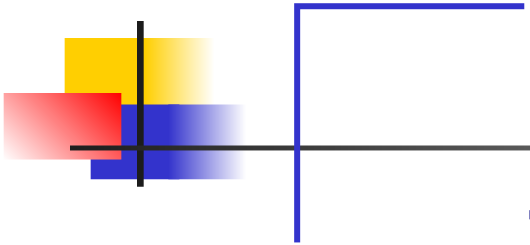
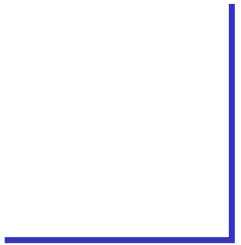


Tipos de Dados e Comandos de Entrada e Saída

Algoritmos e Programação A



Tipos de Dados

A linguagem C tem 5 tipos básicos de dados:

- `char` (caractere)
- `int` (inteiro)
- `float` (ponto flutuante)
- `double` (ponto flutuante de precisão dupla)
- `void` (sem valor)

Tipos de Dados

- Em C, outros tipos de dados são derivados doss 5 tipos básicos de dados.
  - Tipo booleano, por exemplo, não existe na linguagem C.

Tipos de Dados definidos pelo padrão ANSI

Tipo	Tamanho aprox. em bits	Faixa
char	8	-127 a 127
int	16	-32.767 a 32.767
unsigned int	16	0 a 65.535
short int	O mesmo que int	
long int	32	-2.147.483.647 a 2.147.483.647
unsigned long int	32	0 a 2.294.967.265
float	32	6 dígitos de precisão
double	64	10 dígitos de precisão
long double	80	10 dígitos de precisão

Tipos de Dados definidos pelo padrão ANSI

```c
#include <stdio.h>

int main(){

    printf("Um dado inteiro utiliza %d bytes\n", sizeof(int));

    return 0;
}

```

Variáveis

- **Variável**: posição nomeada de memória.
- Possui um nome para identificação e um tipo de dado.
- Declaração de variáveis:

```
tipo_de_dado  nome_var1, nome_var2, … nome_varN;
```

Constantes

- São dados que não variam durante a execução do programa.

- Por exemplo:

  Área e perímetro do círculo

  $A = \pi r^2$ $\quad\quad$ $P = 2\pi r$

- π é um dado constante, os demais dados são variáveis.

Operadores Aritméticos

- Atuam sobre variáveis, constantes e funções numéricas e produzem um resultado numérico.
- Operadores e suas prioridades:

Prioridade	Operador	Operação
1	-	Inversão de sinal
2	*	Multiplicação
2	/	Divisão
2	%	Resto de Divisão
3	+	Adição
3	-	Subtração

Operadores Aritméticos

- Os operadores com mesma prioridade são executados na ordem, da esquerda para a direita.
- O uso do parênteses define uma ordem de execução prioritária em relação à prioridade dos operadores da linguagem.
- Por exemplo:

x + 10 * Y - 2

(x + 10) * (Y - 2)

Comandos de Entrada e Saída

- Para possibilitar a interação do usuário com os programas, existem os comandos que fazem a entrada (leitura) e saída (escrita) de dados.
- O dispositivo padrão de entrada é o teclado.
- O dispositivo padrão de saída de dados é o monitor de vídeo.
- Necessário incluir a biblioteca stdio.h

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

Comando de Entrada

- Atribui um valor digitado a uma variável.

```
scanf(“     “,          );
```

Diretivas — Endereços das variáveis

```
int x;
scanf("%d", &x);
```

```
char y;
scanf("%c", &y);
```

```
float n;
scanf("%f", &n);
```

Diretivas

Formato	Tipo
%s	string
%c	char
%d	int
%f	float, double

Comando de Saída

- Mostra mensagem e/ou dados na tela do computador

```
printf(“                “,        );
```

Texto da mensagem com ou sem diretivas.

Variáveis (somente quando forem usadas diretivas, na parte inicial do comando printf).

```
int x = 5, y = 10;

printf("Exemplo de mensagem!!!\n");

printf("Exibixão de valores %d %d", x, y);
```

Comando de Saída

- Códigos podem ser usados no comando de saída (printf).
- Na tabela a seguir, temos alguns exemplos:

Código	Significado
\n	Nova linha
\"	Aspas duplas
\´	Aspas simples
\\	Barra invertida
\a	Beep
\t	Tabulação

- Outros formatos e códigos podem ser encontrados na bibliografia básica da disciplina.

Para resolver...

- O que será exibido na tela pelo programa ao lado?

```c
#include <stdio.h>

int main(){

    int a, b;
    float c, d;

    a = 3;
    b = a * 3;
    c = b / 2;
    d = a + c;
    d = c - (d/2) + c * d;
    c = a % 2;
    c = c + d;
    printf("c = %f\n", c);

    return 0;
}
```

Cast

- Um *cast* faz com que uma expressão assuma determinado tipo de dado.

```c
#include <stdio.h>

int main(){

      int i = 7;
      float a, b;
      a = i/2;
      b = (float) i/2;
      printf("a = %f, b= %f\n", a, b);

     return 0;
}
```

Outras possibilidades de C

- Atribuições múltiplas

```
a = b = i;
```

```
a = b = c = 10;
```

- Comandos aritméticos reduzidos

```
x += 20;
```
equivale à
```
x = x + 20;
```

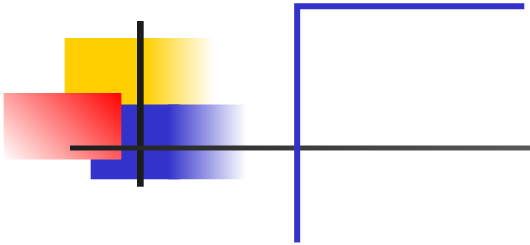
```
x -= 4;
```
equivale à
```
x = x - 4;
```

Operadores ++ e --

- Podem ser usados antes ou depois do operando.
  - Quando usado antes, a operação de incremento ou decremento é realizada antes da avaliação de toda a expressão.
  - Quando usado depois, a operação é realizada após a avaliação da expressão.

```
x = 10;
y = ++x;
printf("x = %d, y= %d\n", x, y);

x = 10;
y = x++;
printf("x = %d, y= %d\n", x, y);
```



Vamos praticar!

Finalizar a Lista 1

Iniciar a Lista 1B