CSSE 2310

Computer Systems Principles + Programming

Today

Week 2.2

More C

School of Information Technology and Electrical Engineering The University of Queensland

CSSE 7231

Switch statements

```
Switch = multi-way decision
Example:
  switch (argc)
  ł
      case 1:
                   /* No argument supplied */
          debug=0;
          break;
      case 2:
                   /* One argument supplied */
          if(strcmp(argv[1], "-debug") == 0)
               /* First argument was -debug
                                              */
          {
               debug=1;
               break;
          /* else drops through */
      default: /* All other cases */
          printf("Usage: %s [-debug]\n", argv[0]);
          exit(1);
                                                 3
  }
```

break and continue statements

Can be used inside loops to alter flow of control

break

CSSE 2310 7231

> terminates execution of innermost while, do, for or switch statement

continue

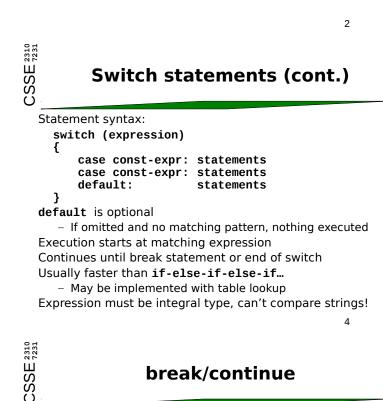
- terminates execution of **body** of innermost while, do or for statement and transfers control to end of body
 - i.e. will perform loop again if conditional allows it

Illustration in class

More C

- Function pointers
- Casting
- Scope
- Storage classes and qualifiers

Linked lists



Unions

CSSE 2310 7231

CSSE 2310 7231

CSSE 2310 7231

union vs struct layout

Like a structure (struct), but can only contain *one* of its elements at a time Example:

- .

CSSE 2310 7231

> union U { double d; char c[2]; int i;

```
};
```

CSSE 2310 7231

Illustration in class

Member access is as for structures

- selection (.)

indirection (->) for pointers to unions

Programmer has to keep track of which type is stored

Function Pointers

Often useful to be able to dynamically choose the function to be called

<pre>- e.g. instead of</pre>	<pre>use</pre>
if(i==1) {	void (*fnArray[NUM])();
fnOne();	/* Declares fnArray to be
} else if (i==2) {	** an array of pointers to
fnTwo();	** functions which return
} else if (i==3) {	** void. */
fnThree();	fnArray[1] = fnOne;
} else	fnArray[2] = fnTwo;
Note: can combine	 Then call with fnArray[i](); or equivalently (*fnArray[i])();

CSSE 2310 7231

Some examples

int (*fp)(int,char*);

 Declares fp to be a pointer to a function which takes int and char* arguments and returns int

void (*fp2[10])(double);

- Declares fp2 to be an array (of size 10) of pointers to functions taking a double parameter and returning nothing
- int (*fp3)();
 - Declares fp3 to be a pointer to a function returning int.
 - Argument types unknown and won't be checked by the compiler.

• Up to programmer to use this correctly. 11

Exercise (4)

What type is var in each of the following? char *var[]; long var[10]; int **var[10]; void (*var)(int,double); int *(*var[5])(); void (*var)(int* (*)(int), int);

You have 2 minutes

8

10

Type casting

Often necessary to convert from one type to another

Some conversions happen automatically

- e.g. function arguments, assignment operations, arithmetic expressions
- · Note: doesn't happen for functions like printf which support variable argument types – Up to programmer to get it right!
- Other conversions require a cast
- e.g. dest = (type-name)source;
- Good to use an explicit cast anyway

13



CSSE 2310 7231

Variable Scope

Scope is the region of a program over which the declaration is visible

Common scopes are

- file scope
 - · visible from declaration point to end of file
- function scope
 - visible from declaration point to end of function includes arguments to function
- block scope
- · visible from declaration point to end of block
- Variable declarations can be hidden

Example to be given in class

15

CSSE 2310

CSSE 72310

Storage classes

C variable declarations have an extent or storage class

– auto

- · Variable has local (automatic) extent, i.e. removed at end of block Permitted within a block only (i.e. not top level)
- This is the default so rarely seen
- extern (for variables or functions)
 - Variable/function is external to all functions, i.e. can be accessed by name by any function
 - Globally accessible linker must know about the name
 - Must be <u>defined</u> once somewhere (can be <u>declared</u> anywhere)
- register
- · Hint to compiler to put variable in a register, otherwise like auto static (for variables or functions)
- Name is only accessible in this file (i.e. not exported to linker)
- · For variables extent is static variable lasts for life of program Examples to be presented in class

Automatic conversions

Automatic conversions		
То	From	
Any real type	Any integer type	
(void *)	 (a) The constant 0 (b) Pointer to object (c) (void*) 	
Pointer to object	(a) The constant 0(b) Pointer to compatible object	
Pointer to function	(a) The constant 0(b) Pointer to compatible function	
CSSE 2310	14	



Type qualifiers

const

- Indicates that the value can't change, e.g.
 - int atoi(const char* str);
 - characters pointed to by str can't be changed
 - const int constant_value = 37;
 - int * const const_pointer;
 - const int * pointer_to_const;

volatile

- Indicates that the value can change in ways not under control of the program
- Often used for interacting with hardware, special memory addresses etc

19