# PCS2056 – Linguagens e Compiladores

**Assunto**: Tradução das expressões.

**Objetivo**: Com base na especificação do ambiente de execução, desenvolvido anteriormente, projetar o mecanismo de tradução para as expressões aritméticas e booleanas de uma linguagem imperativa.

**Palavras-chave**:
tabelas de símbolos e de atributos
expressões aritméticas
expressões booleanas

**Atividades de projeto**:

1) Escolher uma linguagem de saída para a qual as expressões deverão ser traduzidas. Escolher as convenções de representação das variáveis e constantes aritméticas e booleanas: formatos internos para números inteiros e reais, e a convenção para os valores true e false.

2) Projetar um esquema de tradução de expressões aritméticas simples, sem parênteses, envolvendo variáveis simples e constantes aritméticas e as cinco operações aritméticas binárias usuais: soma, subtração, multiplicação, divisão, potenciação.

3) Completar o projeto de expressões aritméticas ampliando as expressões da questão anterior para incorporarem operadores + e – unários, parênteses e chamadas de funções.

4) Projetar um esquema de tradução para a comparação entre expressões aritméticas usando operadores de comparação $>$, $<$, $=$, $\neq$, $\geq$, $\leq$.

5) Projetar a compilação de expressões booleanas simples, envolvendo variáveis e constantes booleanas, e os operadores booleanos usuais and, or, not.

6) Ampliar as expressões booleanas incorporando as comparações entre expressões aritméticas, chamadas de funções booleanas e parênteses.

7) Completar as expressões booleanas permitindo a comparação entre expressões booleanas através dos operadores de comparação $=$ e $\neq$.

# PCS2056 – Linguagens e Compiladores

**Assunto**: Tradução dos comandos

**Objetivo**: Completar a especificação do código gerado pelo compilador e das rotinas do ambiente de execução da linguagem de alto nível.

**Palavras-chave**:
estruturas de controle de fluxo: desvios, if-then, if-then-else, decisões múltiplas, while, do-until comandos imperativos: chamadas de procedimentos, atribuição de valores, entrada e saída

**Atividades de Projeto**:

1) Definir detalhadamente o mapeamento de cada um dos comandos da linguagem de alto nível que você está implementando para a forma de código-objeto de nível mais baixo, a ser produzido cada vez que tais comandos forem encontrados no programa-fonte pelo compilador. Isso deve completar a especificação do código-objeto que deve ser gerado pelo compilador que está sendo construído.
2) Definir, para cada um dos comandos da linguagem de alto nível a ser compilada, as rotinas do ambiente de execução que forem necessárias para o correto funcionamento do código gerado em tempo de execução, completando dessa forma a especificação do conjunto das rotinas que compõem o ambiente de execução da linguagem implementada.
3) Verificar as especificações acima desenvolvidas em conjunto as especificações elaboradas nas aulas anteriores, testando sua consistência e compatibilidade mútua. Isso é importante para que nas próximas atividades as rotinas semânticas possam ser construídas e integradas com o reconhecedor sintático para formar o compilador desejado.

**Bibliografia complementar:**
Para servirem como referência para o projeto e implementação de linguagens de programação, os livros abaixo, todos relativamente recentes, enriquecem e diversificam a bibliografia clássica apresentada anteriormente e fornecem um vasto material complementar para o estudo desse assunto.

**sobre linguagens de programação:**
H. Ledgard e M. Marcotty – The Programming Language Landscape – SRA, 1986
R. Sebesta – Concepts of Programming Languages – Addison Wesley, 2002 (5th edition)
T. W. Pratt e M. V. Zelkowitz – Programming Languages - Design and Implementation – Prentice Hall, 1999 (3rd edition)
D. Appleby e J. J. VandeKopple – Programming Languages Paradigm and Practice – McGraw-Hill, 1997 (2nd edition)

**sobre compiladores:**
J. Holmes – Object-oriented compiler construction – Prentice Hall, 1995 (conceitos e teoria)
J. Holmes – Building your own compiler with C++ – Prentice Hall, 1995 (implementação)
A. W. Appel – Modern compiler implementation in Java - basic techniques – Cambridge, 1997
A. W. Appel – Modern compiler implementation in ML - basic techniques – Cambridge, 1997
A. W. Appel – Modern compiler implementation in C - basic techniques – Cambridge, 1997
J.P. Tremblay e P.G. Sorenson – The theory and practice of compiler writing – McGraw-Hill, 1985
R. Wilhelm e D. Maurer – Compiler Design – Addison Wesley, 1995