Aula 16 – Laços (parte 3)

Norton T. Roman & Luciano A. Digiampietri

For

- Os `while` vistos tinham características em comum em suas variáveis de controle:

```
int main() {
  double area = 100;
  int tipo = ALVENARIA;

  printf("Material\tValor\n");
  while (tipo <= PLASTICO) {
    printf("%8i\t%9.2f\n", tipo,
        valorPiscina(area,tipo));
    tipo = tipo+1;
  }
  return 0;
}
```

For

- Os `while` vistos tinham características em comum em suas variáveis de controle:
  - Ambos variavam em passos constantes (de 1 em 1 ou de 50 em 50)

```
int main() {
  double area = 100;
  int tipo = ALVENARIA;

  printf("Material\tValor\n");
  while (tipo <= PLASTICO) {
    printf("%8i\t%9.2f\n", tipo,
        valorPiscina(area,tipo));
    tipo = tipo+1;
  }
  return 0;
}
```

For

- Não haveria um modo de deixar esse código mais enxuto?

```
int main() {
  double area = 100;
  int tipo = ALVENARIA;

  printf("Material\tValor\n");
  while (tipo <= PLASTICO) {
    printf("%8i\t%9.2f\n", tipo,
        valorPiscina(area,tipo));
    tipo = tipo+1;
  }
  return 0;
}
```

For

- Não haveria um modo de deixar esse código mais enxuto?
- Um modo de dizer "para tipo começando em 0, variando de 1 em 1, até 3, faça..."

```
int main() {
  double area = 100;
  int tipo = ALVENARIA;

  printf("Material\tValor\n");
  while (tipo <= PLASTICO) {
    printf("%8i\t%9.2f\n", tipo,
        valorPiscina(area,tipo));
    tipo = tipo+1;
  }
  return 0;
}
```

For

- Ou “para area começando em 50, variando de 50 em 50, até 200, faça...”

```
int main() {
  double area = 50;
  int tipo = ALVENARIA;

  printf("Área\tValor\n");
  while (area <= 200) {
    printf("%6.1f\t%9.2f\n",area,
        valorPiscina(area,tipo));
    area = area+50;
  }
  return 0;
}
```

For

- O laço `for`:

```
for (inicialização;
            condição;
          atualização) {
    comandos;
}
```

```
inicialização;
while (condição) {
  comandos;
  atualização;
}
```

For

- O laço `for`:

```
for (inicialização;
            condição;
          atualização) {
    comandos;
}
```

```
inicialização;
while (condição) {
  comandos;
  atualização;
}
```

- Primeiro, há a inicialização das variáveis de controle

For

- O laço `for`:

```
for (inicialização;
            condição;
          atualização) {
    comandos;
}
```

```
inicialização;
while (condição) {
  comandos;
  atualização;
}
```

- Primeiro, há a inicialização das variáveis de controle
  - Esse passo é executado uma única vez

For

```
for (inicialização;
            condição;
          atualização) {
    comandos;
}
```

```
inicialização;
while (condição) {
  comandos;
  atualização;
}
```

- Em seguida, a condição é testada

For

```
for (inicialização;
            condição;
          atualização) {
    comandos;
}
```

```
inicialização;
while (condição) {
  comandos;
  atualização;
}
```

- Em seguida, a condição é testada
- Se resultar verdadeira, os comandos do corpo do `for` são executados

For

```
for (inicialização;
            condição;
          atualização) {
    comandos;
}
```

```
inicialização;
while (condição) {
  comandos;
  atualização;
}
```

- Ao final do corpo, é executada a atualização

For

```
for (inicialização;
            condição;
          atualização) {
    comandos;
}
```

```
inicialização;
while (condição) {
  comandos;
  atualização;
}
```

- Ao final do corpo, é executada a atualização
- Inicia-se o laço novamente, voltando ao teste da condição

For

```
for (inicialização;
            condição;
          atualização) {
    comandos;
}
```

```
inicialização;
while (condição) {
  comandos;
  atualização;
}
```

- Ao final do corpo, é executada a atualização
- Inicia-se o laço novamente, voltando ao teste da condição
- Se a condição for falsa, o corpo é ignorado

For

```
int main() {
  double area = 100;
  int tipo = ALVENARIA;
  printf("Material\tValor\n");
  while (tipo <= PLASTICO) {
    printf("%8i\t%9.2f\n", tipo,
        valorPiscina(area,tipo));
    tipo = tipo+1;
  }
  return 0;
}
```

```
int main() {
  double area = 100;

  printf("Material\tValor\n");
  int tipo;
  for(tipo = ALVENARIA;
                  tipo <= PLASTICO;
                   tipo = tipo+1) {
    printf("%8i\t%9.2f\n", tipo,
        valorPiscina(area,tipo));
  }
  return 0;
}
```

- São totalmente equivalentes: dependem da conveniência do programador

For

- Incremento de um em um não é só o que o `for` é capaz de fazer:

```
int main() {
  double area = 50;
  int tipo = ALVENARIA;
  printf("Área\tValor\n");
  while (area <= 200) {
    printf("%6.1f\t%9.2f\n",area,
        valorPiscina(area,tipo));
    area = area+50;
  }
  return 0;
}
```

```
int main() {
  double area;
  int tipo = ALVENARIA;

  printf("Área\tValor\n");
  for(area = 50; area <= 200;
                    area = area+50) {
    printf("%6.1f\t%9.2f\n",area,
        valorPiscina(area,tipo));
  }
  return 0;
}
```

For

- Qualquer expressão algébrica pode ser usada

```
int main() {
  double area;
  int tipo = ALVENARIA;

  printf("Área\tValor\n");
  for(area = 50; area <= 200;
                      area = area+50) {
    printf("%6.1f\t%9.2f\n",area,
        valorPiscina(area,tipo));
  }
  return 0;
}
```

For

- Qualquer expressão algébrica pode ser usada
- Até mesmo coisas como `area = 2*area + pow(area,3)`

```
int main() {
  double area;
  int tipo = ALVENARIA;

  printf("Área\tValor\n");
  for(area = 50; area <= 200;
                      area = area+50) {
    printf("%6.1f\t%9.2f\n",area,
        valorPiscina(area,tipo));
  }
  return 0;
}
```

For

- E não é apenas o int que pode ser usado como variável de controle
- Podemos também usar outros tipos – Cuidado com comparações em ponto flutuante!!!

```
int main() {
  int tipo = ALVENARIA;

  printf("Área\tValor\n");
  double area;
  for(area = 50; area <= 200;
                      area = area+50) {
    printf("%6.1f\t%9.2f\n",area,
        valorPiscina(area,tipo));
  }
  return 0;
}
```

For

- Da mesma forma, na condição qualquer expressão lógica ou relacional pode ser usada
- Ex:
  `(area <= 200)||(area == 300)`

```
int main() {
  double area;
  int tipo = ALVENARIA;

  printf("Área\tValor\n");
  for(area = 50; area <= 200;
                      area = area+50) {
    printf("%6.1f\t%9.2f\n",area,
        valorPiscina(area,tipo));
  }
  return 0;
}
```

For

- Também nada nos impede de fazer um decremento

```
int main() {
  double area;
  int tipo = ALVENARIA;

  printf("Área\tValor\n");
  for(area = 200; area >= 50;
                    area = area-50) {
    printf("%6.1f\t%9.2f\n",area,
        valorPiscina(area,tipo));
  }
  return 0;
}
```

For

- E o resultado seria apenas a inversão da tabela:

```
Área     Valor
 200.0   300000.00
 150.0   225000.00
 100.0   150000.00
  50.0    75000.00
```

```
int main() {
  double area;
  int tipo = ALVENARIA;

  printf("Área\tValor\n");
  for(area = 200; area >= 50;
                      area = area-50) {
    printf("%6.1f\t%9.2f\n",area,
        valorPiscina(area,tipo));
  }
  return 0;
}
```

For Aninhado

- Laços *for* podem também ser aninhados

```
int main() {
  double area = 50;
  int tipo;
  printf("Área\tTipo\tValor\n");
  while (area <= 200) {
    tipo = ALVENARIA;
    while (tipo <= PLASTICO) {
      printf("%6.1f\t%4i\t%9.2f\n",
       area, tipo,
        valorPiscina(area,tipo));
      tipo = tipo+1;
    }
    area = area+50;
  }
  return 0;
}
```

```
int main() {
  double area;
  int tipo;

  printf("Área\tTipo\tValor\n");
  for(area = 50; area <= 200;
                        area = area+50) {
    for(tipo = ALVENARIA;
                    tipo <= PLASTICO;
                        tipo = tipo+1) {
      printf("%6.1f\t%4i\t%9.2f\n",
       area, tipo,
        valorPiscina(area,tipo));
    }
  }
  return 0;
}
```

For Aninhado

- Laços *for* podem também ser aninhados

```
int main() {
  double area = 50;
  int tipo;
  printf("Área\tTipo\tValor\n");
  while (area <= 200) {
    tipo = ALVENARIA;
    while (tipo <= PLASTICO) {
      printf("%6.1f\t%4i\t%9.2f\n",
       area, tipo,
        valorPiscina(area,tipo));
      tipo = tipo+1;
    }
    area = area+50;
  }
  return 0;
}
```

```
int main() {
  double area;
  int tipo;

  printf("Área\tTipo\tValor\n");
  for(area = 50; area <= 200;
                        area = area+50) {
    for(tipo = ALVENARIA;
                    tipo <= PLASTICO;
                        tipo = tipo+1) {
      printf("%6.1f\t%4i\t%9.2f\n",
       area, tipo,
        valorPiscina(area,tipo));
    }
  }
  return 0;
}
```

- Podem ficar até mais fáceis de serem entendidos

For

- Embora a condição tenha que ser única

```
int a;
int b;
for(???; a<b; ???) {
  printf("a=%i\n",a);
  printf("b=%i\n",b);
}
```

For

- Embora a condição tenha que ser única
- Aceita múltiplas inicializações
  - Separadas por vírgula
  - Declaradas fora do `for`

```
int a;
int b;
for(a=1, b=4; a<b; ???) {
  printf("a=%i\n",a);
  printf("b=%i\n",b);
}
```

For

- Embora a condição tenha que ser única
- Aceita múltiplas inicializações
  - Separadas por vírgula
  - Declaradas fora do `for`
- E múltiplas atualizações
  - Também separadas por vírgula

```
int a;
int b;
for(a=1, b=4; a<b; a++,b--) {
  printf("a=%i\n",a);
  printf("b=%i\n",b);
}
```

For

- Embora a condição tenha que ser única
- Aceita múltiplas inicializações
  - Separadas por vírgula
  - Declaradas fora do `for`
- E múltiplas atualizações
  - Também separadas por vírgula
- `a++`? `b--`?

```
int a;
int b;
for(a=1, b=4; a<b; a++,b--) {
  printf("a=%i\n",a);
  printf("b=%i\n",b);
}
```

Expressões Contraídas

- São "atalhos" $\rightarrow$ Expressões contraídas

```
int a;
int b;
for(a=1, b=4; a<b; a++,b--) {
  printf("a=%i\n",a);
  printf("b=%i\n",b);
}
```

Expressões Contraídas

- São "atalhos" $\rightarrow$ Expressões contraídas
- Úteis para realizar a operação e armazenar o resultado na mesma variável

```
int a;
int b;
for(a=1, b=4; a<b; a++,b--) {
   printf("a=%i\n",a);
   printf("b=%i\n",b);
}
```

Expressões Contraídas

Expressão	*Contraída*
`x = x + 5`	`x += 5`
`x = x - 5`	`x -= 5`
`x = x * 5`	`x *= 5`
`x = x / 5`	`x /= 5`
`x = x % 5`	`x %= 5`

```
int a;
int b;
for(a=1, b=4; a<b; a++,b--) {
  printf("a=%i\n",a);
  printf("b=%i\n",b);
}
```

Expressões Contraídas

Expressão	*Contraída*
`x = x + 5`	`x += 5`
`x = x - 5`	`x -= 5`
`x = x * 5`	`x *= 5`
`x = x / 5`	`x /= 5`
`x = x % 5`	`x %= 5`

```
int a;
int b;
for(a=1, b=4; a<b; a++,b--) {
  printf("a=%i\n",a);
  printf("b=%i\n",b);
}
```

- E o ++?

Expressões Contraídas

Expressão	*Contraída*
`x = x + 5`	`x += 5`
`x = x - 5`	`x -= 5`
`x = x * 5`	`x *= 5`
`x = x / 5`	`x /= 5`
`x = x % 5`	`x %= 5`

```
int a;
int b;
for(a=1, b=4; a<b; a++,b--) {
  printf("a=%i\n",a);
  printf("b=%i\n",b);
}
```

- E o ++?
- `x++` é a expressão contraída para `x = x + 1`

Expressões Contraídas

- Tem duas formas: `x++` ou `++x`

```
int x = 2;
int y = 2;
x++;
++y;
printf("x = %i, y = %i\n",
        x, y);
```

Saída:

Expressões Contraídas

- Tem duas formas: `x++` ou `++x`

```
int x = 2;
int y = 2;
x++;
++y;
printf("x = %i, y = %i\n",
        x, y);
```

Saída:

```
x = 3, y = 3
```

Expressões Contraídas

- Tem duas formas: `x++` ou `++x`
- Usados isoladamente, tanto `++x` quanto `x++` correspondem a `x = x+1`

```
int x = 2;
int y = 2;
x++;
++y;
printf("x = %i, y = %i\n",
        x, y);
```

Saída:

```
x = 3, y = 3
```

Expressões Contraídas

- Mas coisas acontecem quando usados em conjunto com outros comandos...

```
int x = 2;
int y = 2;
printf("x = %i, y = %i\n",
        x++, ++y);
printf("x = %i, y = %i\n",
        x, y);
```

Saída:

```
x = 2, y = 3
x = 3, y = 3
```

Expressões Contraídas

- Mas coisas acontecem quando usados em conjunto com outros comandos...
- O que houve?

```
int x = 2;
int y = 2;
printf("x = %i, y = %i\n",
        x++, ++y);
printf("x = %i, y = %i\n",
        x, y);
```

Saída:

```
x = 2, y = 3
x = 3, y = 3
```

Expressões Contraídas

- Mas coisas acontecem quando usados em conjunto com outros comandos...
- O que houve?
- x++ é um pós-incremento

```
int x = 2;
int y = 2;
printf("x = %i, y = %i\n",
        x++, ++y);
printf("x = %i, y = %i\n",
        x, y);
```

Saída:

```
x = 2, y = 3
x = 3, y = 3
```

Expressões Contraídas

- Mas coisas acontecem quando usados em conjunto com outros comandos...
- O que houve?
- x++ é um pós-incremento

```
int x = 2;
int y = 2;
printf("x = %i, y = %i\n",
        x++, ++y);
printf("x = %i, y = %i\n",
        x, y);
```

Saída:

```
x = 2, y = 3
x = 3, y = 3
```

- Diz que o compilador deve usar o valor que está em x e só então incrementá-lo

Expressões Contraídas

- ++y é um pré-incremento

```
int x = 2;
int y = 2;
printf("x = %i, y = %i\n",
        x++, ++y);
printf("x = %i, y = %i\n",
        x, y);
```

Saída:

```
x = 2, y = 3
x = 3, y = 3
```

Expressões Contraídas

- ++y é um pré-incremento
- Diz que o compilador deve primeiro incrementar o valor de y, e só então usá-lo

```
int x = 2;
int y = 2;
printf("x = %i, y = %i\n",
        x++, ++y);
printf("x = %i, y = %i\n",
        x, y);
```

Saída:

```
x = 2, y = 3
x = 3, y = 3
```

Expressões Contraídas

- De forma semelhante ao ++, o -- decrementa, em vez de incrementar
- Também em suas duas formas: `x--` e `--x`

```
int x = 2;
int y = 2;
printf("x = %i, y = %i\n",
        x--, --y);
printf("x = %i, y = %i\n",
        x, y);
```

Saída:

Expressões Contraídas

- De forma semelhante ao ++, o -- decrementa, em vez de incrementar
- Também em suas duas formas: `x--` e `--x`

```
int x = 2;
int y = 2;
printf("x = %i, y = %i\n",
        x--, --y);
printf("x = %i, y = %i\n",
        x, y);
```

Saída:

```
x = 2, y = 1
x = 1, y = 1
```

Expressões Contraídas

Mais exemplos:

Código

```
int x = 2;
int y = x++;
printf("x = %i, y = %i\n", x, y);
int z = ++x;
printf("x = %i, z = %i\n", x, z);
```

Expressões Contraídas

Mais exemplos:

Código

```
int x = 2;
int y = x++;
printf("x = %i, y = %i\n", x, y);
int z = ++x;
printf("x = %i, z = %i\n", x, z);
```

Saída

```
x = 3, y = 2
x = 4, z = 4
```

y = x++ fará y conter 2, se x contiver 2 antes do ++

z = ++x fará z conter 4, se x contiver 3 antes do ++

Expressões Contraídas

Mais exemplos:

Código

```
int x = 2;
int y = x++;
printf("x = %i, y = %i\n", x, y);
int z = ++x;
printf("x = %i, z = %i\n", x, z);
```

Saída

```
x = 3, y = 2
x = 4, z = 4
```

y = x++ fará y conter 2, se x contiver 2 antes do ++

z = ++x fará z conter 4, se x contiver 3 antes do ++

Código

```
int x = 1;
int y = x++ + 4;
printf("x = %i, y = %i\n", x, y);
int z = ++x + 4;
printf("x = %i, z = %i\n", x, z);
```

Expressões Contraídas

Mais exemplos:

Código

```
int x = 2;
int y = x++;
printf("x = %i, y = %i\n", x, y);
int z = ++x;
printf("x = %i, z = %i\n", x, z);
```

Saída

```
x = 3, y = 2
x = 4, z = 4
```

y = x++ fará y conter 2, se x contiver 2 antes do ++

z = ++x fará z conter 4, se x contiver 3 antes do ++

Código

```
int x = 1;
int y = x++ + 4;
printf("x = %i, y = %i\n", x, y);
int z = ++x + 4;
printf("x = %i, z = %i\n", x, z);
```

Saída

```
x = 2, y = 5
x = 3, z = 7
```

Laços

- Considere o código ao lado:

```
int main() {
  int x = 1;
  for (; x<5; x++) {
    printf("%i ", x);
  }
  return 0;
}
```

Laços

- Considere o código ao lado:
- O que será impresso?

```
int main() {
  int x = 1;
  for (; x<5; x++) {
    printf("%i ", x);
  }
  return 0;
}
```

Laços

- Considere o código ao lado:
- O que será impresso?
  - `1 2 3 4`

```
int main() {
  int x = 1;
  for (; x<5; x++) {
    printf("%i ", x);
  }
  return 0;
}
```

Laços

- Considere o código ao lado:
- O que será impresso?
  - `1 2 3 4`
- A inicialização em um laço `for` é opcional

```
int main() {
  int x = 1;
  for (; x<5; x++) {
    printf("%i ", x);
  }
  return 0;
}
```

Laços

- Considere agora esse código:

```
int main() {
  int x = 1;
  for (; x<5;) {
    printf("%i ", x);
  }
  return 0;
}
```

Laços

- Considere agora esse código:
- O que será impresso?

```
int main() {
  int x = 1;
  for (; x<5;) {
    printf("%i ", x);
  }
  return 0;
}
```

Laços

- Considere agora esse código:
- O que será impresso?
  - `1 1 1 1 1 1 1 1 1...`

```
int main() {
  int x = 1;
  for (; x<5;) {
    printf("%i ", x);
  }
  return 0;
}
```

Laços

- Considere agora esse código:
- O que será impresso?
  - `1 1 1 1 1 1 1 1 1 1...`
- Laço infinito: a condição de parada nunca é satisfeita

```
int main() {
  int x = 1;
  for (; x<5;) {
    printf("%i ", x);
  }
  return 0;
}
```

Laços

- Também a atualização da variável de controle é opcional

```
int main() {
  int x = 1;
  for (; x<5;) {
    printf("%i ", x);
  }
  return 0;
}
```

Laços

- E esse código?

```
int main() {
  int x;
  for (x=1;;x++) {
    printf("%i ", x);
  }
  return 0;
}
```

Laços

- E esse código?
  - `1 2 3 4 5 6 7 8...`

```
int main() {
  int x;
  for (x=1;;x++) {
    printf("%i ", x);
  }
  return 0;
}
```

Laços

- E esse código?
  - `1 2 3 4 5 6 7 8...`
- De novo! Ninguém disse ao laço o que testar para parar

```
int main() {
  int x;
  for (x=1;;x++) {
    printf("%i ", x);
  }
  return 0;
}
```

Laços

- E esse código?
  - `1 2 3 4 5 6 7 8...`
- De novo! Ninguém disse ao laço o que testar para parar
- A condição de parada em um laço `for` também é opcional

```
int main() {
  int x;
  for (x=1;;x++) {
    printf("%i ", x);
  }
  return 0;
}
```

Laços

Em Suma:

- Inicialização, condição e atualização são opcionais
- A condição aceita qualquer expressão que resulte em verdadeiro ou falso (expressões lógicas e relacionais)
- Inicialização e atualização são apenas códigos rodados, respectivamente, antes da primeira iteração e ao fim de cada iteração do laço

Aula 16 – Laços (parte 3)

Norton T. Roman & Luciano A. Digiampietri