UFSC / CTC / INE
Disciplina: Paradigmas de Programação

CCO: INE5416 / SIN:INE5636

Prof. Dr. João Dovicchi*

Aula Prática 2 - Tipos de argumentos

Nesta aula, vamos testar algumas das colocações da aula teórica referentes aos tipos de argumentos em uma linguagem. A linguagem procedural (ansi C) apresenta algumas facilidades para se compreender como a tipagem ocorre em nível mais baixo.

Roteiro 1

Observe como estão organizadas as variáveis em C na memória:

Listagem 1: Programa exemplo de tipos em C.

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

int main(){
  printf("(short int): %li\n", sizeof(short int));
  printf("(int): %li\n", sizeof(int));
  printf("(uint): %li\n", sizeof(unsigned int));
  printf("(long int): %li\n", sizeof(long int));
  printf("(ulong int): %li\n", sizeof(unsigned long int));
  printf("(float): %li\n", sizeof(float));
  printf("(double): %li\n", sizeof(double));
  printf("(long double): %li\n", sizeof(long double));
  printf("(char): %li\n", sizeof(char));
  return 0;
}
```

*http://www.inf.ufsc.br/~dovicchi --- dovicchi@inf.ufsc.br

O programa pode ser baixado de `http://www.inf.ufsc.br/~dovicchi/paradigma/tipos.c`. Baixe o programa e compi-le o e execute-o:

```
$ wget http://www.inf.ufsc.br/~dovicchi/paradigma/tipos.c .
$ gcc -O2 -Wall -o tipos tipos.c
$ ./tipos
```

Observe o resultado.

Roteiro 2

Use os programas abaixo para observar como os conteúdos estão organizados na memória. O primeiro programa (listagem 2) aloca a 2 bytes para um `short int` e lê apenas um byte como um `char` que aponta para o endereço do `short int`.

Listagem 2: Programa bytes.c de alocação em C.

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

int main(){
  short int x=1372;
  char a;
  a=*(&x);
  printf("a= %c\n", a);
  printf("cont a: %x\n", a);
  printf("cont end x: %X\n", *(&x));
  exit (0);
}
```

Baixe, compile e execute o programa. Observe o resultado.

```
$ wget http://www.inf.ufsc.br/~dovicchi/paradigma/bytes.c .
$ gcc -O2 -Wall -o bytes bytes.c
$ ./bytes
```

O outro programa (listagem 3) faz o mesmo, tentando ler o primeiro byte, no endereço de um `double` (8 bytes).

Listagem 3: Programa double_pi.c de alocação em C.

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

int main(){
  double pi;
  char ch;

  pi = 4* atan(1);
  ch = *(char *) &pi;

  printf("pi=%1.10f\tch=%c\n", pi, ch);
  return 0;
}
```

Baixe, compile e observe o resultado.

```
$ wget http://www.inf.ufsc.br/~dovicchi/paradigma/double_pi.c .
$ gcc -O2 -Wall -o double_pi double_pi.c
$ ./double_pi
```

O programa seguinte (listagem 4) aloca a memória para um `int`, faz um *casting* para `float`, tentando ler o conteúdo, no endereço do `int`.

Listagem 4: Programa int_float.c de alocação em C.

```
#include <stdio.h>

int main(){
  int i = 37;
  float f = i;
  float f2 = *(float *) &i;

  printf("%f\n", f);
  printf("%1.80f\n", f2);
  return 0;
}
```

Baixe, compile e observe o resultado.

```
$ wget http://www.inf.ufsc.br/~dovicchi/paradigma/int_float.c .
$ gcc -O2 -Wall -o int_float int_float.c
$ ./int_float
```

Roteiro 3

Faça um pequeno relatório, anotando os resultados e discutindo o que aconteceu em cada execução e guarde para enviar no final do semestre.