Introduction to Programming

CSCE 110

Drawn from James Tam’s material
Computer Programs

Binary is the language of the computer

1) A programmer writes a computer program

2) A __________ converts the program into a form that the computer can understand (e.g., gpc)

3) An __________ program is created

4) Anybody who has this __________ installed on their computer can run (use) it.
Translators

- Convert computer programs to machine language
- Types
  - ________________

- Each time that the program is run the ________________ translates the program (translating a part at a time).
- If there are any errors during the process of ________________ the program, the program will stop running right when the error is encountered.
Translators

• Convert computer programs to machine language

• Types
  – ___________________
  – ___________________

• Before the program is run the _________ translates the program (compiling it all at once).

• If there are any errors during the ____________ process, no machine language executable will be produced.

• If there are no errors during _______________ then the translated machine language program can be run.
Compiling Programs: Basic View

Pascal program

_filename.p_

input

Pascal compiler

output

Machine language program

a.out
Basic Structure Of Pascal Programs

Program name.p (Pascal source code)

Part I: Header
Program documentation
program name (input, output);

Part II: Declarations
const
: :

Part III: Statements
begin
: :
end.
Header

• Program documentation
  – Comments for the reader of the program (and not the computer)
    • (* Marks the __________ of the documentation
    • *) Marks the __________ of the documentation

• Program heading
  – Keyword: program, Name of program, if input and/or output operations performed by the program.
Tax-It v1.0: This program will electronically calculate your tax return.

This program will only allow you to complete a Canadian tax return.

program taxIt (input, output);
Declarations

• List of constants
• More to come later during this term regarding this section
Statements

• The instructions in the program that actually gets things done
• They tell the computer what to do as the program is running
• Statements are ____________________________
• Example statements: display a message onscreen, prompt the user for input, open a file and write information to that file etc.
• Much more to come later throughout the rest of the term regarding this section
The Smallest Pascal Program

program smallest;
begin
end.

Note: The name in the header "smallest" should match the filename "smallest.pas". You can find an copy of this program [here](#) and the compiled version is [here].
Creating and Compiling Pascal in the CS Department

Use a text editor to create file “filename.pas”

To compile the program in Unix type “gpc filename.pas”

To run the program in Unix type “a.out”
Source Code Vs. Executable Files

- **Source Code**
  - A file that contains the Pascal program code.
  - It must end with a ‘dot-p’ or a ‘dot-pas’ suffix (program name.p).
  - Can be viewed and edited.
  - Cannot be executed.

```pascal
program smallest;
begin
    // Code
end.
```
Source Code Vs. Executable Files

- Executable Code
  - A file that contains machine language (binary) code.
  - By default this file will be called “a.out”.
  - It cannot be directly viewed or edited (meaningless).
  - It can be executed.

- A file that contains machine language (binary) code.

- By default this file will be called “a.out”.
- It cannot be directly viewed or edited (meaningless).
- It can be executed.
1. Syntax/Compile Errors

They occur as the program is being compiled.
2. Runtime Errors

They occur as the program is running and ____________________________.
3. Logic Errors

They occur as the program is running.
Reserved Words

- Have a predefined meaning in Pascal that cannot be changed

- and
- array
- begin
- case
- const
- div
- do
- downto
- else
- end
- file
- for
- foward
- function
- goto
- if
- in
- label

For more information on reserved words go to the url: http://www.gnu-pascal.de/gpc/index.html
Reserved Words

• Have a predefined meaning in Pascal that cannot be changed

• and
• array
• begin
• case
• const
• div
• do
• downto
• else
• end
• file
• for
• foward
• function
• goto
• if
• in
• label
• mod
• nil
• not
• of
• or
• packed
• procedure
• program
• record
• repeat
• set
• then
• to
• type
• until
• var
• while

For more information on reserved words go to the url: http://www.gnu-pascal.de/gpc/index.html
Standard Identifiers

- Have a predefined meaning in Pascal that **SHOULD NOT** be changed

- Predefined constants
  - false
  - true
  - maxint

- Predefined types
  - boolean
  - char
  - integer
  - real
  - text

- Predefined files
  - input
  - output

For more information on standard identifiers go to the url: http://www.gnu-pascal.de/gpc/index.html
Predefined Functions

- abs
- arctan
- chr
- cos
- eof
- eoln
- exp
- ln
- odd
- ord
- pred
- round
- sin
- sqr
- sqrt
- succ
- trunc

Know the ones in Table 3.1 of your book.
Predefined Procedures

• dispose
• get
• new
• pack
• page
• put
• read

• readln
• reset
• rewrite
• unpack
• write
• writeln
Variables

• Set aside a location in memory
  – This location can store __________ ‘piece’ of information

• Used to store information (temporary)
  – *At most* the information will be accessible

Picture from Computers in your future by Pfaffengerber B
Variable Types

- **integer** – whole numbers
- **real** – whole numbers and fractions
- **char** – alphabetic, numeric and miscellaneous symbols (in UNIX type “man ascii”)
- **boolean** – a true or false value
Using Variables

• Usage (______________________________!)
  – Declaration
  – Accessing or assigning values to the variables
Declaring Variables

- Sets aside memory
- Memory locations are addressed through the name of the variable
Declaring Variables

- Sets aside memory
- Memory locations are addressed through _________

Name of variable

RAM

RESERVED
Declaring Variables

Declare variables between the ‘begin’ and ‘end.’

Part I: Header
Program documentation
program name (input, output);

Part II: Declarations
const
:

Part III: Statements
begin
:
end.

Declare variables just after the ‘begin’
Declaring Variables

Format:

var name of first variable : type of first variable;
var name of second variable : type of second variable;
Example Declaring Variables

```pascal
program variableExample1;
begin
  var height : real;
  var weight : real;
  var age : integer;
end.
```

• You can find an copy of this program [here](#) and the compiled version is [here](#).
Global Variables

• Variables declared outside of the begin-end pair.

```plaintext
program anExample;
var num1 : integer;
begin
  var num2 : integer;
end.
```

For now avoid doing this (additional details will be provided later in the course): generally this is regarded as ______ programming style.
Variable Naming Rules

• Should be ________________

• Any combination of letters, numbers or underscore (_________ begin with a number and ___________begin with an underscore)

• Can't be a reserved word (see the “Reserved Words” slide)
Variable Naming Conventions (2)

• Avoid using ________ identifiers (see the “Standard Identifiers” slides)
• Avoid distinguishing variable names only by ________
• For variable names composed of multiple words separate each word by __________
  ___________________________ (except for __________) or by using ____________.
Variable Naming Conventions

• Okay:
  – tax_rate
  – firstName

• Not Okay
  (_________________)
  - 1abc
  – test.msg
  – good-day
  – program

• Not okay
  (_________________)
  – x
  – writeln
Accessing Variables

• Can be done by referring to the name of the variable

• Format:
  name of variable

• Example:
  num
Assigning Values To Variables

• Format:

    Destination := Source;

    The source can be any ________________
    (_________________,
     ___________________ or
     ________________________)


Example of Assigning Values

program variableExample2;
begin
  var height : real;
  var weight : real;
  var age : integer;
  weight := height * 2.2;
end.

• You can find an copy of this program [here](#) and the compiled version is [here](#).
Assigning Values To Variables

program variableExample2;
beg
\begin{verbatim}
  var height : real;
  var weight : real;
  var age : integer;
  height := 69;
  weight := height * 2.2;
end.
\end{verbatim}

Important lesson: **ALWAYS** _______________ your variables to some default starting value before using them.
Assigning Values To Variables

```pascal
program variableExample;
begin
    var num1 : integer;
    var num2 : real;

    num1 := 12;
    num2 := 12.5;
    num2 := num1;
    num1 := num2;
end.
```

Avoid assigning different types
Reminder

Variables Must First Be Declared Before They Can Be Used!

The ________ Way

```pascal
program anExample;
begin
  var num : integer;
  num := 888;
end.
```
Reminder

Variables Must First Be Declared Before They Can Be Used!

The __________ Way

program anExample;
begin
  num := 888;
  var num : integer;
end.

Compile Error:
Where is num???
Named Constants

• A memory location that is assigned a value that
  ________________________________
• Declared in the constant declaration ("const") section
• The naming conventions for choosing variable names generally apply to constants but the name of constants should be all _____________. (You can separate multiple words with ________________).
Named Constant Format

const
    NAME_OF_FIRST_CONSTANT = value of 1st const;
    NAME_OF_SECOND_CONSTANT = value of 2nd const;
    etc.
Named Constant Examples

const
    TAX_RATE = 0.25;
    SAMPLE_SIZE = 1000;
    YES = True;
    NO = False;
Declaring Named Constants

Named constants are declared in the declarations section

Part I: Header
Program documentation
program name (input, output); 

Part II: Declarations
const
:

Part III: Statements
begin
:
end.
program anExample;
const
    TAX_RATE = 0.25;
    SAMPLE_SIZE = 1000;
    YES = True;
    NO = False;
    MY_FIRST_INITIAL = 'J';
begin
    var grossIncome : real;
    var afterTaxes : real;
    grossIncome := 100000;
    afterTaxes :=
        grossIncome - (grossIncome * TAX_RATE);
end.
Purpose Of Named Constants

- Makes the program easier to understand

\[
populationChange := (0.1758 - 0.1257) \times \text{currentPopulation};
\]

Better

```plaintext
const
    BIRTH_RATE = 0.1758;
    DEATH_RATE = 0.1257;
begin
    populationChange := (BIRTH_RATE - DEATH_RATE) \times \text{currentPopulation};
```
Purpose Of Named Constants

• Makes the program __________________
  • If the constant is referred to _______________ throughout the program, changing the value of the constant ________ will change it throughout the program.
Purpose Of Named Constants

```pascal
program population (output); 
const
  BIRTH_RATE = 0.1758;
  DEATH_RATE = 0.1257;
begin
  var populationChange : real;
  var currentPopulation : real;
  populationChange := (BIRTH_RATE - DEATH_RATE) * currentPopulation;
  if (populationChange > 0) then
    writeln('Births: ', BIRTH_RATE, ' Deaths:', DEATH_RATE, ' Change:',
            populationChange)
  else if (populationChange < 0) then
    writeln('Births: ', BIRTH_RATE, ' Deaths:', DEATH_RATE, ' Change:',
            populationChange)
end.
```
program population (output);
const
    BIRTH_RATE = 0.5;
    DEATH_RATE = 0.1257;
begin
    var populationChange : real;
    var currentPopulation : real;
    populationChange := (BIRTH_RATE - DEATH_RATE) * 
currentPopulation;
    if (populationChange > 0) then
        writeln('Births: ', BIRTH_RATE, ' Deaths:', DEATH_RATE, ' Change:', populationChange)
    else if (populationChange < 0) then
        writeln('Births: ', BIRTH_RATE, ' Deaths:', DEATH_RATE, ' Change:', populationChange)
end.
Purpose Of Named Constants

program population (output);
const
  BIRTH_RATE = 0.1758;
  DEATH_RATE = 0.01;
begin
  var populationChange : real;
  var currentPopulation : real;
  populationChange := (BIRTH_RATE - DEATH_RATE) * currentPopulation;
  if (populationChange > 0) then
    writeln('Births: ', BIRTH_RATE, ' Deaths:', DEATH_RATE, ' Change:', populationChange)
  else if (populationChange < 0) then
    writeln('Births: ', BIRTH_RATE, ' Deaths:', DEATH_RATE, ' Change:', populationChange)
end.
Performing Calculations

<table>
<thead>
<tr>
<th>Operation</th>
<th>Symbol (Operator)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Addition</td>
<td>+</td>
</tr>
<tr>
<td>Subtraction</td>
<td>-</td>
</tr>
<tr>
<td>Multiplication</td>
<td>*</td>
</tr>
<tr>
<td>Real number division</td>
<td>/</td>
</tr>
<tr>
<td>Integer division</td>
<td>DIV</td>
</tr>
<tr>
<td>Remainder (modulo)</td>
<td>MOD</td>
</tr>
</tbody>
</table>
Priority of Operators

• High:
  – *
  – /
  – DIV
  – MOD

• Low:
  – +
  – –
  – unary +
  – unary -
Program Documentation

• It doesn’t __________________________
• It doesn’t __________________________
• It is for ____________________________
Program Documentation

• What does the program do e.g., tax program.
• What are it’s capabilities e.g., it calculates personal or small business tax.
• What are it’s limitations e.g., it only follows Canadian tax laws and cannot be used in the US.
Program Documentation

• What is the version of the program
  – If you don’t use numbers for the different versions of your program then consider using dates.

• How does the program work.
  – This is often a description in English (or another high-level) language that describes the way in which the program fulfills its functions.
  – The purpose of this description is to help the reader more quickly understand how the program works.