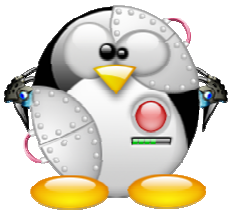
Algorithms

- **Problem**: Write pseudocode for a program that keeps asking the user to input integers until the user enters zero, and then determines and outputs the smallest integer. (Hint: Think about keeping a variable that stores the minimum value. Then you can compare the minimum to each value read in. If the number you read in is smaller than the minimum, it should become the new minimum.)

- Sample input/output for program is listed below. Your program's output is in bold.

  Please input an integer, 0 to end: -2
  Please input an integer, 0 to end: 10
  Please input an integer, 0 to end: -8
  Please input an integer, 0 to end: 17
  Please input an integer, 0 to end: 0

  The smallest integer entered was -8.



Introduction to C

Topics

- Compilation
- Using the gcc Compiler
- The Anatomy of a C Program
- 104 C Programming Standards and Indentation Styles

Reading

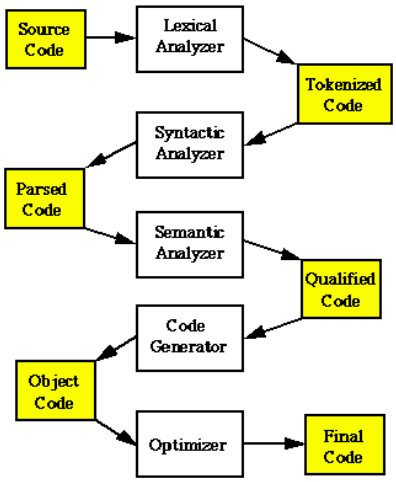
- **Sections Chapter 1, 2.1**

Writing C Programs

- A programmer uses a **text editor** to create or modify files containing C code.
- Code is also known as **source code**.
- A file containing source code is called a **source file**.
- After a C source file has been created, the programmer must **invoke the C compiler** before the program can be **executed** (**run**).

Compiler

- The five stages of a compiler combine to translate a high level language to a low level language, generally closer to that of the target computer.
- Each stage, or sub-process, fulfills a single task and has one or more classic techniques for implementation.



Component	Purpose
Lexical Analyzer	Analyzes the Source Code Removes "white space" and comments Formats it for easy access (creates tokens) Tags language elements with type information Begins to fill in information in the SYMBOL TABLE **
Syntactic Analyzer	Analyzes the Tokenized Code for structure Amalgamates symbols into syntactic groups Tags groups with type information
Semantic Analyzer	Analyzes the Parsed Code for meaning Fills in assumed or missing information Tags groups with meaning information
Code Generator	Linearizes the Qualified Code and produces the equivalent Object Code
Optimizer	Examines the Object Code to determine whether there are more efficient means of execution

The Symbol Table is the data structure that all elements of the compiler use to collect and share information about symbols and groups of symbols in the program being translated

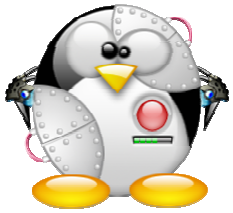
Using the C Compiler at UMBC

- Invoking the compiler is system dependent.
  - At UMBC, we have two C compilers available, **cc** and **gcc**.
  - For this class, we will use the **gcc** compiler as it is the compiler available on the Linux system.

Invoking the gcc Compiler

At the prompt, type

gcc -ansi -Wall *pgm.c*



where ***pgm.c*** is the C program source file.

- **-ansi** is a **compiler option** that tells the compiler to adhere to the **ANSI C standard**.
- **-Wall** is an option to turn on all compiler **warnings** (best for new programmers).

The Result : a.out

- If there are no errors in pgm.c, this command produces an **executable file**, which is one that can be executed (run).
- The **gcc** compiler names the executable file **a.out**
- To execute the program, at the prompt, type

a.out

- Although we call this process “compiling a program,” what actually happens is more complicated.

3 Stages of Compilation

Stage 1: **Preprocessing**

- Performed by a program called the **preprocessor**
- Modifies the source code (in RAM) according to **preprocessor directives (preprocessor commands**) embedded in the source code
- Strips comments and whitespace from the code
- The source code as stored on disk is not modified.

3 Stages of Compilation (con't)

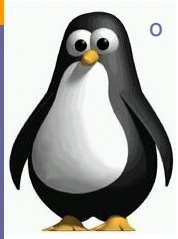
Stage 2: **Compilation**

- Performed by a program called the **compiler**
- Translates the preprocessor-modified source code into **object code (machine code)**
- Checks for **syntax errors** and **warnings**
- Saves the object code to a disk file, if instructed to do so (we will not do this).
  - If any compiler errors are received, no object code file will be generated.
  - An object code file will be generated if only warnings, not errors, are received.

3 Stages of Compilation (con't)

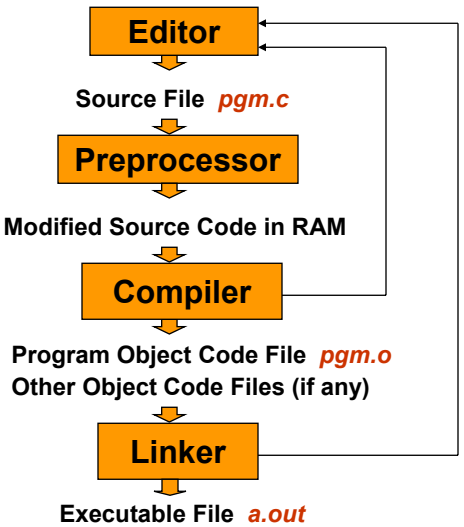
Stage 3: **Linking**

- Combines the program object code with other object code to produce the executable file.
- The other object code can come from the **Run-Time Library**, other libraries, or object files that you have created.



- Saves the executable code to a disk file. On the Linux system, that file is called **a.out**.
  - If any linker errors are received, no executable file will be generated.

Program Development Using gcc



A Simple C Program

```
/* Filename:      hello.c
   Author:          Brian Kernighan & Dennis Ritchie
   Date written:  ?/?/1978
   Description:   This program prints the greeting
                  "Hello, World!"
*/
#include  <stdio.h>
int main ( )
{
    printf ("Hello, World!\n") ;
    return 0 ;
}
```

Anatomy of a C Program

program header comment

preprocessor directives (if any)

```
int main ( )
{
    statement(s)
    return 0 ;
}
```

Program Header Comment

- A **comment** is descriptive text used to help a reader of the program understand its content.
- All comments must begin with the characters /* and end with the characters */
- These are called **comment delimiters**
- The program header comment always comes first.
- Look at the class web page for the required contents of our header comment.

Preprocessor Directives

- Lines that begin with a # in column 1 are called **preprocessor directives** (**commands**).
- Example: the **#include <stdio.h>** directive causes the preprocessor to include a copy of the standard input/output header file **stdio.h** at this point in the code.
- This header file was included because it contains information about the printf ( ) function that is used in this program.

int main ( )

- Every program must have a **function** called **main**. This is where program execution begins.
- main() is placed in the source code file as the first function for readability.
- The **reserved word** "int" indicates that main() **returns** an integer value.
- The parentheses following the reserved word "main" indicate that it is a function.

The Function Body

- A left brace (curly bracket) -- **{** -- begins the **body** of every function. A corresponding right brace -- **}** -- ends the function body.
- The style is to place these braces on separate lines in column 1 and to indent the entire function body 3 to 5 spaces.

printf ("Hello, World!\n") ;

- This line is a C **statement**.
- It is a **call** to the function **printf ( )** with a single **argument (parameter)**, namely the **string** "Hello, World!\n".
- Even though a string may contain many characters, the string itself should be thought of as a single quantity.
- Notice that this line ends with a semicolon. All statements in C end with a semicolon.

return 0 ;

- Because function main() returns an **integer value**, there must be a statement that indicates what this value is.
- The statement

  return 0 ;

  indicates that main() returns a value of zero to the operating system.
- **A value of 0 indicates that the program successfully terminated execution.**
- Do not worry about this concept now. Just remember to use the statement.

Another C Program

```
/*****************************************
** File: proj1.c
** Author: Joe Student
** Date: 9/15/01
** SSN: 123-45-6789
** Section: 0304
** E-mail: jstudent22@umbc.edu
**
** This program prompts the user for two integer values then displays
** their product.
**
*********************************************/
```

Another C Program (con't)

```
#include <stdio.h>
int main()
{
   int value1, value2, product ;
   printf("Enter two integer values: ") ;
   scanf("%d%d", &value1, &value2) ;
   product = value1 * value2 ;
   printf("Product = %d\n", product) ;
   return 0 ;
}
```

Good Programming Practices

- C programming standards and indentation styles are available on the 104 course homepage.
- You are expected to conform to these standards for all programming projects in this class and in CMSC 201. (This will be part of your grade for each project!)
- The program just shown conforms to these standards, but is uncommented (later).
- Subsequent lectures will include more “Good Programming Practices” slides.