IPC144
Introduction to C Programming

Week-2

Types (and variables)
A Simple Calculation
Expressions

Types…

- C is a "typed programming language"

  Each type has its own rules:
  - How values are stored in memory
  - What operations can be used on that type

- Two common main groups of data types:

Integral	Whole numbers:	`1, 13, 157 …`
Floating-Point	Fractional numbers:	`1.1,13.304, 157.54648 …`

- Thinking back to the components of the CPU:

ALU (Arithmetic and Logic Unit)
  - Integral types only (whole numbers): → **int, char**

FPA (Floating Point Accelerator)
  - Floating-point types only (fractional numbers): → **float, double**

Types…

Smallest unit of storage (RAM): 1 byte (8-bits)

4-Common Data Types

char (1-byte)

int (4-bytes)

Integral → ALU

---

float (4-bytes)

double (8-bytes)

Floating-Point → FPA

CPU

You can further fine-tune memory allocation size by using **SIZE SPECIFIERS**

Types…

Integral Size Specifiers (**int** type only) * *Not applicable to **char***

Long Form	Short Form	Min.Bytes
`short int`	`short`	2
`int`		4
`long int`	`long`	4
`long long int`	`long long`	8

Floating-Point Size Specifier (**double** type only) * *Not applicable to **float***

Form	Min.Bytes
`double`	8
`long double`	8

Specifiers are an available option to ensure a minimum memory allocation size is reserved when the variable is declared

Types...

Integral Types

The ranges of values for the integral types are shown below. Ranges for some types depend on the execution environment:

Type	Size	Min	Max
`char`	8 bits	-128	127
`char`	8 bits	0	255
`short`	>=16 bits	-32,768	32,767
`int`	2 bytes	-32,768	32,767
`int`	4 bytes	-2,147,483,648	2,147,483,647
`long`	>=32 bits	-2,147,483,648	2,147,483,647
`long long`	>=64 bits	-9,223,372,036,854,775,808	9,223,372,036,854,775,807

Note that the limits on both a `char` and an `int` vary with the execution environment.

Taken from IPC144 Course Notes (http://scs.senecacollege.ca/~ipc144/pages/content/varia.html)

Types…

Floating-Point Types

The limits on a `float` and `double` depend on the execution environment:

Type	Size	Significant	Min Exponent	Max Exponent
`float`	minimum	6	-37	37
`float`	typical	6	-37	37
`double`	minimum	10	-37	37
`double`	typical	15	-307	307
`long double`	typical	15	-307	307

Note that both the number of significant digits and the range of the exponent are limited. The limits on the exponent are in base 10.

Taken from IPC144 Course Notes (http://scs.senecacollege.ca/~ipc144/pages/content/varia.html)

Types…

Constant Value (`const`)

- A value which cannot be changed
- Reserved keyword/qualifier: **`const`**

Variable

- A variable is a named placeholder (identifier) for the storage and retrieval of data held in memory

Sample Variable Declarations

[qualifier]	[size-specifier]	Type	Identifier	[=initialization];
`const`	**`long`**	**`int`**	`maxCount`	`= 20000;`
`const`	**`long`**		`maxCount`	`= 20000;`
`const`	**`long`**	**`double`**	`maxGrossAmt`	`= 1000000000L;`
	`short`	**`int`**	`totTrillionaires`	`;`
		`float`	`cost`	`= 0.0f;`

Types…

More Variable Declarations

```
char  children;
int   nPages;
float cashFare;
const double pi = 3.14159265;
```

* Suffix not required (double is the default)

Multiple Variable Declarations

```
char   children, digit;
int    nPages, nBooks, nRooms;
float  cashFare, height, weight;
double loan, mortgage;
```

Taken from IPC144 Course Notes (http://scs.senecacollege.ca/~ipc144/pages/content/varia.html)

Types…

Type: **`char (1 byte or 8-bits)`**

Treated as an integral type (integer) using a collating sequence table where each character or symbol is given a unique integer value

Collating Systems

ASCII (most popular and aligns with newer Unicode standards)
(http://scs.senecacollege.ca/~ipc144/pages/resources/ascii.html)

EBCDIC
(http://scs.senecacollege.ca/~ipc144/pages/resources/ebcdic.html)

Types…

Variable Naming Restrictions

- 1st character must start with:
  - UPPER/lower case letter OR
  - Underscore (_)
- 2nd character onward can contain:
  - UPPER/lower case letters
  - Numbers (0-9)
  - Underscore (_)
- Length of name < 32 characters (to ensure portability)
- Name cannot be a C reserved word

Invalid

```
int 3LegStool = 1;
```

Invalid

```
int three-Leg-Stool = 1;
```

Types…

Reserved Words

The C language reserves the following words for its own use:

```
auto        _Bool       break       case
char        _Complex    const       continue
default     restrict    do          double
else        enum        extern      float
for         goto        if          _Imaginary
inline      int         long        register
return      short       signed      sizeof
static      struct      switch      typedef
union       unsigned    void        volatile
while
```

Taken from IPC144 Course Notes (http://scs.senecacollege.ca/~ipc144/pages/content/varia.html)

Types…

C++ Reserved Words

For upward compatibility with C++, we avoid using the following C++ reserved words

```
asm
bool
catch
class
const_cast
delete
dynamic_cast
explicit
export
false
friend
mutable
namespace
new
operator
private
protected
public
reinterpret_cast
static_cast
template
this
throw
true
try
typeid
typename
using
virtual
wchar_t
```

Taken from IPC144 Course Notes (http://scs.senecacollege.ca/~ipc144/pages/content/varia.html)

Simple Calculation…

Numeric Constants

Type	Suffix	Example
int		1456234
long	L or l	75456234L
long long	LL or ll	75456234678LL
float	F or f	1.234F
double		1.234
long double	L or l	1.234L

Examples

```
const int i = 1456234;                    // No suffix required (default)
const long len = 75456234L;               // Required to enforce long
const long long len = 75456234678LL;      // Required to enforce long long
const float cost = 1.234F;                // Required to enforce float
const double cost = 1.234;                // No suffix required (default)
const long double = 1.234L;               // Required to enforce long
```

Taken from IPC144 Course Notes (http://scs.senecacollege.ca/~ipc144/pages/content/input.html)

Simple Calculation…

Character Constants

- Digit or letter enclosed in SINGLE quotes: `'A'` ***(this is the preferred way)***
- Decimal value (base 10): `65`
  - ASCII table value for uppercase letter A
- Hexidecimal value (base 16): `0x41`
  - ASCII table value for uppercase letter A

Examples

```
char exQuote = 'A';// system will lookup the collating sequence value
char exDec = 65;
char exHex = 0x41;
```

Simple Calculation…

Escape Sequences

- These special constants define actions and symbols
- The blackslash (\) identifies the escape sequence

Character	Sequence	ASCII	EBCDIC
alarm	\a	7	47
backspace	\b	8	22
form feed	\f	12	12
newline	\n	10	37
carriage return	\r	13	13
horizontal tab	\t	9	5
vertical tab	\v	11	11
backslash	\\	92	*
single quote	\'	39	125
double quote	\"	34	127
question mark	\?	63	111

Taken from IPC144 Course Notes (http://scs.senecacollege.ca/~ipc144/pages/content/input.html)

Simple Calculation…

Constant String Literals

- A sequence of characters enclosed in DOUBLE quotes

Example:

```
printf("Welcome to IPC144\n");
```

Also includes an escape sequence for a **NEWLINE** constant character (**\n**)

Simple Calculation…

Input

```
scanf( format,  address );
```

- Procedure used to capture values from the standard input device (keyboard)
- ***format*** argument: The expected input value type
- ***address*** argument: Stores the input value to the specified variable's address

Example:

```
int age;
scanf("%d", &age);
```

Specifier	Input Text	Destination Type
%c	Single character	char
%d	Decimal Number	int, short
%f	Floating-Point Number	float
%lf	Floating-Point Number	double

- Prefix **&** specifies the memory address of a variable
- Age is the variable to store the value to

Simple Calculation…

Output

```
printf( format,  expression );
```

- Procedure used to display values to the standard output device (monitor)
- ***format*** argument: Specifies how to convert data into readable text
- ***expression*** argument: Supplies the data value to be translated to the output device

Example:

```
int age = 19;
printf("Your age is %d", age);
```

Specifier	Output Text	Source Type
%c	Single character	char
%d	Decimal Number	char, int
%f	Floating-Point Number	float
%lf	Floating-Point Number	double

Source variable to be converted to text

NOTE: There is NO (**&**) used as this procedure requires a ***value*** not an address

Expressions…

Expressions

- Should contain at least one operator and one or more operand(s)

Operators (3 most common)

Arithmetic

Relational

Logical

Operands

Can be a combination of variables, constants and other expressions

Expressions…

Arithmetic Operators

Integral *(Binary)*:

Arithmetic Expression			Meaning
operand	+	*operand*	add the operands
operand	-	*operand*	subtract the right from the left operand
operand	*	*operand*	multiply the operands
operand	/	*operand*	divide the left by the right operand
operand	%	*operand*	remainder of the division of left by right

Floating-Point *(Binary)*:

Arithmetic Expression			Meaning
operand	+	*operand*	add the operands
operand	-	*operand*	subtract the right from the left operand
operand	*	*operand*	multiply the operands
operand	/	*operand*	divide the left by the right operand

Unary *Applies to both Integral and Floating-Point*

Arithmetic Expression			Meaning
+	*operand*		evaluates to the operand
-	*operand*		changes the sign of the operand

Taken from IPC144 Course Notes (http://scs.senecacollege.ca/~ipc144/pages/content/expre.html)

Expressions…

Arithmetic Expressions

Order of operator evaluation

1. Brackets: **( )**
2. Multiplication: ***** Division: **/** Modulus: **%**
3. Addition: **+** Subtraction: **-**
4. Assignment: **=**

Expressions…

Relational Operators:

Evaluates to
1 = true
0 = false

Relational Expression			Meaning
operand	==	*operand*	operands are equal in value
operand	>	*operand*	left is greater than the right
operand	>=	*operand*	left is greater than or equal to the right
operand	<	*operand*	left is less than the right
operand	<=	*operand*	left is less than or equal to the right
operand	!=	*operand*	left is not equal to the right

Logical Operators:

Evaluates to
1 = true
0 = false

Expression			Meaning
operand	&&	*operand*	both operands are true
operand	||	*operand*	one of the operands is true
	!	*operand*	the operand is not true

Taken from IPC144 Course Notes (http://scs.senecacollege.ca/~ipc144/pages/content/expre.html)

Expressions…

Shorthand Operators

Integral

Expression	Shorthand	Longhand	Meaning
operand `+=` *operand*	`i += 4`	`i = i + 4`	add 4 to i and assign to i
operand `-=` *operand*	`i -= 4`	`i = i - 4`	subtract 4 from i and assign to i
operand `*=` *operand*	`i *= 4`	`i = i * 4`	multiply i by 4 and assign to i
operand `/=` *operand*	`i /= 4`	`i = i / 4`	divide i by 4 and assign to i
operand `%=` *operand*	`i %= 4`	`i = i % 4`	remainder after i/4 and assign to i

Taken from IPC144 Course Notes (http://scs.senecacollege.ca/~ipc144/pages/content/expre.html)

Expressions…

Shorthand Operators

Floating-Point

Expression	Shorthand	Longhand	Meaning
operand `+=` *operand*	`x += 4.1`	`x = x + 4.1`	add 4.1 to x and assign to x
operand `-=` *operand*	`x -= 4.1`	`x = x - 4.1`	subtract 4.1 from x and assign to x
operand `*=` *operand*	`x *= 4.1`	`x = x * 4.1`	multiply x by 4.1 and assign to x
operand `/=` *operand*	`x /= 4.1`	`x = x / 4.1`	divide x by 4.1 and assign to x

Taken from IPC144 Course Notes (http://scs.senecacollege.ca/~ipc144/pages/content/expre.html)

Expressions…

Unary Operators

Integral

Expression	Shorthand	Longhand	Meaning
++*operand*	++i	i = i + 1	increment i by 1
operand++	i++	i = i + 1	increment i by 1
--*operand*	--i	i = i - 1	decrement i by 1
operand--	i--	i = i - 1	decrement i by 1

Floating-Point

Expression	Shorthand	Longhand	Meaning
++*operand*	++x	x = x + 1	increment x by 1.0
operand++	x++	x = x + 1	increment x by 1.0
--*operand*	--x	x = x - 1	decrement x by 1.0
operand--	x--	x = x - 1	decrement x by 1.0

Taken from IPC144 Course Notes (http://scs.senecacollege.ca/~ipc144/pages/content/expre.html)

Expressions…

Conversion

- **Explicit**
  - Casting (programmer specifies the type)

Example

```
float balance = 5.75f;
int loonies = (int)balance;
```

- **Implicit**
  - Coercion/implied (rely on compiler and standards to determine the type)

Expressions…

Implicit

Combining multiple types in an expression without casting

Example

```
int minutes = 45;
float hours = minutes / 60;
// hours = 0.0!!! (int/int)
```

Possible Fixes:

```
float hours=minutes/60.0f;
float hours=(float)minutes/60;
```

	Type	Size (Bytes)	Priority
Floating-Point	long double	* 8+	*Highest*
	double	8	
	float	4	
Integral Types	long long	8	
	long	4	
	int	4	
	short	2	
	char	1	*Lowest*

* ***NOTE:*** *Size of 'long double' is not consistant across environments (determined by the compiler) but will be at least the size of a 'double'*

Expressions…

Conversion

- **Promotion**

```
double balance;
int loonies = 50;
balance = loonies; // loonies gets promoted to double
```

- **Narrowing**

```
double balance = 57.75;
int loonies;
loonies = balance; // balance gets truncated to an int (57)
```