Linguagem de Programação C++

Universidade Federal de Santa Catarina

Departamento de Engenharia Elétrica, CTC

Prof. Eduardo Augusto Bezerra

"Esse material foi adaptado a partir de um website não identificado."

Variáveis, tipos de dados, constantes, palavras reservadas

Identificadores

Um identificador válido é uma sequência de uma ou mais letras, dígitos ou caracteres "sublinhado" (_). Espaços em branco e caracteres acentuados não podem ser utilizados na formação de identificadores. É preciso tomar cuidado para não utilizar palavras reservadas do C++ como nome de identificador. Abaixo estão listadas as palavras reservadas, de acordo com o padrão ANSI-C++, que não podem ser utilizadas como identificadores em programas:

```
asm, auto, bool, break, case, catch, char, class, const, const_cast, continue, default, delete, do, double,
dynamic_cast, else, enum, explicit, extern, false, float, for, friend, goto, if, inline, int, long, mutable,
namespace, new, operator, private, protected, public, register, reinterpret_cast, return, short, signed, sizeof,
static, static_cast, struct, switch, template, this, throw, true, try, typedef, typeid, typename, union, unsigned,
using, virtual, void, volatile, wchar_t
```

ATENÇÃO!! C++ é "case sensitive", e identificadores definidos com caracteres maiúsculos e minúsculos não são equivalentes. Por exemplo, a variável RESULT é diferente de result, que por sua vez é diferente de Result.

Tipos de dados

Lembrando que em um byte é possível armazenar valores inteiros entre 0 e 255, em C++ existem diversos tipos de dados que possibilitam armazenar valores com diferentes tipos e tamanhos. Os tipos de dados fundamentais do C++ estão listados a seguir:

Tipo	Bytes	Descrição	Faixa de valores
char	1	Caracter ou inteiro de 8 bits.	**signed:** -128 a 127 **unsigned:** 0 a 255
short int	2	Inteiro de 16 bits.	**signed:** -32768 a 32767 **unsigned:** 0 a 65535
long int	8	Inteiro de 64 bits.	**signed:**-9223372036854775808 a 9223372036854775807 **unsigned:** 0 a 18446744073709551615
int	*	Inteiro. O tamanho depende do tamanho da palavra da arquitetura destino. Em sistemas 32 bits tais como diversas edições do Windows e Linux, o inteiro possui 4 bytes (32 bits).	Ver **short**, **long**
float	4	Número em ponto flutuante.	3.4e + / - 38 (7 digitos)
double	8	Número em ponto flutuante precisão dupla.	1.7e + / - 308 (15 digitos)
long double	10	Número em ponto flutuante precisão dupla "longo".	1.2e + / - 4932 (19 digits)
bool	1	Valor "booleano". Pode receber apenas os valores `true` ou `false`. Por ser um tipo adicionado recentemente ao ANSI-C++, pode ser que alguns compiladores ainda não aceitem.	**`true`** ou **`false`**
wchar_t	2	Caracteres "largos". Armazena caracteres internacionais de dois bytes. Por ser um tipo adicionado recentemente ao ANSI-C++, pode ser que alguns compiladores ainda não aceitem.	Caracteres extendidos

Obs. Valores das colunas "Bytes" e "Faixa de valores" podem variar dependendo do sistema destino. Esses são os valores mais comumente utilizados pela maioria dos compiladores C++ da atualidade.

Outros tipos de dados do C++ incluem os ponteiros e o parâmetro **void** que serão discutidos adiante.

Declaração de variáveis

Variáveis em C++ precisam ser declaradas antes de serem utilizadas:

```
int a;
int a, b, c;
float meuNumero;
```

Os tipos de dados inteiros (**char**, **short**, **long** e **int**) podem ser definidos com sinal ou sem sinal (**signed** ou **unsigned**), de acordo com a faixa de números que se deseja representar:

```
unsigned short numeroDeFilhos;
signed int meuSaldo;
```

Caso não seja utilizado explicitamente a palavra **signed** ou **unsigned** na frente do tipo, será assumido o tipo **signed.**

Caso seja utilizado apenas **signed** ou **unsigned** sem nenhum tipo associado, é assumido o tipo inteiro para a variável.

Exemplo de programa com declaração de variáveis:

```
// trabalhando com variaveis                                    4

#include <iostream>
using namespace std;

int main ()
{
  // declarando variaveis:
  int a, b;
  int resultado;

  // processamento:
  a = 5;
  b = 2;
  a = a + 1;
  resultado = a - b;

  // apresentacao dos resultados:
  cout << resultado;

  // terminando o programa:
  return 0;
}
```

Inicialização de variáveis

Ao ser declarada a variável não possui um valor. Caso seja desejado atribuir um valor inicial a uma variável, isso deve ser feito explicitamente:

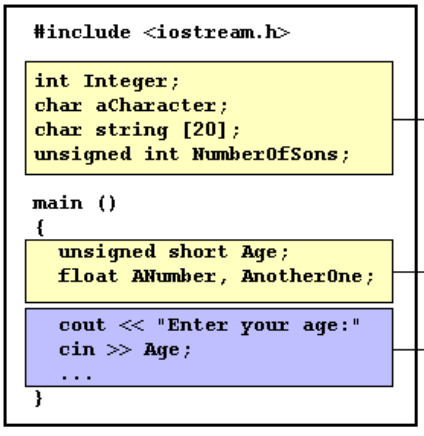
```
int a = 0;
```

que equivale a:

```
int a (0);
```

Uma diferença importante entre C e C++, é que em C++ as variáveis podem ser declaradas em qualquer lugar do código fonte.

Escopo das variáveis



Variáveis globais (**global variables**) podem ser referenciadas em qualquer ponto do programa. Variáveis locais (**local variables**) só podem ser acessadas no escopo em que forem definidas. No exemplo ao lado a variável Age, é válida apenas na função main, e não pode ser acessada em nenhuma outra função do programa.

Em C++ o escopo das variáveis é definido pelo bloco no qual a variável é declarada. Os limites do bloco são definidos por chaves {}. Se a declaração for no início de uma função, o escopo da variável será a função (qualquer parte da função). Se a declaração for no início de um laço (após a primeira abertura de chave), então o escopo será o laço em questão.

Adicionalmente, existe também o escopo externo, que leva a variável a ser visível não apenas no arquivo em que foi declarada, mas também em outros arquivos que forem “ligados” juntos.

Definindo constantes (*#define*)

```
#define PI 3.14159265
#define NEWLINE '\n'
#define WIDTH 100
```

Uma vez declaradas, as constantes podem ser utilizadas no código normalmente:

```
circulo = 2 * PI * r;
cout << NEWLINE;
```

Ao encontrar a diretiva **`#define`** o compilador substitui todas as ocorrências do símbolo definido (**`PI`**, **`NEWLINE`** ou **`WIDTH`**) pelo código associado (**`3.14159265`**, **`'\n'`** e **`100`**, respectivamente).

A diretiva `#define` não é uma instrução, e por essa razão não necessita o ponto-e-vírgula (`;`) no final da linha.

Declarando constantes (`const`)

Com o uso de **`const`** constantes podem ser definidas com um tipo de forma similar a variáveis:

```
const int largura = 100;
const char tab = '\t';
const zip = 12440;
```

Quando o tipo não é definido (como no último exemplo), o compilador assume o tipo inteiro.