```
- Com isso, os dados são sempre adicionas àqueles que já existiam, gravados nas execuções anteriores.
- Se o arquivo não existe ainda, ele é criado vazio.

Nosso exemplo (2)

- Vamos ler os números de um arquivo e separar entre pares e ímpares
- Para ler, também existe uma função especial, parecida com o `scanf`
- Função `fscanf(<variável arquivo>, <formatos>,`
- Para isso, o arquivo deve ser aberto, usando um modo que permita a leitura

Nosso exemplo – V2.0

```
int main()
{
int k = 0;
FILE *arq_impar, *arq_par, *arq_todos;
   arq_todos = fopen("numeros.txt", "r");
   arq_impar = fopen("impares.txt", "a");
   arq_par = fopen("pares.txt", "a");
   while ( k >= 0 )
   {
       fscanf(arq_todos, "%d", &k);
       if ( k >= 0 )
       {
           if ( k % 2 == 0)
               fprintf(arq_par, "%d ", k);
            else
                fprintf(arq_impar, "%d ", k);
       }
   }
   fclose(arq_impar);
   fclose(arq_par);
   fclose(arq_todos);
}
```

Fim do arquivo

- No nosso exemplo é necessário que coloquemos um número negativo no fim da sequência que queremos ler
- `1 2 3 4 5 6 -1`
- Podemos também alterar o programa de modo que ele consiga reconhecer quando o arquivo de entrada acabou
- Basta lembrar que o função `scanf` (e também o `fscanf`) retorna o número de leituras que conseguiu fazer
- Portanto, quando esse número for 0, o arquivo terminou

Nosso exemplo – V2.1

```
int main()
{
int k, quantos = 1;
FILE *arq_impar, *arq_par, *arq_todos;
   arq_todos = fopen("numeros.txt", "r");
   arq_impar = fopen("impares.txt", "a");
   arq_par = fopen("pares.txt", "a");
   while ( quantos == 1 )
   {
       quantos = fscanf(arq_todos, "%d", &k);
       if ( quantos == 1 )
       {
           if ( k % 2 == 0)
               fprintf(arq_par, "%d ", k);
            else
                fprintf(arq_impar, "%d ", k);
       }
   }
   fclose(arq_impar);
   fclose(arq_par);
   fclose(arq_todos);
}
```

Será que o arquivo existe?

- Com a função `fopen` também é possível verificar se o arquivo existe ou não
- Por exemplo, se o arquivo de entrada não foi criado pelo usuário, o programa deveria dar uma mensagem
- Basta verificar se o valor retornado pelo `fopen` é igual à conatante **NULL**
- Se for, um erro ocorreu

Nosso exemplo – V2.2

```
int main()
{
int k, quantos = 0;
FILE *arq_impar, *arq_par, *arq_todos;
   arq_todos = fopen("numeros.txt", "r");
   if ( arq_todos == NULL )
   { // erro ocorreu
       printf("Arquivo de entrada não existe!\n");
       return -1;
   }
   arq_impar = fopen("impares.txt", "a");
   arq_par = fopen("pares.txt", "a");
   :::
   :::
}
```

Exercício

- Escreva um programa que lê um arquivo texto e imprime, em um outro arquivo: o número de linhas, o número total de caracteres, o número de espaços em branco e o número de caracteres não brancos encontrados.
- O nome do arquivo de entrada deve ser digitado pelo pelo usuário.
- Sugestão: use a função `fgets` para ler as linhas da entrada

Função `fgets`

- `fgets(<vetor>, <tamanho>, <arquivo>)`
- Vetor indica a variável que vai ser lida
- Tamanho é o tamanho máximo da linha a ser lida
- Arquivo é a variável que referencia o arquivo a ser lido
- Se um erro ocorrer, retorna NULL

Solução

```
#include <stdio.h>
#include <string.h>
char filename[100];
char linha[100];
int main()
{
int cont_linha = 0, cont_char = 0, cont_branco = 0, t, i;
FILE *arq_in, *arq_out;
   printf("Qual o nome do arquivo? ");
   gets(filename);
   arq_in = fopen(filename, "r");
   if ( arq_in == NULL )
   { // erro ocorreu
       printf("Arquivo de entrada não existe!\n");
       return -1;
   }
   arq_out = fopen("saida.txt", "w");
```

Solução

```
    while ( fgets(linha, 100, arq_in) != NULL )
    {
        cont_linha++;
        t = strlen(linha);
        cont_char = cont_char + t;
        for (i = 0; i < t; i++)
        {
            if (linha[i] == ' ' )
               cont_branco++;
        }
    }
    fprintf(arq_out, "Numero de linhas: %d\n", cont_linha);
    fprintf(arq_out, "Numero de caracteres: %d\n", cont_char);
    fprintf(arq_out, "Numero de brancos: %d\n", cont_branco);
    fprintf(arq_out, "Numero de nao brancos: %d\n", cont_char - con
    fclose(arq_out);
    fclose(arq_in);
    return 0;
}
```