University of Washington Computer Programming I

Structuring Program Files

Structuring Programs

The function is the basic unit of a C program

Programs often use many functions

Some are defined within the program

Some are in libraries

Organizing and ordering the functions and other parts within the .c file is important

Order in the Program

General principle: identifiers (names of things) must be declared before they are used.

Variables:

place them first within each function

#define constants:

placed at the top of the .c file

What about functions?

Order for Functions in the .c File

Function names are identifiers, so… they too must be declared before they are used:

```
#include <stdio.h>

void fun2  (void) { ... }
void fun1  (void) { ...; fun2(); ... }
int    main (void) { ...; fun1(); ... return 0; }
```

fun1 calls *fun2*, so *fun2* is defined before *fun1*, etc.

Function Prototypes

Insisting that all the code of each function precede all calls to that function is sometimes:

Impossible: function A calls B, and B calls A

Inconvenient: printf() is a function, but we don't want its code in our program

But the ordering rule requires that the function names be declared before they can be used (in a call).

Is there any solution?

Solution: Function Prototypes

Function prototypes allow us to define the name, so that it can be used, without giving the code for the function.

The prototype gives the function name, return type, and the types of all the parameters but no code.

In place of the { } code block, there is a semicolon.

Example Function Prototypes

```
void Useless(void);

void PrintInteger(int value);

double CalculateTax (double amount,
                     double rate);
```

G3-7

Using Prototypes

Write prototypes for your functions near the top of the program

Can use the function anywhere thereafter

Fully define the function later, wherever convenient

Highly recommended to aid program organization

G3-8

Library Functions

What about library functions, like printf?

You must also tell the compiler that you are going to use the library which contains printf

This is the purpose of the #include directive

The linker knows where the libraries are

G3-9

#include <stdio.h>

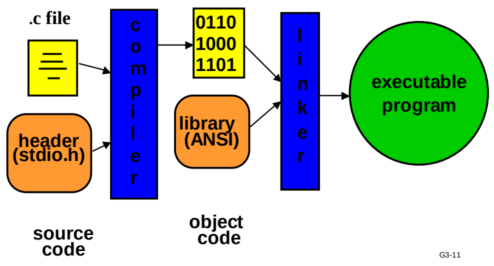
The "#include <...>" means "go get the file ... and insert what's in it right here (as if it had been typed here)"

stdio.h contains function prototypes for scanf and printf and the other functions in the standard I/O library

The actual code for them is NOT there, just prototypes. The (result of compiling) the code is in a library that is combined with your code by the linker

G3-10

Compilers, Linkers, etc.



G3-11

Putting it All Together

#include directives

…

#define constants

…

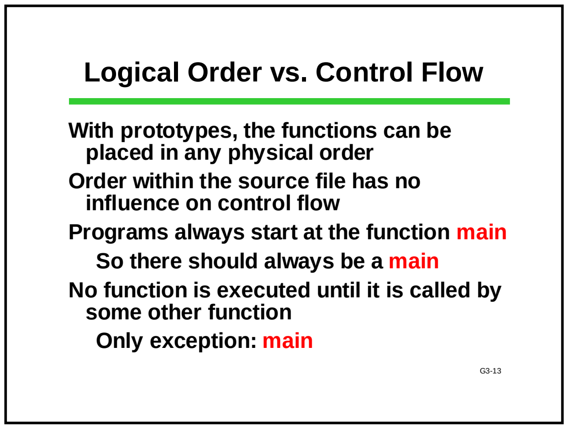
Function prototypes

…

Full function definitions

…

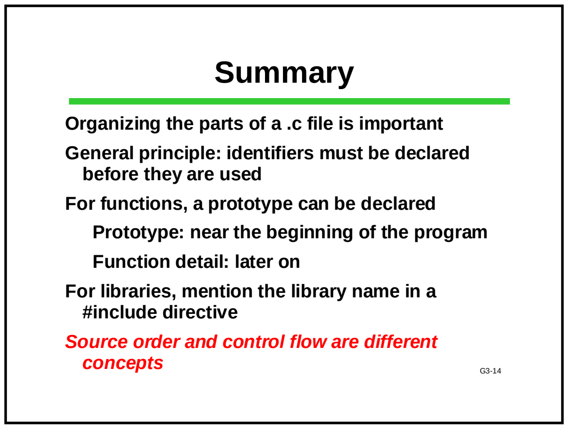
G3-12



Logical Order vs. Control Flow

- With prototypes, the functions can be placed in any physical order
- Order within the source file has no influence on control flow
- Programs always start at the function main
  - So there should always be a main
- No function is executed until it is called by some other function
  - Only exception: main

G3-13



Summary

- Organizing the parts of a .c file is important
- General principle: identifiers must be declared before they are used
- For functions, a prototype can be declared
  - Prototype: near the beginning of the program
  - Function detail: later on
- For libraries, mention the library name in a #include directive
- *Source order and control flow are different concepts*

G3-14