Structured Programming

Mr. Dave Clausen

La Cañada High School

http://www.lcusd.net/lchs/dclausen/

I. What is programming?

- A computer can only carry out a small number of instructions or simple calculations. A computer can only solve the problem if it is broken down into smaller steps.

II. What is Structured programming?

- An attempt to formalize the logic and structure of programs. (i.e.)

```
Procedure Bubble_Sort (Var Original, Duplicate,
Sorted : ListType);
{Pre: The array is filled with ramdom numbers.
Post: The numbers will be sorted.}
Var
  Element,
  Index : Integer;
Begin
  WriteLn ('Sorting...');
  For Element := 1 to MaxEntries do
    For Index := MaxEntries downto (Element+1) do
      If Original[Index] < Original[Index-1]
       Then Swap (Original[Index], Original[Index-1]);
  Create_Sorted(Original, Sorted);
  Recreate_Original(Original, Duplicate);
  Pause
End;          {Bubble}
```

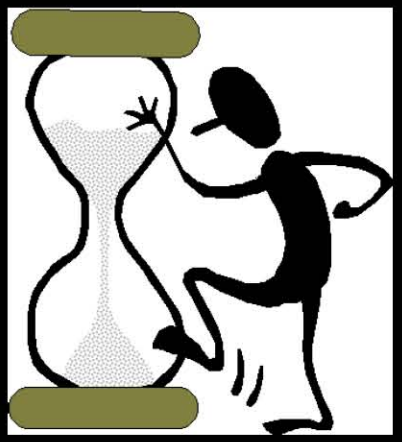
III. What is the Purpose of Structured Programming?

- To make computer programs
  - Easier to read
  - Easier to debug
  - Easier to understand
  - Easier to maintain

- To allow programmers to work as a team

Purpose 2

- To reduce testing time



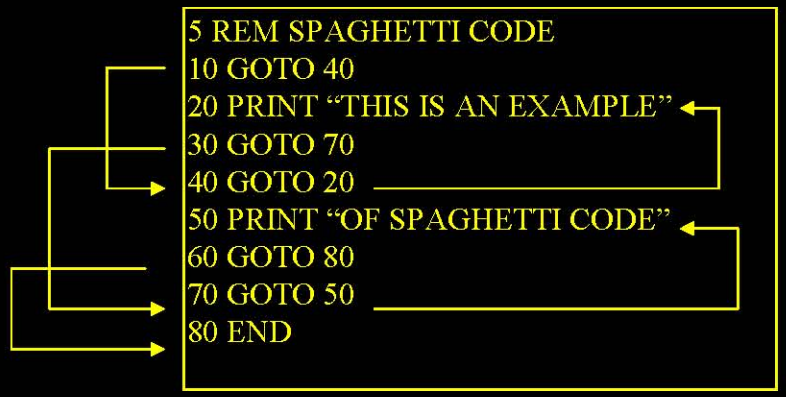
- To increase programming productivity
- To increase clarity by reducing the programs' complexity
- To decrease maintenance and effort

IV. Why is this special programming necessary?

- Programming in the 60's: "fiddling" with the program until it worked.
- "Spaghetti Code"

Spaghetti Code

Spaghetti Code.txt



Avoid using GOTO statements

- Edger W. Dijkstra 1968

  "The GOTO statement should be abolished from all higher level programming languages…"

  "…The GOTO statement is just too primitive; it is too much of an invitation to make a mess of one's program."

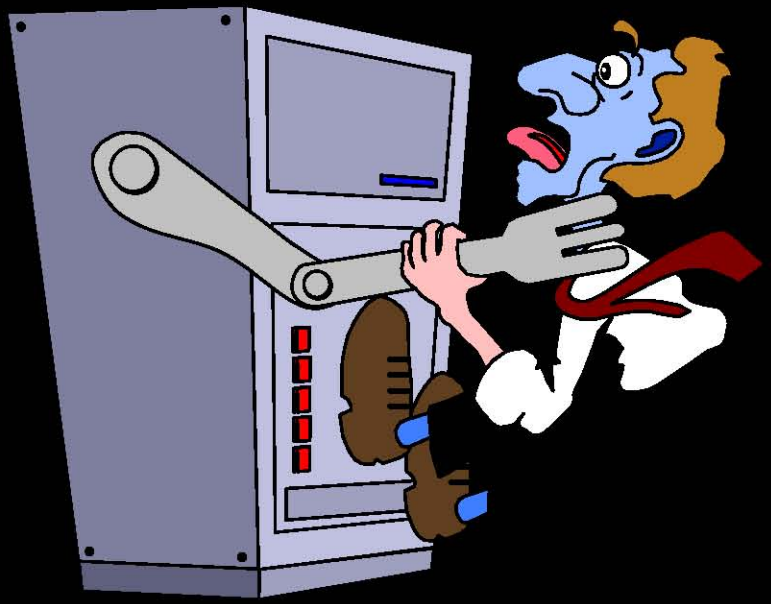
- Mr. Clausen "If you use a GOTO statement in your program, you will get a "0" zero."

Approaches to Programming

- Top Down Design
  - like the main points of an outline
  - divide the program into modules
- Bottom Up Approach
  - experimenting with the program
  - "let's try this and see if it works (test programs)

V. Define The Process of Structured Programming

- A. Analysis
  - Determine if the computer is the proper tool for the problem

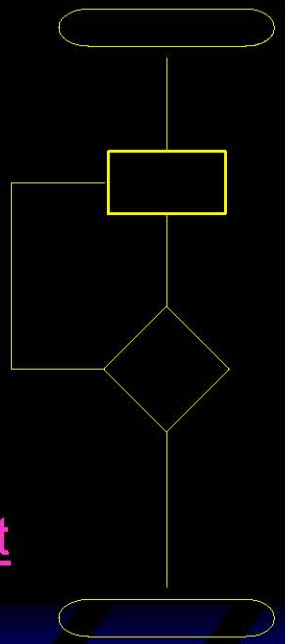


Process 2

- B. Specification
  - Define the problem in a clear and unambiguous manner considering:
    - 1. Input
    - 2. Output
    - 3. Processing
      - a) Sort
      - b) Calculate
      - c) Search
      - d) Store

Process 3

- C. Algorithm Design
  - Plan the solution as a series of separate steps using:
    - 1. Pseudo-code (English)
    - 2. Flowcharts
    - 3. Stepwise Refinement:
      - Level 0, Level 1, etc.
    - Use function or procedure Stubs
      - BASIC BASIC.txt Pascal
      - C+ + C++.txt Pascal.txt



Process 4

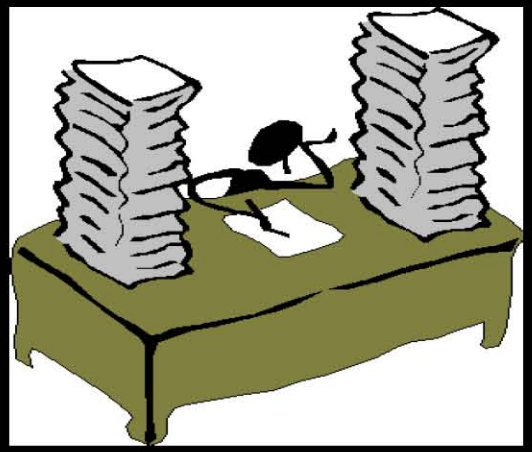
- 4. Top Down Design (modules/procedures/functions)
  - each module performs 1 task
  - each module has 1 entry & 1 exit
  - Conditional statements should not exit the module
  - You could have a module "call" another module
  - Variables are local to each module
  - Declare & Initialize all variables
  - Variable names should be descriptive
  - Use Value and/or Variable Parameters

Process 5

- D. Code the program
  - Convert your Pseudo Code and/or Algorithms into Level 0 Commands
  - Use Stepwise Refinement to add detail to Level 1 Commands
  - Keep Refining until all Procedures and Functions are complete
  - Use proper syntax

Process 6

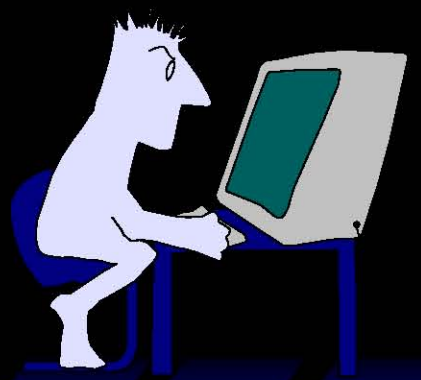
- E. Test the program
  - 1. Desk check



  - 2. Run the program, using sample data

i.e. Celsius To Fahrenheit

Celsius To Fahrenheit.txt



Process 7
(Testing the Program)

- 3. Test Border, Edge or "Extreme" cases
  Test Border, Edge or “Extreme” cases.txt
- 4. Debug the program (all paths)

- Syntax errors
- Logical errors
- Run Time errors
- Compiler errors

Debugging Research

"If the source of the problem is not immediately obvious, leave the computer and go somewhere where you can quietly look over a printed copy of the program. Studies show that people who do all of their debugging away from the computer actually get their programs to work in less time and in the end produce better programs than those who continue to work on the machine-more proof that there is still no mechanical substitute for human thought."*

Dale, Weams, Headington "Programming and Problem Solving with C++", Jones and Bartlett Publishers, 1997, pp80-81

Basili, V.R., Selby, R.W., "Comparing the Effectiveness of Software Testing Strategies",IEEE Trans. On Software Engineering, Vol. SE-13, No.12, pp 1278-1296, Dec. 1987

Process 8

- F. Interpretation
  - Does it produce the results that "solves" the problem?
- G. Documentation
  - Comments within the program

IV. Control Structures

- Corrado Bohm & Guiseppe Jacopini
  - 1964 Structure Theorem

    proved that any program logic, regardless of the complexity, can be expressed using the control structures of sequencing, selection, and iteration.

Control Structures 2

- A. Sequence
  - Instructions executed in order 1st, 2nd, 3rd, etc.
- B. Selection
  - (Branching, Conditionals)
  - If, If else, If then, If then else
  - Switch or Case statements

Control Structures 3

- C. Iteration (Repetition)
  - Indefinite Loops
    - while, while do
      - (condition checked at beginning)
    - do...while, Repeat Until
      - (condition checked at the end of the loop)
  - Definite Loops
    - for, for do (FOR NEXT LOOP)
  - (Recursion)

BASIC Line Numbers

- An outline of line numbers can help add structure to our program. The following list is a suggested format for a Structured Program using Top Down Design.

Structured Program Line Numbers

Program Design (APCS College Board)

- Program Design
  - The goal in designing a program is to solve the problem correctly, but also to design a program that is understandable, can adapt to changing circumstances, and has the potential to be reused in whole or in part. The design process needs to be based on a thorough understanding of the problem to be solved.

Program Design Process

- Problem Definition
  - Specification of the purpose and goals
  - Identification of subtasks to be performed
  - Identification of the ADT's (Abstract Data Types) and operations needed to solve the problem

Program Design Process 2

- Program Design
  - Identification of reusable components from existing code.
  - Subprogram decomposition
  - Choice of data structures and algorithms
  - Design of the user interface

Program Implementation

- The goals of program implementation parallel those of program design. Modules of the program that fill common needs should be built so that they can be reused easily in other programs. Control and data abstraction are important parts of program implementation.

Program Implementation 2

- Implementation Techniques
  - Methodology
    - Top Down Design (using stub procedures)
    - Bottom Up Development
  - Use of Abstraction
    - Control Abstraction
    - Data Abstraction
      - abstract data types
      - encapsulation and information hiding

Program Implementation 3

- Programming Constructs
  - Input and output
    - Interactive
    - Files
  - Control
    - Sequential
    - Conditional
      - Iteration
      - Recursion

Program Analysis

- Program Analysis
  - Analyze and test programs to determine whether they correctly meet their specifications.
  - Analyze programs to understand their time and space requirements when applied to different data sets.

Program Analysis 2

- Testing
  - Testing modules in isolation
  - Identify boundary cases and generate appropriate test data
  - Integration testing

Program Analysis 3

- Debugging
  - Categorizing errors
    - syntax
    - run-time
    - logic
  - Identifying and Correcting Errors
  - Techniques using a debugger, adding extra output statements, desk checking

Program Analysis 4

- Understanding and Modifying Existing Code
- Handling Errors
  - Robust Behavior
- Reasoning About Programs
  - Pre & Post Conditions
  - Assertions

Program Analysis 5

- Analysis of Algorithms
  - Informal Comparisons of running times
  - Exact Calculation of statement execution counts (Big - O notation)
- Numerical Limits
  - Limitations of finite representations (ie integer bounds, imprecision of floating point representations, & round off error)

Standard Data Structures

- Standard Data Structures
  - Data structures are the means by which the information used by a program is represented within the program. An important theme of the development and application of data structures is abstraction.
  - Simple Data Types (int , double, char, bool)
  - Records (Structs)
  - Arrays (Vectors or Matrices)

Standard Algorithms

- Standard algorithms can serve as examples of good solutions to standard problems. Programs implementing them can serve as models of good program design. They provide examples for analysis of program efficiency. Many are intertwined with standard data structures.

Standard Algorithms 2

- Searching
  - Sequential (Linear)
  - Binary
- Sorting
  - Selection
  - Bubble
  - Insertion
  - Merge sort
  - Quick sort

Standard Algorithms 3

- Operations
  - Insertion
  - Deletion
  - Traversals