LP1

Linguagem de Programação 1

Introdução

Objetivos:

- proporcionar ao aluno o desenvolvimento do raciocínio lógico voltado à programação de computadores.

- capacitar o aluno a interpretar os problemas e transpor para os algoritmos e para a linguagem de programação C.

Ementa

- Introdução a Linguagem C:
- Tipos de Dados;
- Variáveis e constantes;
- Comentários e indentação;
- Expressões em C;
- Operadores;
- Funções de Entrada/Saída pelo Console;
- Conversão de tipos.

Programas em C - Declarações para controle de programas;

Comandos e instruções: decisão e repetição;

Vetores, Matrizes e Strings;

Estruturas e Tipos Definidos Pelo Usuário;

Ponteiros e Alocação Dinâmica;

Funções;

Funções de String de Caracteres;

Funções Recursivas.

Bibliotecas e Arquivos de Cabeçalho.

Arquivos em C.

Trabalhos e Provas:

- T1: 06 de outubro
- P1: 07 de outubro

- T2: 02 de dezembro
- P2: 09 de dezembro

- IFA: 16 de dezembro

- **Pontuação**

Média 1º bim = 0,6 * P1 + 0,4 * T1

Média 2º bim = 0,5 * P2 + 0,4 * T2 + Sim. Enade*

Média >= 6,0 – Aprovado

Média < 4,0 – Reprovado

Média >= 4,0 e < 6,0 → IFA

*Sim. Enade acontece dia 31/10

Lógica de programação

É A TÉCNICA DE ENCADEAR PENSAMENTOS PARA ATINGIR UM OBJETIVO.

Sequência lógica

SÃO PASSOS EXECUTADOS ATÉ ATINGIR UM OBJETIVO OU SOLUÇÃO DE UM PROBLEMA

Instruções

Um conjunto de regras ou normas definidas para a realização ou emprego de algo.

Por exemplo, se precisar trocar o pneu de um carro, precisaremos colocar em prática uma série de instruções:

Pegar o macaco hidráulico

Encaixar o macaco no carro

Soltar os parafusos do pneu

Erguer o carro

Etc...

É evidente que essas instruções devem ser executadas em uma ordem adequada – não se pode erguer o carro antes de pegar o macaco.

Dessa maneira, uma instrução tomada em separado não tem muito sentido; para obtermos o resultado, precisamos colocar em prática o conjunto de todas as instruções, na ordem correta.

UM PROGRAMA É UMA SEQUÊNCIA DE INSTRUÇÕES ORDENADAS FEITAS AO COMPUTADOR DE FORMA A RESOLVER UM PROBLEMA

Algoritmo

É uma sequência **finita** de passos que levam à execução de uma tarefa.

Estas tarefas não podem ser redundantes nem subjetivas na sua definição, devem ser claras e precisas.

Muitas vezes um algoritmo é comparado a uma receita de bolo, onde cada passo da confecção do bolo seria representado pelas instruções do algoritmo.

Em LP1 estudaremos:

- Como traduzir um algoritmo para a **linguagem C**, com o intuito de construir um programa de computador que resolva um dado problema.

Essa disciplina anda em paralelo com a disciplina de **APO**.

- Código fonte → código que escrevemos.

- Compilador → traduz o que escrevemos para linguagem de máquina.
  - Checa a sintaxe do que está escrito (de acordo com a linguagem escrita) e, se não houver erros, traduz para linguagem de máquina.
  - Faz isso sequencialmente para todas as instruções do código.

- Gera um arquivo do tipo .OBJ que ainda não é executável.

- Linkeditor → agrega as funções necessárias ao código gerado pelo compilador, gerando assim o arquivo executável.

C é uma linguagem de programação compilada de propósito geral, estruturada, imperativa, procedural, padronizada pela ISO, criada em 1972, por Dennis Ritchie, no AT&T Bell Labs, para desenvolver o sistema operacional Unix.

C é uma das linguagens de programação mais populares e existem poucas arquiteturas para as quais não existem compiladores para C.

C tem influenciado muitas outras linguagens de programação, mais notavelmente C++, que originalmente começou como uma extensão para C.

Passos para a criação de um programa executável na linguagem C:

- Digitar seu programa com o auxílio de um processador de textos no modo texto e gravá-lo em disco, dando a ele um sufixo .c – esse é o código fonte.
- Exemplo: **programa1.c**
- Compilar o código fonte seguindo as instruções do seu compilador, o que criará um programa com sufixo .OBJ em disco – chamado de objeto.
- Linkeditar o objeto seguindo as instruções do seu linkeditor, o que resultará em um programa com sufixo .exe em disco (no Windows) ou com o atributo executável (no Linux) – programa executável.

Conhecendo o Dev-C++

O Dev-C++ é uma IDE (*Integrated Development Environment*) simples que auxilia na criação de programas usando a linguagem C.

Estrutura básica de um programa em C

diretivas

```
tipo nomeFunc(declaração de parâmetros)
{
    declaração de variáveis;
    instrução_1;
    instrução_2;
    …
    instrução_n;
    retorno valor;
}
```

Menor programa C

```
#include <stdio.h>

int main()
{

    return 0;

}
```

Menor programa C

```
#include <stdio.h>

int main()
{
    printf(“Ola, mundo!”);

    return 0;

}
```

Função de escrita na tela (saída)

```
printf(“frase a ser escrita na tela…”);
```

```
printf(“bla bla bla”);
```

A função *printf* admite o uso de caracteres/comandos especiais para diferentes efeitos na tela.

Exemplo: *\n*

Esses dois caracteres, quando colocado na frase (**string!**) de um *printf*, faz com que seja pulada uma linha.

```c
#include <stdio.h>

int main()
{

  printf("Ola,\nmundo!\n");

  return 0;

}
```

Alguns caracteres especiais para o *printf*

\n – nova linha

\t - tabulação

\f – salto de página

\a – sinal sonoro

\r – retorna o cursor no início da linha

\\ - barra invertida

\0 – caracter nulo

\’ – aspas simples

\” – aspas duplas

Variáveis

Para que um programa use valores (para fazer cálculos, apresentar informações na tela, etc), é preciso utilizar *variáveis*.

Variáveis são elementos que, como o próprio nome diz, terá valor variável.

Uma variável é um espaço reservado na memória do computador para armazenar um valor binário.

Esse valor será interpretado pelo programa de acordo com o tipo definido.

Na linguagem C, as variáveis possuem **tipo**.

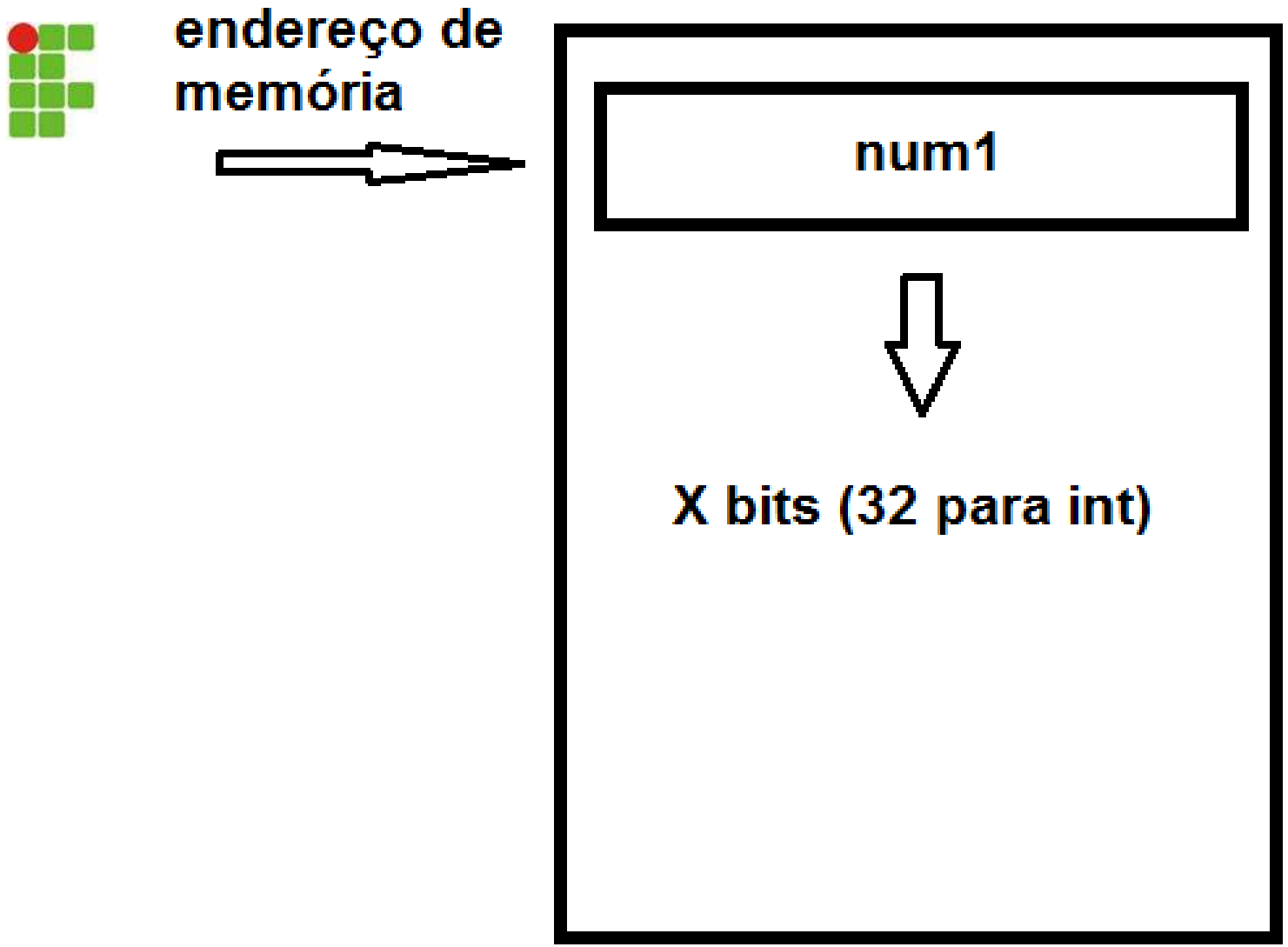
O tipo de uma variável define qual o tipo de dado que a variável poderá armazenar.

Tipos básicos na linguagem C:

int – armazena valores numéricos inteiros.

float – armazena valores numéricos fracionados.

char – armazena um caracter.


endereço de
memória
num1
X bits (32 para int)

Atribuição de valores

É feita da direita para esquerda.

Exemplos

```
num = 5;
altura = 1.80;
```

Exemplo

```
#include <stdio.h>

int main()
{
  int num;

  num = 5;

  printf("Valor de num = %d\n", num);

  return 0;
}
```

Exemplo

```
#include <stdio.h>

int main()
{
  float num;

  num = 5.5;

  printf("Valor de num = %f\n", num);

  return 0;
}
```

Apresentando várias variáveis em um único *printf*

```
printf(“var1 = %d; var2 = %d; var3 = %d”, var1, var2, var3);
```

Exemplo

```
#include <stdio.h>

int main()
{
  int num1;
  float num2;

  num1 = 5;
  num2 = 7.2;

  printf("num1 = %d, num2 = %f\n", num1, num2);

  return 0;
}
```

Outros tipos de variáveis

char – 8 bits, de -128 a 127

int – 32 bits, de -2.147.483.648 a 2.147.483.647

short – 16 bits, de -32.765 a 32.767

long – 32 bits, de -2.147.483.648 a 2.147.483.647

unsigned char – 8 bits, de 0 a 255

unsigned – 32 bits, de 0 a 4.294.967.295

unsigned long – 32 bits, de 0 a 4.294.967.295

float – 32 bits, de $3,4x10^{-38}$ a $3,4x10^{38}$

double – 64 bits, de $1,7x10^{-308}$ a $1,7x10^{308}$

long double – 80 bits, $3,4x10^{-4932}$ a $3,4x10^{4932}$

void – 0 bits, sem valor

Códigos de formatação do printf

%c – imprime no formato caracter (*char*);

%d – imprime no formato decimal (*int*);

%i – imprime no formato inteiro (*int*);

%e – imprime no formato científico com e minúsculo

%E - imprime no formato científico com E maiúsculo

%f – imprime no formato ponto flutuante (*float*)

%s – imprime como uma string de caracteres

%u – imprime como decimal sem sinal

%x - imprime um número na base hexadecimal. As letras serão minúsculas.

%X - imprime um número na base hexadecimal. As letras serão maiúsculas.

Para escrever casas decimais:

%.xf – onde x é o número de casas decimais

Para escrever espaços em branco antes do valor:

%xf – onde x é o número de espaços em branco antes do valor.

Inicializando variáveis

É possível criar a variável e já inserir um valor dentro dela.

Exemplos:

```
int i = 0;

char c = ‘A’;

float f = 3.4;
```

Etc…

É possível criar várias variáveis de um mesmo tipo em uma única linha.

Exemplos:

```
int var1, var2, var3;

float f1, f2, f3 = 3.5, f4;
```

Nomes de variáveis

Não devem começar com número
Exemplo: 2num (errado!)

Não devem conter caracteres especiais
Exemplo: num$, teste% (errado!)

Boa prática!
Os nomes de variáveis devem ser significativos dentro do programa.

Exemplos: num, salario, media, nota, ...

Operadores com variáveis

Binários (usados com dois valores)

+ soma

- subtração

* multiplicação

/ divisão

% módulo (resto da divisão)

Exemplos

x = y + 10;

x = a – 5;

x = h * i;

x = j / i;

x = x + 10;

Atalhos

x = x + 10;

é igual se for feito como

x += 10;

Exemplo

```
#include <stdio.h>

int main()
{
  int num1, num2, soma;

  num1 = 10;
  num2 = 5;

  soma = num1 + num2;

  printf("soma = %d\n", soma);

  return 0;
}
```

Exercícios!

Na próxima aula...

Função de entrada de dados

e

mais exercícios!