Anhanguera

Aqui o seu esforço ganha força.

Aula 02 – Revisão da Linguagem C

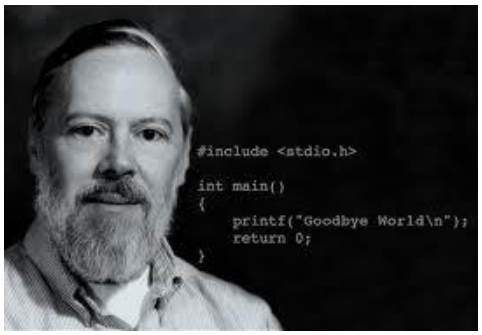
Prof. Esp. Rodrigo Hentz

Revisão

- Origem do C
- Estrutura de um programa em C
- Tipos de dados básicos
- Identificadores
- Variáveis
- Constantes
- Operadores
- Comandos de controle
- Matrizes e vetores
- Estruturas

Origem do C

- Inventada e implementada primeiramente por Dennis Ritchie em um DEC PDP-11 que utiliza o sistema operacional UNIX.
- C é o resultado de um processo de desenvolvimento que começou com uma linguagem mais antiga chamada BCPL, desenvolvida por Martin Richards e resultou em uma linguagem chamada B, inventada por Ken Thompson
- Em 1970 B levou ao desenvolvimento do C



Ken Thompson (esq.) e Dennis Ritchie, criadores do Unix, trabalham em um computador DEC PDP11 e dois terminais Teletype modelo 33

Estrutura de um programa em C

- Os programas em C consistem em uma ou mais funções.
- A única função que necessariamente precisa estar presente é a denominada main(), pois é a primeira função chamada quando a execução do programa começa.
- A forma geral de um programa em C pode ser considerada como:

```
<declarações globais>
<tipo desenvolvido> main(lista de parametros)
{
        <sequencia de comandos>
}
<tipo desenvolvido> fN(lista de parametros)
{
        <sequencia de comandos>
}
```

Estrutura de um programa em C

- Exemplo de um programa em C

```
int main(int argc, char *argv[]) {

    printf("\n");
    printf("Hello World!");
    printf("\n");
    return 0;
}
```

Tipos de dados básicos

- Todas as variáveis em C tem um tipo
- Cada tipo define os valores que uma variável pode armazenar;
- Cada tipo ocupa uma certa quantidade de memória.

Tipo	Tamanho	Valores Válidos
char	**1 byte**	**letras e símbolos: 'a', 'b', 'H', '^', '*','1','0'**
int	**2 bytes**	**de -32767 até 32767 (apenas números inteiros)**
float	**4 bytes**	**de -3.4E38 +3.4E38 com até 6 dígitos de precisão**
double	**8 bytes**	**de -1.7E308 até +1.7E308 com até 10 dígitos de precisão**

Identificadores

- Identificadores são os nomes dados a variáveis, funções, rótulos e vários outros objetos definidos pelo programador e podem variar de um a vários caracteres

- O primeiro caractere deve ser uma letra ou um sublinhado e os subsequentes devem ser caracteres

Correto	**Incorreto**
count	1count
test23	hi!there
high_balance	high..balance

Identificadores

- Observação: em C letras maiúsculas e minúsculas são tratadas de maneira diferente (case sensitive).

- Logo:

  - _variavelnome é diferente de _VARIAVELNOME
  - _varialvelnome é diferente de _variavelNome

Variáveis

- Uma variável é uma posição nomeada de memória utilizada para guardar um valor que pode ser manipulada pelo programa
- Todas as variáveis devem ser declaradas antes de seu uso
- Forma geral de declaração:

  tipo *lista_de_variaveis*;

- Tipo é um tipo de dado válido em C e ***lista_de_variaveis*** pode consistir em um ou mais nomes de identificadores separados por vírgulas.
- Exemplo:

  int i, j, k;

  double total, saldo, perda;

Variáveis

- As variáveis podem ser declaradas em três lugares básicos:

- Dentro das funções (denominadas **variáveis locais**)

- Na definição de parâmetros das funções (denominadas **parâmetros formais**)

- Fora de todas as funções (denominadas **variáveis globais**)

Variáveis

- Quais são as variáveis globais, variáveis locais e parâmetros formais no exemplo abaixo?

```
int count;

void somar(int valor);

int main(int argc, char *argv[]) {

     count = 100;
     somar(50);
     printf("Valor : %d", count);
     return 0;
}

void somar(int valor)
{
     count += valor;
}
```

Constantes

- Valores fixos que o programa não pode alterar.

```
#define PI 3.14159265

int main(int argc, char *argv[]) {

    float raio;
    double area;
    printf ("Digite o raio do seu circulo:");

    scanf ("%f", &raio);
    area = (raio*raio)*PI;
    printf ("A area do seu circulo e: %.2f ", area);

    return 0;
}
```

Operadores

- A linguagem C define quatro classe de operadores:
  - Aritméticos
  - Relacionais
  - Lógicos
  - Bit a Bit
- Os operadores são utilizados para realizar operações no programa com os valores de suas variáveis.

Operadores Aritméticos

- Os operadores -, +, * e / trabalham em C da mesma forma em que a maioria das outras linguagens.

Operadores Aritméticos	
+	Soma
-	Subtração
/	Divisão
%	Resto de divisão
++	Incremento
--	Decremento

Operadores Relacionais

- Os operadores relacionais se referem as relações que os valores podem ter uns com os outros. No termo operadores lógicos refere-se a maneira como as relações podem ser conectadas.

Operadores Relacionais	
>	Maior que
>=	Maior que ou igual
<	Menor que
<=	Menor que ou igual
==	Igual
!=	Diferente

Operadores Lógicos

- Os operadores lógicos refere-se a maneira como as relações (operadores relacionais) podem ser conectadas.

Operadores Lógicos	
&&	AND
||	OR
!	NOT

Operadores

- É permitido combinar diversas operações em uma expressão como mostrado aqui:

```
(10 > 5 && !(10 < 9) || 3 <= 4)
```

Qual o resultado desta expressão?

Precedência

- Define quais expressões serão avaliadas primeiro

Precedência	
Maior	
	!
	> >= < <=
	== !=
	&&
Menor	
	||

```
(10 > 5 && !(10 < 9) || 3 <= 4)
```

10 > 5 && !(10 < 9) || 3 <= 4

10 > 5 && TRUE || 3 <= 4

TRUE && TRUE || TRUE

TRUE || TRUE

TRUE

Comandos de controle

- Os comandos de controle em C definem a forma como a programa irá executar seus procedimentos.

- O C é dividido nestes grupos
  - Seleção
  - Iteração
  - Desvio
  - Rótulo
  - Expressão
  - Bloco

Comandos de controle - Seleção

- C suporta dois tipos de comandos de seleção: **if** e **switch**

- Além disso o operador **?** é uma alternativa em certas circunstancias.

- Forma geral if:

If (expressao) comando; **else** comando;

Comandos de controle – Seleção If

- Exemplo:

```
int magic = rand(); int guess;
printf("Adivinhe o número mágico: ");
scanf("%d", &guess);
if (guess == magic)
     printf("**Certo**");
else
     printf("**Errado**");
```

Comandos de controle – Seleção If

- Um if aninhado é um comando if que é objeto de outro if ou else.

```
if (i > 5) {
     if (i > 8) printf("Pontuacao otima");
     else
          if (i > 6) printf("Pontuacao boa");
          else printf("Pontuacao regular");
}
else printf("Nao alcancou pontuacao");
```

Comandos de controle – Seleção switch

- Comando de seleção múltipla. Ao encontrar o valor coincidente as expressões são executadas.

- Forma geral:

```
switch (expressão) {
        case constante1:
                comandos;
                break;
        ...
        default:
                comandos;
}
```

Comandos de controle – Seleção switch

- Exemplo:

```
switch (i){
     case 10:
          printf("Excelente");
     break;
     case 5:
          printf("Bom");
     break;
     default:
          printf("Regular");
     break;
}
```

Comandos de controle – for

- Comando para execução de procedimentos até que uma condição seja atendida.

- Forma geral:

```
for (expressao_inicial; condicao_parada; iteração_controle)
        comandos;
```

Comandos de controle – for

- Exemplo:

```
int x;
for (x = 1; x <= 100; x++)
    printf("Valor de x é %d", x);
```

Comandos de controle – while

- Comando para execução de procedimentos enquanto uma condição é atendida.

- Forma geral:

while (condição) comando;

Comandos de controle – while

- Exemplo:

```
int x = 0;
while (x == 0) {
    printf("Entre com x");
    scanf("%d", &x);
}
```

Comandos de controle – do while

- Comando para execução de procedimentos enquanto uma condição é atendida.
- Sempre executa uma vez pois a verificação da condição é realizada no final.

do { comando } while (condição);

Comandos de controle – do while

- Exemplo:

```
int num = 0;
do {
    printf("Entre com numero");
    scanf("%d", &num);
} while (num == 0);
```

Comandos de saída- printf

- A função printf exibe um ou mais dados na tela. Para tanto ele deve receber pelo menos dois parâmetros, separados por vírgula.

```
#include <stdio.h> //Necessário para usar a função printf

void main (void)
{
   printf("%s","Isto é uma string ....\n");
   printf("%s","Outra string ....\n");
   printf("%s","Terceira string\n");
}
```

Comandos de entrada - scanf

- A função scanf recebe valores e atribui para uma variável.

```
#include <stdio.h>
#include <stdlib.h>
int main(int argc, char *argv[]) {
{
     char sexo;
     printf("Digite o sexo\n[F] - Feminino\n[M] - Masculino\n");
     scanf("%c",&sexo);
     return 0;
}
```

Exercício

- Escrever um programa em linguagem C para que o usuário informe o raio de um círculo e retorne a área e o comprimento do mesmo bem como sua descrição de área: pequena, média ou grande.
- O programa deve exibir um menu de escolha. Caso entre com 1 o programa vai fazer o calculo, caso 2 termina o programa e exibe uma mensagem de finalização.
- A descrição da área será definida pela verificação da área como abaixo:
  - Se a área do circulo ser menor do que 100 exibir a mensagem “Área pequena”.
  - Se a área do círculo ser maior que 100 mas menor do que 500 exibir a mensagem “Área média”
  - Se a área do círculo ser maior que 500 exibir a mensagem “Área grande”.

Utilizar constantes, variáveis, do while e if.

area = (raio*raio) * PI; comprimento = raio * 2 *PI;

Exercício

```c
#define PI 3.14159265

int main(int argc, char *argv[]) {
    float raio;  double area, comprimento;  int opcao;
    do{
        printf("Digite uma opcao: \n 1 Calcular \n 2 Sair \n");
        scanf("%d", &opcao);
        if (opcao == 1){
            printf ("Digite o raio do seu circulo:\n");
            scanf ("%f", &raio);
            area = (raio*raio)*PI;
            comprimento = raio*2*PI;
            printf ("A area do seu circulo e: %.2f cm\n", area);
            printf ("O comprimento do seu circulo e: %.2f cm\n", comprimento);
        } else printf("programa finalizado");
     } while (opcao == 1);
    return 0;
}
```

Matriz Unidimensional

- Coleção de variáveis de mesmo tipo referenciada por um nome em comum.

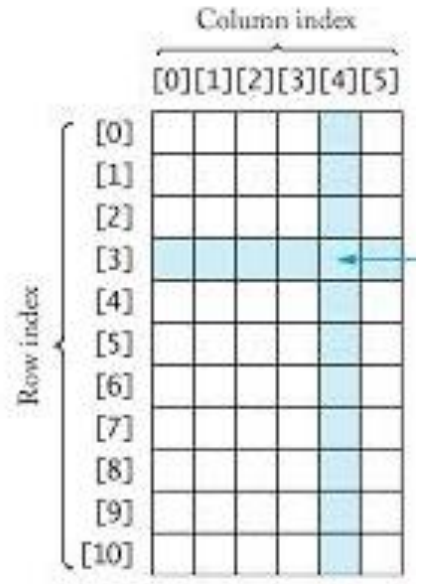
```
int x[10 ]; int t;
for(t = 0; t < 10 ; ++t)
{
     x[t] = t;
}
```

0	1	2	3	4	5	6	7	8	9	Índices
65	78	2	4	10	35	38	9	12	50	Conteúdo da Matriz

Matriz Bidimensional

- São “tabelas” na forma de linha x coluna.

	Coluna 0	Coluna 1	Coluna 2	Coluna 3	Coluna 4
Linha 0	9	7	4	3	5
Linha 1	8	6	4	2	4
Linha 2	5	8	9	1	4



Definição: int matriz[3][5];

Exercício

- Escrever um programa em linguagem C para que o usuário informe um número por 10 vezes
- Estes números devem ser armazenados em uma matriz de tamanho de 10 posições.
- Ao final, mostrar os 10 números apresentados.

Estruturas

- STRUCT
- Uma struct é uma variável especial que contém diversas outras variáveis normalmente de tipos diferentes.
- As variáveis internas contidas pela struct são denominadas membros da struct.
- Podemos dizer que as structs da linguagem C são o equivalente ao que se denomina registros em outras linguagens de programação.

```
struct identificador>
{
    <listagem dos tipos e membros>;
}
struct <identificador> <variavel>;
```

Estruturas

- Exemplo:

```
struct ficha_de_aluno
{
   char nome[50];
   char disciplina[30];
   float nota_prova1;
   float nota_prova2;
};

struct ficha_de_aluno aluno;
```

Exercício

- Criar a struct ficha_de_aluno e fazer um pequeno programa para cadastrar os dados deste aluno.

Exercício

```c
/* Criando a struct  */
struct ficha_de_aluno
{
  char nome[50];
  char disciplina[30];
  float nota_prova1;
  float nota_prova2;
};


int main(int argc, char *argv[]) {

  /*Criando a variável aluno do tipo struct ficha_de_aluno */
  struct ficha_de_aluno aluno;

  printf("\n---------- Cadastro de aluno ----------\n");

  printf("\nNome do aluno ......: ");
  fflush(stdin);
  scanf("%s", &aluno.nome);

  printf("\nDisciplina ......: ");
  fflush(stdin);
  scanf("%s", &aluno.disciplina);
```

Exercício

```
  printf("\nInforme a 1a. nota ..: ");
  fflush(stdin);
  scanf("%f", &aluno.nota_prova1);

  printf("\nInforme a 2a. nota ..: ");
  fflush(stdin);
  scanf("%f", &aluno.nota_prova2);

  printf("\n --------- Lendo os dados da struct ---------\n");
  printf("Nome ...........: %s \n", aluno.nome);
  printf("Disciplina .....: %s \n", aluno.disciplina);
  printf("Nota da Prova 1 ...: %.2f\n" , aluno.nota_prova1);
  printf("Nota da Prova 2 ...: %.2f\n" , aluno.nota_prova2);

  getch();

  return 0;
}
```

Finalização