

Section 12

Summary

The C programming language

- general purpose language
- widespread use in industry and education
- developed at Bell Labs
- ANSI standard X3.159.1989
- reference: *The C Programming Language, Second Edition*, Kernighan and Ritchie, Prentice-Hall, 1988
- implementation and system dependencies require vendor documentation

Data types

- integer
- float
- pointer
- array
- struct
- union
- enum
- void

as seen

integers are king, most operators defined to work conveniently with ints

Storage classes and qualifiers

- extern (known to entire program)
- static (known to compilation unit where defined)
- register, auto (local – known in block where defined)
- dynamic (known where explicitly made available)
- const, volatile (read only, modified in unknown ways)

4

1) auto -- default class for local variables, never need the keyword so omitted, register means allocate to a machine register if possible

2) Volatile: modified in a way the compiler cannot know (e.g., interrupt vector)

implications for optimizers flow control, storage alloc, etc.

```
void some_function( int p )
{
    auto char *str; /* as usual */
    register int i; /* in a machine register */
    const double PI = 3.14159;
    volatile long int IOPSW;

}
```

Control structures

- if
- while
- for
- do
- switch
- break
- continue

also goto

Program structure facilities

- functions
- separate compilation
- preprocessor and macro language
- block structure
- scope facilities

functions are important

Standard C library

- **commonly used facilities**
 - string and character manipulation (with international support)
 - floating-point functions (math, conversions)
- **system facilities**
 - input/output
 - memory allocation
 - date and time, resource usage
 - exception handling
- **numerous add-on libraries**
 - access to system-dependent facilities
 - data structures (AVL trees, B-trees, ...)

reference TOC of C library help

Primary benefits of C

- development: easier and more reliable than assembler
- efficiency: close to assembler
- convenience: high-level-language benefits
- portability: available on most systems

9

easier to program

small run-time environment

nice for ASM-class apps, system programs

vendor independence on most platforms

Downside: hard to learn, harder to get good at, easy to write bad, buggy programs