Sistemas Operacionais I

Laboratório 02

Introdução a Linguagem C e Desenvolvimento na plataforma Linux

História da Linguagem C

- Desenvolvimento inicial AT&T Bell Labs por Dennis Ritchie em 1972

- Derivada da Linguagem B (versão de BCPL)
- Criada para implementar o Unix
- Livro Clássico:
  - (Primeira Edição 1978) : The C Programming Language, Second Edition ,Brian W. Kernighan and Dennis M. Ritchie, Prentice Hall, Inc., 1988.

Linguagem C é

- Propóstio geral
- Estruturada
- Procedural / imperativa
- Para uso com o Unix
- Muito usada para desenvolver sistemas e aplicativos
- Disponível para muitas plataformas
- Padronização ANSI (American National Standards Institute)

Linguagem C é

- Considerada de nível médio
- C permite manipular bits, bytes e endereços de memória como uma linguagem de baixo nível
- C possui construções e rotinas como linguagens de alto nível
- C é para profissionais; Uso primário para desenvolvimento de sistemas

Estrutura de um programa em C

Programa C

- **Directivas ao Pré-Processador**
  - **Includes**
  - **Macros**
- **Declarações Globais**
  - **Funções**
  - **Variáveis**
- **Definição das Funções**

```
main ()
{ /* begin */
} /* end */
```

Estrutura de um programa em C

```
/* Primeiro Programa em C */
#include <stdio.h>

main()
{
printf("Meu primeiro programa em C\n");
}
```

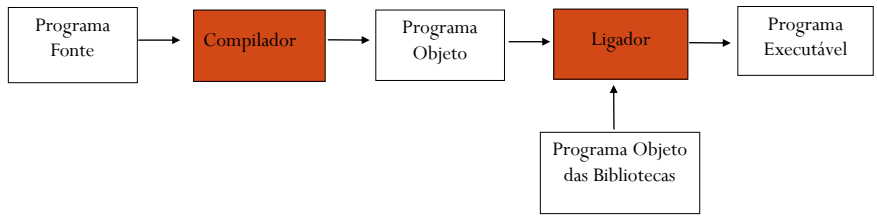
Estrutura de um programa em C

```
/* Primeiro Programa em C */ comentários
#include <stdio.h> /*biblioteca de E/S */

main() /*função principal – inicio do programa*/
{   /*marca início da função*/
    printf("Meu primeiro programa em C\n");
  /*função para escrever na tela*/
} /*marca fim da função*/
```

Processo de Compilação em C

- O processo de compilação, na verdade, se dá em duas etapas:
  - Fase de tradução: programa-fonte é transformado em um programa-objeto.
  - Fase de ligação: junta o programa-objeto às instruções necessárias das bibliotecas para produzir o programa executável.



Bibliotecas

- Conjunto de funções que permitem realizar tarefas específicas.
- Biblioteca padrão C - ANSI - funções básicas.
- As primeiras linhas do programa indicam as bibliotecas utilizadas

#include "minha_biblioteca.h" ou
#include <minha_biblioteca.h>

Dicas

- Termine todas as linhas com **;**
- Sempre salve o programa antes de compilar
- Sempre compile o programa antes de executar
- Verifique também a linha anterior, que pode ser a responsável pelo erro, especialmente se faltar o **;**
- Use comentários, iniciados por //

Declarações

- Declaram as variáveis e seus tipos
- Os nomes das variáveis devem conter apenas letras, dígitos e o símbolo _
- Os principais tipos são: **int**, **float**, **double** e **char**
- Exemplos

```
int n;
int quantidade_valores;
float x, y, somaValores;
char sexo;
char nome[40];
```

C diferencia letras maiúsculas de minúsculas!

int n, N;
n é diferente de N!

Comando de atribuição

- Atribui o valor da direita à variável da esquerda
- O valor pode ser:
  - uma *constante*,
  - uma variável ou
  - uma expressão
- Exemplos

```
x = 4;  --> lemos x recebe 4
y = x + 2;
y = y + 4;
valor = 2.5;
sexo = 'F'
```

12

Entrada e Saída de Dados

Entrada de Dados

- Função **scanf**

```
scanf ("formatos", &var1, &var2,...)
```

Exemplos:

```
int i, j;
float x;
char c;
scanf("%d", &i);
scanf("%d %f", &j, &x);
scanf("%c", &c);
scanf("%s", nome);
```

%d	inteiro
%f	float
%lf	double
%c	char
%s	palavra

Operadores Matemáticos

Operador	Exemplo	Comentário
+	`x + y`	Soma x e y
-	`x - y`	Subtrai y de x
*	`x * y`	Multiplica x e y
/	`x / y`	Divide x por y
%	`x % y`	Resto da divisão de x por y
++	`x++`	Incrementa em 1 o valor de x
--	`x--`	Decrementa em 1 o valor de x

15

Saída de Dados

- Função **printf**

```
printf ("formatos", var1, var2,...)
```

Exemplos:

```
int i, j;
float x;
char c;
printf("%d", i);
printf("%d, %f", j, x);
printf("%c", c);
printf("%s", nome);
```

%d	inteiro
%f	float
%lf	double
%c	char
%s	palavra

16

Operadores de Atribuição

Operador	Exemplo	Comentário
=	x = y	Atribui o valor de y a x
+=	x += y	Equivale a x = x + y
-=	x -= y	Equivale a x = x – y
*=	x *= y	Equivale a x = x * y
/=	x /= y	Equivale a x = x / y
%=	x %= y	Equivale a x = x % y

17

Funções Matemáticas

Função	Exemplo	Comentário
ceil	ceil(x)	Arredonda o número real para cima; ceil(3.2) é 4
cos	cos(x)	Cosseno de x (x em radianos)
exp	exp(x)	**e** elevado à potencia x
fabs	fabs(x)	Valor absoluto de x
floor	floor(x)	Arredonda o número deal para baixo; floor(3.2) é 3
log	log(x)	Logaritmo natural de x
log10	log10(x)	Logaritmo decimal de x
pow	pow(x, y)	Calcula x elevado à potência y
sin	sin(x)	Seno de x
sqrt	sqrt(x)	Raiz quadrada de x
tan	tan(x)	Tangente de x

18

```
#include <math.h>
```

Operadores Relacionais

Operador	Exemplo	Comentário
==	`x == y`	O conteúdo de x é igual ao de y
!=	`x != y`	O conteúdo de x é diferente do de y
<=	`x <= y`	O conteúdo de x é menor ou igual ao de y
>=	`x >= y`	O conteúdo de x é maior ou igual ao de y
<	`x < y`	O conteúdo de x é menor que o de y
>	`x > y`	O conteúdo de x é maior que o de y

As expressões relacionais em C retornam :
- **1 se verdadeiro e;**
- **0 se falso.**

19

Operadores Lógicos

- **&&** (E lógico): retorna verdadeiro se ambos os operandos são verdadeiros e falso nos demais casos.
  Exemplo: if(a>2 && b<3).

- || (OU lógico): retorna verdadeiro se um ou ambos os operandos são verdadeiros e falso se ambos são falsos.
  Exemplo: if(a>1 || b<2).

- **!** (NÃO lógico): usada com apenas um operando. Retorna verdadeiro se o operando é falso e vice-versa.
  Exemplo: if(!var).

20

Funções em C

Introdução às funções

Exercício: Implemente um programa que apresenta a seguinte saída:

********** ← escrever 20 asteriscos

Números entre 1 e 5

****

1

2

3

4

5

****

Introdução às funções

```
#include<stdio.h>

main()
{
        int i;

        for (i=1; i<=20; i++)
                putchar('*');
        putchar('\n');

        puts("Números entre 1 e 5");

        for (i=1; i<=20; i++)
                putchar('*');
        putchar('\n');

        for (i=1; i<=5; i++)
                printf("%d\n",i);

        for (i=1; i<=20; i++)
                putchar('*');
        putchar('\n');
}
```

Código repetido

23

Solução???

Tipo de funções

- Funções Pré-definidas pela linguagem (funções de biblioteca)
- Funções Definidas pelo programador

25

Funções definidas pelo usuário

- Reduzir a complexidade de um programa
- Evita-se a repetição de código ao longo do programa
- Precisa ter um nome único
- A localização da função é indiferente desde que coloque o protótipo (cabeçalho) da função antes da função main()

26

Funções

SINTAXE

Especificação de_tipo nome_da_função (lista de parâmetros)

{

corpo da função

}

- **Especificacação_de_tipo - especifica o tipo de valor que o comando return da função devolve, podendo ser qualquer tipo válido.**
- **nome_da_função - é um identificador escolhido pelo programador que não se deve repetir.**
- **lista de parâmetros - é uma lista de nomes de variáveis separadas por virgulas e seus tipo, que recebem os valores dos argumentos quando a função é chamada.**

27

Funções

Resolução do programa anterior usando funções

```
#include<stdio.h>
linha()
{
  int i;
  for (i=1; i<=20; i++)
       putchar('*');
  putchar('\n');
}
main()
{ int i;
  linha();
  puts("Números entre 1 e 5");
   linha();
  for (i=1; i<=5; i++)
       printf("%d\n",i);
  linha();
}
```

28

Características de uma função

- Cada função tem que ter um nome único, o qual serve para a sua invocação;
- Uma função pode ser invocada a partir de outras funções;
- Uma função deve realizar uma única tarefa bem definida;
- Uma função deve comportar-se como uma caixa preta. Não interessa como funciona, o que interessa é que o resultado final seja o esperado, sem efeitos colaterais;

29

Características de uma função

- O código de uma função deve ser o mais independente possível do resto do programa, e deve ser tão genérico quanto possível, para poder ser reutilizado em outros projetos.
- Uma função pode receber parâmetros que alteram o seu comportamento de forma a adaptar-se facilmente a situações distintas;
- Uma função pode retornar, para a entidade que a invocou, um valor como resultado da sua execução.

30

Funções - notas

- se nenhum tipo é especificado, o compilador assume que a função devolve um resultado inteiro (int)
- uma função pode não ter parâmetros e neste caso a lista de parâmetros é vazia.
- todos os parâmetros da função devem incluir o tipo e o nome da variável.

31

Passagem de Parâmetros

- Passagem de parâmetros por valor - uma cópia do argumento é passada para a função
- O parâmetro comporta-se como uma variável local

```
.....
printf("Introduza o valor de x e N\n\n");
scanf("%d%d",&x,&n);
Y = 2* soma(x , n)/ soma (x , n);
.....

int soma(int a, int b)
{
   return a+b;
}
```

argumentos

parâmetros

32

Passagem de Parâmetros

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

char minusculo (char ch);

main ( )
{
   printf (" %c", minusculo ('A') );

}

char minusculo (char ch)
{
  if (( ch >= 'A') &&  (ch <= 'Z'))
     return (ch + 'a'- 'A');
  else
       return (ch);
}
```

33

Passando vários argumentos

Ex 1:

```
float area_retangulo  (float
   largura, float  altura)
{
   return (largura * altura);
}
```

Ex 2:

```
float potencia (float  base,  int  expoente)
{
  int  i;   float  resultado = 1;
  if (expoente == 0)
    return 1;
  for (i = 1; i <= expoente; i++)
     resultado *= base
  return  resultado;
}
```

34

Procedimentos

- "Funções" que não retornam valores → retornam *void*

```
#include <stdio.h>
void desenha();
main ( )
{
    desenha ( );
    puts ("\n1ª função");
    desenha ( );
}
void   desenha( )
{
   int i;
   for (i = 0;  i < = 10;  i++)
     printf ("*");
}
```

Output

```
**********
1ª função
**********
```

Funções

```
#include <stdio.h>
int   fatorial (int);
int   fatorial (int  n)
{
   int  i, resultado = 1;
   for ( i = 1;  i <= n;  i ++)
     resultado *=  i;
   return  resultado;
}
main ( )
{
   printf (" o fatorial de 4 = %d", fatorial(4) );
   printf (" o fatorial de 3 = %d", fatorial(3) );
}
```

Retornam valores

Variáveis locais

- Variáveis declaradas dentro de uma função são denominadas locais e apenas podem ser usadas dentro do próprio bloco
- São criadas apenas na entrada do bloco e destruídas na saída (automáticas)

Variáveis Locais

```
void desenha ( )
  {
     int  i, j;
       .  .  .
  }
main ( )
  {
      int a;
      desenha();
      a = i;
      erro
      .  .  .
  }
```

i, j em desenha são variáveis diferentes de i, j em calcula.

```
void desenha ( )
{
   int i, j;
     .  .  .
     .  .  .
}
void calcula ( )
{
    int i, j;
    .  .  .
    .  .  .
}
```

Variáveis Globais

- Variável que é declarada externamente podem ser usadas por qualquer função

```
#include <stdio.h>

int i;
main ( )
{
   .........
  .........
desenha ( );
calcula ( );
}
```

```
void desenha ( )
{
     int j;
     i = 0;
     .  .  .
}
 void calcula ( )
{
      int m;
      i = 5;
      .  .  .
}
```

Comando Return

- Permite a atribuição de uma expressão a uma função e

força o retorno imediato ao ponto de chamada da função.

```
#include <stdio.h>
main ( )
{
 char letra;
 printf ("Letra minúscula");
 letra = minúsculo ( );
 if (letra  == 'a')
   printf ("ok");
}
```

```
char  minusculo ( )
{
   char ch;

   scanf("%c", ch);
   if  ( (ch >= 'A')  &&   (ch <= 'Z'))
       return (ch + 'a' - 'A');
   else
       return (ch);
}
```

Compilador GCC

- Compilador GCC
- GNU Compiler Collection
- Conjunto de compiladores de linguagens de programação produzido pelo projecto GNU
- É software livre distribuído pela Free Software Foundation (FSF) sob os termos da GNU GPL, e é um componente-chave do conjunto de ferramentas GNU
- É o compilador padrão para sistemas operacionaisUNIX e Linux e certos sistemas operacionais derivados tais como o Mac OS X
- Escrito por Richard Stallman em 1987

Usando o GCC

- Forma simples de compilação
- **gcc nomedoarquivo.c -o nomedoexecutavel**
- O comando acima gera um arquivo executável a partir da compilação e linkedição do programa em C (caso não haja erros!!)
- Para executar o programa basta colocar o seguinte comando em um terminal shell:
- **./nomedoexecutavel**
- OBS: Se o nome do executável não for informado o default é a.out