Projeto de Sistemas Embarcados Microcontrolados
Departamento de Engenharia Elétrica
Universidade de Brasília

Introdução à Linguagem C

Prof. Geovany A. Borges
gaborges@ene.unb.br

Atenção

A parte inicial deste material foi visto em sala-de-aula. Ela não está nestas transparências, sendo que compreendia os seguintes tópicos:

- Estrutura básica de um programa
- Tipos devariáveis
- Constantes
- Declarações
- Operadores
- Controle de fluxo: if-else, else-if, switch

Controle de fluxo

- while
  - Sintaxe:

```
while(condição){
    comandos
}
```

  - Exemplo:

```
int tempo_ms = 100;

while(tempo_ms-- > 0){
    delay_1ms();
}
```

Controle de fluxo

- do-while
  - Sintaxe:

```
do{
    comandos
} while(condição);
```

  - Exemplo:

```
int tempo_ms = 100;

do{
    delay_1ms();
} while(--tempo_ms > 0);
```

Controle de fluxo

- do-while
  - Exemplo:

```
char s[10];
unsigned int n;
int i = 0;

n = 2500;
do{
    s[i++] = n%10 + '0';
} while((n/=10) > 0);
s[i] = '\0';
```

Controle de fluxo

- for
  - Sintaxe:

```
for(expressão1;condição;expressão2){
    comandos
}
```

  - Exemplos:

```
for(i=0; i<N ;++i){...}
for(i=0,x=0.5; i<N ;++i){...}
for(; a < b;++a){...}
for(;;++a){...}
for(;;){...}
```

Controle de fluxo

- break
  - Saída do laço imediatamente anterior
  - Exemplo:

```
int x;

x = 100;
while(x>=0){
        x = (a * x + b) % 4;
        if (x==2) break;
}
```

Controle de fluxo

- continue
  - Continuação do laço imediatamente anterior
  - Exemplo:

```
int i,N = 20,a[20];
.
Comandos que definem os elementos de a[].
.
for(i=0; i<N; ++i){
        if (a[i]>=0) continue;
        a[i] = -a[i];
}
```

Controle de fluxo

- goto
  - Provoca desvios não condicionados
  - Exemplo:

```
if(condição) goto vem_para_ca;
.
.
.
vem_para_ca:
.
.
.
```

Funções

- Definição
- Sintaxe:

```
tipo_de_saída nome(lista variaveis de entrada)
{
 comandos
 return(variavel_de_saida)
}
```

- Exemplo:

```
float multiplique(float a, float b)
{
  return(a*b);
}
```

Funções

- Exemplo:

```
void delay_ms(unsigned int time_ms)
{
  do{
    delay_1ms();
  } while(-- time_ms > 0);
}
```

- Como retornar mais de um resultado?
  - Uso de variáveis globais
  - Uso de ponteiros

Funções

- Recursividade:

```
int fatorial(int X)
{
  if (X==0) return(1);
  return(X*fatorial(X-1));
}
```

Problema com a pilha se X for muito grande.

Funções

- Variáveis globais em funções

```
long int x;

long int funcao(int a, int b)
{
  extern long int x;
  long int N = 17597;

  x = (a*x+b)%N;
  return(x);
}
```

Funções

- Variáveis estáticas em funções

```
long int funcao(int a, int b)
{
  static long int x = 200;
  long int N = 17597;

  x = (a*x+b)%N;
 return(x);
}
```

Ponteiros

- Variável que contém um endereço de memória
  - Pode ser usado para apontar para variáveis
  - Pode ser usado para apontar para funções
- Sintaxe de definição:

```
tipo *nome_do_ponteiro;
```

- Exemplos:

```
int *px;
float *py;
void *pv;
```

Ponteiros

- Exemplos de uso

```
int x = -238, *pint = NULL;


pint = &x;
(*pint)++; //x = -237, pint = 200h
++pint;    //x = -237, pint = 202h
```

Mapa da memória

Variável	Endereço	Conteúdo
x	200h	-238
pint	320h	200h

Ponteiros

- Exemplos de uso

```
char c[] = "teste", *pchar = NULL;

pchar = &c[0]; // ou pchar = c;

*(pchar+2)++; // c[2] = 't'
++pchar;      // pchar = 131h
```

Mapa da memória

Variável	Endereço	Conteúdo
c	130h	't'
	131h	'e'
	132h	's'
	133h	't'
	134h	'e'
	135h	'\0'
pchar	20h	130h

Ponteiros

- Exemplos de uso

```
int x[] = {10,34,40}, *pint= NULL;

pint = &x[0]; // ou pint = x;

*(pint+=2) = 5; // x[2] = 5, pint = 262h
++pint;    // pint = 264h (cuidado aqui!)
```

Mapa da memória

Variável	Endereço	Conteúdo
x	25Eh	10
	260h	34
	262h	40
pint	65h	25Eh

Ponteiros

- Uso de ponteiros para passagem de argumentos

```
void main (void)
{
          double a, b, c = 2.0, d = -1.0;

          if(!cartesian_to_polar(&a, &b, &c, &d)){
                    printf("\n Erro");
          }
}

unsigned char cartesian_to_polar(double *prho, double *palpha, double *px, double *py)
{
          if(*px != 0) *palpha = atan((*py)/(*px));
          else return 0;
          *prho = sqrt((*px)*(*px) + (*py)*(*py));
          return 1;
}
```

Ponteiros

- Ponteiros para funções

```
double norma1(double a, double b){ return(fabs(a)+fabs(b)); }
double norma2(double a, double b){ return(sqrt(a*a+b*b)); }

void main (void)
{
        double (*norma)(double a, double b);
        double x;

        norma = norma1;
        x = norma(4.0,-3.0); // x = 7
        norma = norma2;
        x = norma(4.0,-3.0); // x = 5
}
```

Ponteiros

- Ponteiros para funções (1/2)

```
typedef void (*PMSG_HANDLER)(void);
unsigned char buffer[20];

#define NUM_OF_HANDLERS     10
PMSG_HANDLER pmsg_handlers[NUM_OF_HANDLERS];

void protocol_init(void)
{
          int n;
          for(n=0;n<NUM_OF_HANDLERS;++n) pmsg_handlers[n] = NULL;
}

void protocol_register(PMSG_HANDLER phandler, int code)
{
          pmsg_handlers[code] = phandler;
}
```

Ponteiros

- Ponteiros para funções (2/2)

```
void protocol_callhandler(int code)
{
        if(pmsg_handlers[code] != NULL) pmsg_handlers[code]();
}

void write_temperature(void) {       temperature = *((int *)(buffer)); }

void main (void)
{
        protocol_register(write_temperature, 0x01);
}
```

Ponteiros

- Ponteiros para ponteiros
  - Sintaxe da declaração: `tipo **nome_do_ponteiro`
  - Uso quando se deseja alterar o conteúdo do ponteiro:

```
int x = 10, y = 20;

void funcao(int **ppvar)
{
        *ppvar = &y; // coloca o ponteiro original apontando para y
        **ppvar = 25; // afeta y
}

void main (void)
{
        int *pint;
        pint = &x; // pint aponta para x, então *pint = 10.
        funcao(&pint); // pint aponta para y, então *pint = 25.
}
```

Estruturas

Agrupamento de variáveis/funções em uma única estrutura, introduzindo assim o conceito de *encapsulamento*.

- Exemplo:

```
struct {
  int dia;
  int mes;
  int ano;
} data;

void main (void)
{
  // entrando a data de 26/06/2007
  data.dia = 26;
  data.mes = 06;
  data.ano = 2007;
}
```

Estruturas

- Exemplo:

```
struct {
  int leitura_do_sensor;
  double tempo_s;
} dados[1000];

void main (void)
{
  for(int n=0;n<1000;++n){
        delay(10); //aguarda 10s;
        dados[n].leitura_do_sensor = leitura();
        dados[n].tempo_s = n*10;
  }
}
```

Estruturas

- Muito útil se encapsular dados em um tipo:

```
typedef struct {
  unsigned char pixel[120][160];
  int formato;
} IMAGE, *PIMAGE;

void processar_imagem(PIMAGE pImage)
{
  pImage->pixel[0][0] = 0;
}

void main (void)
{
  IMAGE Imagem;
  processar_imagem(&Imagem);
}
```

Alocação dinâmica de memória

Em alguns casos não se conhece, com antecedência, o tamanho de uma variável. Nestes casos, se faz necessário lançar mão da alocação dinâmica.

- Exemplo:

```
#include <stdlib.h>

void main (void)
{
    int *pbuffer;

    // Aloca memória
    pbuffer = (int *) malloc(10 * sizeof(int));
    // Faz uso da memória alocada
    pbuffer[2] = 35;
    // Libera a memória alocada
    free(pbuffer);
}
```