Paradigmas de Linguagens de Programação
Tarefa 02
Benedito Ferreira
Alfredo Gabriel de Sousa Oliveira

“Linguagens imperativas (como Pascal e Ada) têm variáveis, que são posições de memória nomeadas.”

Exemplo 1:

```c
#include <stdio.h>

int main(int argc, char const *argv[]) {

  int a = 500;
  printf("O valor %d está no endereço: %p\n", a, &a);

  return 0;
}
```

Saída:

```
O valor 500 está no endereço: 0x7fffd01e4314
```

Exemplo 2:

```c
#include <stdio.h>

int main(int argc, char const *argv[]) {

  char x[] = "Tudo bom?";
  printf("A string '%s' está no endereço: %p\n", x, &x);

  return 0;
}
```

Saída:

```
A string 'Tudo bom?' está no endereço: 0x7fff699df1be
```

Nesses dois exemplos, é possível observar que cada variável de fato está relacionado a um endereço de memória, visto que ao definir duas variáveis, quando pedimos seu endereço ao sistema, nos sera retornado um endereço diferente.

“[…] . As linguagens de programação geralmente restringem os valores que podem ser armazenados nas variáveis, tanto para garantir que os compiladores possam gerar código acurado para manipular esses valores quanto para evitar erros comuns de programação. As restrições geralmente se revestem da informação do tipo. ”

Exemplo 1:

```c
#include <stdio.h>

int main(int argc, char const *argv[]) {

  int a = 500;
  int b = 550;
  printf("A soma é %d\n", a+b);

  return 0;
}
```

Saída:

```
A soma é 1050
```

Exemplo 2:

```c
#include <stdio.h>

int main(int argc, char const *argv[]) {

  int a = 500;
  char b[] = "550";
  printf("A soma é %s\n", a+b);

  return 0;
}
```

Saída:

```
A soma é ?`??@?
```

```
Exemplo.c:7:22: warning: format ‘%d’ expects argument of type ‘int’, but argument 2 has type ‘char *’ [-Wformat=]
    7 |   printf("A soma é %d\n", a+b);
      |                     ~^    ~~~
      |                      |    |
      |                      int  char *
      |                     %s
A soma é -1192944216
```

É possível observar um comportamento anormal no exemplo 2, isso ocorre em razão da da forma que a linguagem trata o tipo “char” e o tipo “int”, isto é, ele permite somar inteiros ao “char” e deslocar o valor que ele representa na tabela ASCII, uma vez que os tipos “char” são inteiros que representam uma determinada posição na tabela ASCII, contudo, esse comportamento não aparece em diversas linguagens.

Complemento exemplo 2:

```c
int main(int argc, char const *argv[]) {

  int a = 500;
  char b[] = "550";
  printf("A soma é %d\n", a*b);

  return 0;
}
```

Saída:

```
Exemplo.c: In function ‘main’:
Exemplo.c:7:29: error: invalid operands to binary * (have ‘int’ and ‘char *’)
    7 |   printf("A soma é %d\n", a*b);
      |                            ^
      |                            |
      |                            char *
```

De modo a complementar o exemplo anterior, a operação de multiplicação “*”, não está disponível entre um inteiro e um char.

“Um literal é um valor, geralmente de um tipo primitivo, expressamente denotado em um programa”. Em outras palavras, o literal é o valor/sentido que é atribuído a uma variável no contexto da programação e não variável em si.

Exemplo 1

```c
int main(int argc, char const *argv[]) {

  int a = 500;

  return 0;
}
```

Exemplo 2

```c
int main(int argc, char const *argv[]) {

  char b[] = "550";

  return 0;
}
```

“Uma expressão é um literal, um construtor, uma constante, uma variável, uma invocação de uma função que retorne de valor, uma expressão condicional ou um operador com operandos que são, por sua vez, expressões.”

Exemplo 1:

```c
int main(int argc, char const *argv[]) {
  int a = 7;
  int b = 10;
  if (a == b) {
    printf("Verdadeiro\n");
  } else {
    printf("Falso\n");
  }
  return 0;
}
```

Saída:

```
Falso
```

Exemplo 2:

```c
int main(int argc, char const *argv[]) {
  int a = 7;
  int b = 10;
  if (a <= b) {
    printf("Verdadeiro\n");
  } else {
    printf("Falso\n");
  }
  return 0;
}
```

Saída:

```
Verdadeiro
```

Como foi possível visualizar, a analise logica das expressões booleanas acima, retorna um literal, “Verdadeiro” ou “Falso”, contudo, vale ressaltar que há diversas outras formas de expressões.