FACAMP
FACULDADES DE CAMPINAS

Estruturas de Dados

Aula 1 – Revisão dos tipos numéricos de dados, operações aritméticas, relacionais e lógicas

1º semestre de 2020
Prof. José Martins Jr

Tipos básicos e seus tamanhos

- C, como outras linguagens de programação de alto nível, define tipos básicos de dados para representar números e caracteres

- Existem poucos tipos básicos em C

char	um byte, utilizado para representar um caractere (ASCII)
int	32 bits (ou 16), representa um número inteiro
float	32 bits, utilizado para ponto flutuante com precisão simples
double	64 bits, utilizado para ponto flutuante com precisão dupla

Inteiros e caracteres

- O tipo inteiro pode ter variações de tamanho

  short (ou short int) inteiro com tamanho mínimo de 16 bits

  long (ou long int) inteiro com tamanho mínimo de 32 bits

- Os tipos inteiro e char ainda podem ter ou não sinal

  signed inteiro, ou char, com sinal

  – Ex.: **signed char** – faixa de valores de -128 a 127

  unsigned inteiro, ou char, sem sinal

  – Ex.: **unsigned char** – vai de 0 a 255

Obtendo os tamanhos

- O operador **sizeof** retorna o número de bytes para um tipo fornecido (ou para o tipo da variável cujo nome foi fornecido)
  - Exemplo: exibir o número de bytes do tipo char

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

int main(int argc, char** argv) {
    printf("Tamanho do tipo char: %d\n", sizeof(char));
    return(EXIT_SUCCESS);
}
```

Declaração de variáveis

- Variáveis guardam a referência (endereço na memória) de campos de dados
- Para a correta interpretação do campo, a variável deve ser associada (declarada) a um tipo
  - Exemplos: declaração de duas variáveis de nomes i e c, e de respectivos tipos int e char

```
int i;
char c;
```

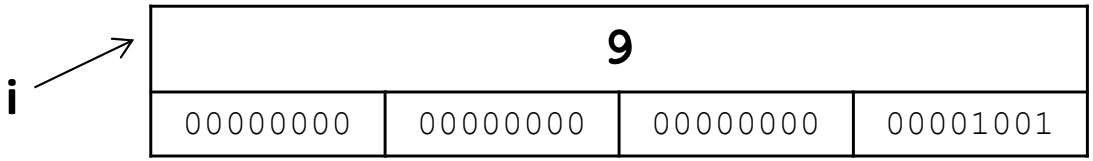
- Serão reservadas quantidades de memória do tamanho de cada tipo declarado
- O endereço do primeiro byte (referência) de cada área na memória é associado ao nome da variável

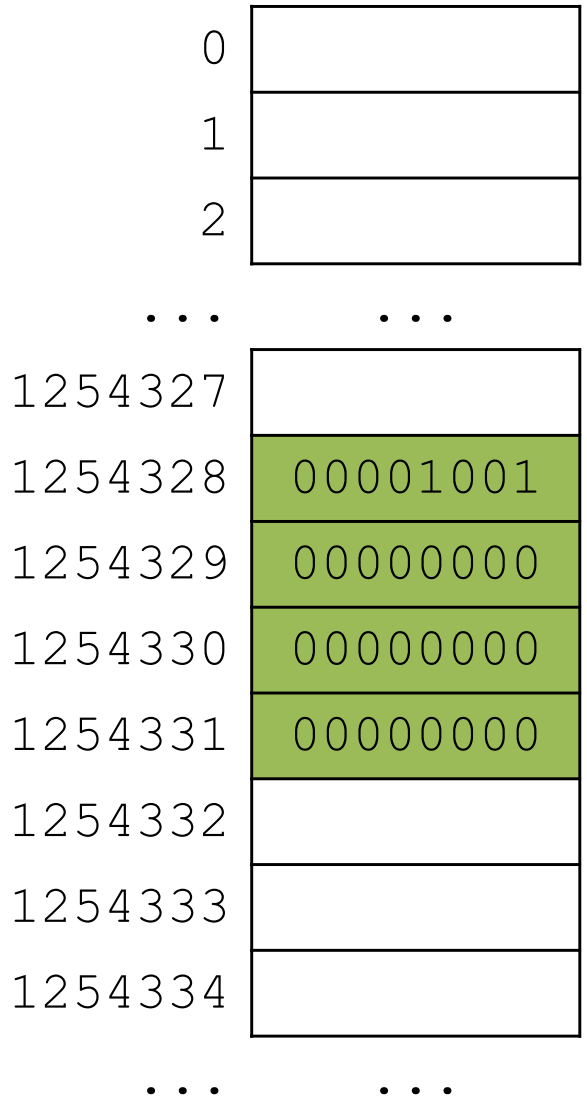
Alocação de variável na memória

- Exemplo em C

```
int i;
i = 9;
```

  - Endereço da variável **i**, neste exemplo
    - 1254328 (do 1º byte)
  - São alocados 4 bytes (32 bits)
    - Tamanho deste tipo na plataforma atual
  - Para facilitar, desenha-se na horizontal
    - **i** é o nome de uma variável de tamanho inteiro (4 bytes) alocada no endereço 1254328, que armazena o valor 9





Conversão entre tipos numéricos

- A conversão entre tipos numéricos diferentes pode ser realizada diretamente se for atribuído um tipo menor para um maior
  - Ex.:
```
int i = 10;
float f = i; //f armazenará 10.000000
```
- Já o contrário (atribuição direta de um tipo maior para um menor) pode ser proibido por alguns compiladores
- Nesse caso, pode-se aplicar o operador de conversão (**casting**), que permitirá a atribuição com perda de precisão
  - Ex.:
```
float f = 10/3.0f;
int i = (int) f; //i armazenará 3
```

Cuidados na conversão

- Operações aritméticas entre diferentes tipos numéricos
  - É reservada memória equivalente ao maior tipo envolvido para armazenar resultados parciais e final
  - Essa particularidade, comum a diversos compiladores e linguagens, é a causa de muitos erros de programação
  - Pode levar a estouros ou mesmo perda de precisão
  - Exemplos
    - Multiplicação int por double: resultado será armazenado em um espaço na memória do tipo double
    - Divisão entre dois inteiros: resultado será armazenado em um espaço do tipo inteiro (mesmo que não seja exato)

Operações aritméticas

- Os operadores aritméticos são

  `+` `-` `*` `/` adição, subtração, multiplicação e divisão

  – A divisão é inteira (se ambos os operadores forem inteiros) ou ponto flutuante (caso contrário)
    - Exs.: `15 / 4 = 3` **e** `15 / 4.0 = 3.75`

  `%` indica o resto (módulo) de uma divisão inteira
    - Ex.: `20 % 6 = 2`

  `++` `--` Incremento e decremento de uma unidade

  – Sufixados: em expressões, são realizados depois da operação

  ```
  i++; i--;
  ```
    - Ex.:
  ```
  int i = 2; int j = 10 * i++; //j será igual a 20
  ```

  – Prefixados: em expressões, são realizados antes da operação

  ```
  ++i; --i;
  ```
    - Ex.:
  ```
  int i = 2; int j = 10 * ++i; //j será igual a 30
  ```

Operadores relacionais

==	igual
!=	diferente
<	menor
>	maior
<=	menor ou igual
>=	maior ou igual

Operadores lógicos e de bit

- Curto-circuito
  - Se o 1° termo for false em && ou true em ||, o 2° não será calculado

    `&&` and

    `||` or

- Operadores bitwise (operam sobre bits)

    `&` and

    `|` or

    `^` xor

    `~` not (complemento de um – unário)

    `>>` deslocamento à direita

    `<<` deslocamento à esquerda

Formas comprimidas

`+=`	`&=`
`-=`	`|=`
`*=`	`^=`
`/=`	`<<=`
`%=`	`>>=`

– Ex.: soma 4 ao valor anterior de x

```
x += 4;
```

Testes e laços de controle

- A sintaxe é idêntica à de Java (já estudada)
  - Testes: if-else
  - Repetição condicionada: while e do-while
  - Repetição por número conhecido de vezes: for

- A diferença fica na seleção múltipla
  - O switch-case (mesma sintaxe) só aceita int ou char

Bibliografia

- KERNIGHAN, B. W., RITCHIE, D. M. C, a linguagem de programação: padrão ANSI. Rio de Janeiro: Campus, 1989. 289 p.
- LOOSEMORE, S.; STALLMAN, R. M. et al. The GNU C Library Reference Manual. Disponível no endereço: http://www.gnu.org/software/libc/manual/