

Estruturas de Dados

Tipos de dados primitivos (1)

Os tipos de dados indicam as formas que podemos armazenar os dados em nossos programas. Ao definir o tipo de dado para uma variável estamos determinando uma série de coisas:

- a) A quantidade de espaço requerida para armazenamento.
- b) A forma como o conteúdo armazenado será interpretado.
- c) A faixa de valores que podem ser armazenados e manipulados.
- d) O conjunto de operações que podem ser realizadas com a variável.

Alguns tipos de dados são básicos, pré-definidos para o compilador da linguagem. Podemos utilizá-los em nossas declarações sem precisar fazer nenhuma definição prévia. Esses tipos básicos costumam ser chamados de **tipos de dados primitivos**, mas podem também ser chamados de *tipos embutidos* ou de *tipos nativos*. Com base neles e em alguns elementos adicionais podemos construir outros tipos de dados, os chamados **tipos de dados construídos**.

Tipos de dados numéricos primitivos inteiros em linguagem C (32 bits)

Tipo	Bytes	Mín	Máx	Obs
char	1	-128	127	Tipo inteiro de 1 byte com sinal
unsigned char	1	0	255	Tipo inteiro de 1 byte sem sinal
short int	2	-32.768	32.767	Tipo inteiro de 2 bytes com sinal
unsigned short int	2	0	65.535	Tipo inteiro de 2 bytes sem sinal
long int	4	-2.147.483.648	2.147.483.647	Tipo inteiro de 4 bytes com sinal
unsigned long int	4	0	4.294.967.295	Tipo inteiro de 4 bytes sem sinal
long long int	8	-9.223.372.036.854.775.808	9.223.372.036.854.775.807	Tipo inteiro de 8 bytes com sinal
unsigned long long int	8	0	18.446.744.073.709.551.615	Tipo inteiro de 8 bytes sem sinal

Na linguagem C existe também o pseudo-tipo `int`, que indica o tipo inteiro padrão da plataforma para a qual o compilador é projetado. Se a plataforma for de 16 bits (2 bytes, típica de microcomputadores antigos), uma variável declarada como `int` corresponde ao tipo primitivo `short int` da linguagem. Se a plataforma for de 32 bits (4 bytes, da maioria dos microcomputadores atuais) uma variável declarada como `int` corresponde ao tipo primitivo `long int`, conforme o compilador utilizado.

Tipos de dados numéricos primitivos inteiros em Pascal

Tipo	Bytes	Mín	Máx	Obs
ShortInt	1	-128	127	Tipo inteiro de 1 byte com sinal
Byte	1	0	255	Tipo inteiro de 1 byte sem sinal
Integer	2	-32.768	32.767	Tipo inteiro de 2 bytes com sinal
Word	2	0	65.535	Tipo inteiro de 2 bytes sem sinal
LongInt	4	-2.147.483.648	2.147.483.647	Tipo inteiro de 4 bytes sem sinal

Além dos tipos inteiros, existem os tipos reais, destinados ao armazenamento de valores cuja parte fracionária é importante.

Tipos de dados numéricos primitivos reais em linguagem C

Tipo	Bytes	Faixa	Obs
float	4	$\pm 10^{-38}$ a $\pm 10^{38}$	Tipo real de precisão simples
double	8	$\pm 10^{-308}$ a $\pm 10^{308}$	Tipo real de precisão dupla
long double	12	$\pm 10^{-4932}$ a $\pm 10^{4932}$	Não suportado em alguns compiladores

Estruturas de Dados
Tipos de dados primitivos (1)

Tipos de dados numéricos primitivos reais em Pascal

<i>Tipo</i>	<i>Bytes</i>	<i>Faixa</i>	<i>Obs</i>
Single	4	$\pm 10^{-45}$ a $\pm 10^{38}$	Precisão de 7 casas depois da vírgula
Real	6	$\pm 10^{-39}$ a $\pm 10^{38}$	Precisão de 11 casas depois da vírgula
Double	8	$\pm 10^{-324}$ a $\pm 10^{308}$	Precisão de 15 casas depois da vírgula
Extended	10	$\pm 10^{-4932}$ a $\pm 10^{4932}$	Precisão de 19 casas depois da vírgula
Comp	8	-10^{18} a 10^{18}	Precisão de 19 casas depois da vírgula

Exemplos de programas em C

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

```
int main(void)
```

```
{ unsigned char cNum;  
  int iNum;
```

```
    cNum = 0;
```

```
    iNum = 0;
```

```
    while (cNum <= 260)
```

```
    { printf("%d %d\n", iNum, cNum);
```

```
      iNum = iNum + 1;
```

```
      cNum = cNum + 1;
```

```
    }
```

```
    return 0;
```

```
}
```

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

```
int main (void)
```

```
{ float valor = 1000.0;
```

```
  int cont = 0;
```

```
    while (cont < 1000 * 10)
```

```
    { valor = valor - 0.1;
```

```
      cont = cont + 1;
```

```
    }
```

```
    printf("\nValor: %f %d",
```

```
          valor, cont);
```

```
    return 0;
```

```
}
```

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

```
int main(void)
```

```
{ float valor;
```

```
    valor = 5 / 2;
```

```
    printf("%f\n", valor);
```

```
    return 0;
```

```
}
```

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

```
int main(void)
```

```
{ char v1;
```

```
  short int v2;
```

```
  long int v3;
```

```
  int v4;
```

```
  float v5;
```

```
    printf("%d %d %d %d %d %d\n",
```

```
          sizeof(v1), sizeof(v2 ),
```

```
          sizeof(v3), sizeof(v4 ),
```

```
          sizeof(v5), sizeof(double)
```

```
    );
```

```
    return 0;
```

```
}
```