Foundations of C Programming (Structured Programming)

- Primary data types and variables

Outline

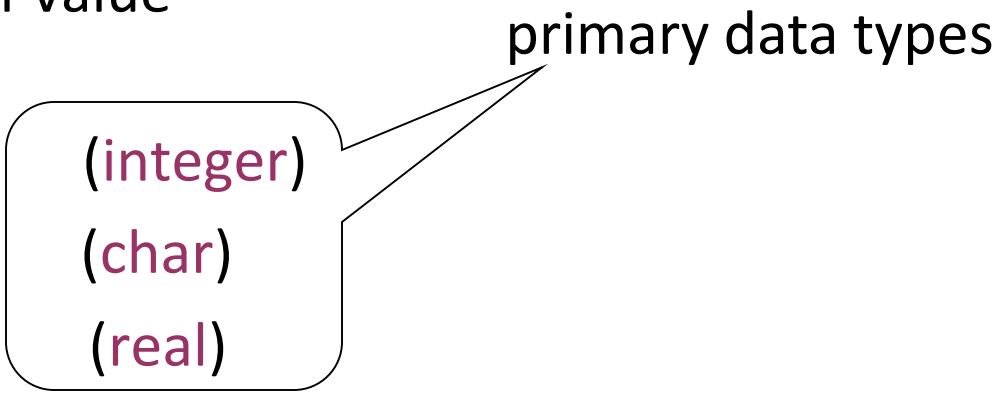
- Values
- Primary data types
- Number representations
- Identifier
- Keywords
- Variables
- Declaration

Values

- There are different types of value
  - E.g.,
  - Age: 19 (integer)
  - Gender: 'm' or 'f' (char)
  - Weight: 82.5 (kg) (real)
  - Name: "Tommy" (string)
  - Time: 13:25:16 (structure)

Values

- There are different types of value
  - E.g.,
  - Age: 19 (integer)
  - Gender: 'm' or 'f' (char)
  - Weight: 82.5 (kg) (real)
  - Name: "Tommy" (string)
  - Time: 13:25:16 (structure)



Primary Data Types

- int
  - Used to express the integer type.
- char
  - Used to express the single characters. Each character corresponds to an integer between 0 and 127
- float
  - Real number (single precision float point)
- double
  - Real number (double precision float point)
- _Bool
  - A Boolean value 0 or 1

int

- Used to express integer values.
- An integer can be a natural number (including 0) or a negative number
  - E.g., 10, 20, 10000
  - Can be expressed in
    - Decimal (base-10)
    - Binary (base-2)
    - Hexadecimal (base-16)
    - Octal (base-8) (will not be introduced in this lecture, learn from the Internet after class)

Decimal

- Decimal number system is also called
  - base-10 number system
- 10 digits
  - 0, 1, 2, 3, 4, 5, 6, 7, 8, 9
- Counting in base-10
  - 0, 1, 2, …, 9, 10, 11, …, 19, 20, 21, …, 99, 100, …

234

$= 2\times10^{2} + 3\times10^{1} + 4\times10^{0}$

Hexadecimal

- Hexadecimal number system is also called
  - base-16 number system
- 16 digits
  - 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F
- Counting in base-16
  - 0, 1, …, 9, A, …,F, 10, 11, …, 19, 1A, …, 1F, 20,…, FF, 100, …
  - 0x is used to indicate base-16.
  - Hexadecimal to decimal

$0x10A$

$= 1\times16^2 + 0\times16^1 + 10\times16^0$

$= 266_{(10)}$

Binary

- Binary number system is also called
  - base-2 number system
- Binary number system has only two digits
  - 0, 1
- Counting in binary system
  - 0, 1, 10, 11, 100, 101, 110, 111, 1000,….
- Binary to decimal

$$101_{(2)}$$
$$= 1\times2^2 + 0\times2^1 + 1\times2^0$$
$$= 5_{(10)}$$

0-15: Decimal, Binary, Hexadecimal

Decimal	4-Bit Binary	Hexadecimal
0	0000	0
1	0001	1
2	0010	2
3	0011	3
4	0100	4
5	0101	5
6	0110	6
7	0111	7
8	1000	8
9	1001	9
10	1010	A
11	1011	B
12	1100	C
13	1101	D
14	1110	E
15	1111	F

Binary: Bit and Byte

- In the source code written in an advanced language
  - Decimal (base-10) is used (sometimes, hexadecimal or octal)
- Binary is used inside machine. All the code and data in a program will be converted into binary by compiler.
  - bit: binary digit
  - 1 Byte = 8 bits
    - the basic unit of addressable memory

Sign Bit in Binary Representation

- There is a bit to represent sign of an integer
- Positive integer and zero
  - The left first bit is 0
    - E.g. ,
      - 1: 0000 0001
      - 28: 0001 1100
- Negative integer
  - The left first is 1
  - E.g.,
    - -127: 1000 0001
    - -28: 1110 0100

How to represent negative numbers in binary is introduced in other course.

int types

- short int
  - 2 bytes, -2^{15} (-32768) ~ $2^{15} - 1$ (32767)
  - E.g., 12, 20
- int
  - 4 bytes, -2^{31} ~ $2^{31} - 1$
  - 20, 12, 60000
- long int
  - 4 or 8 bytes, -2^{31} ~ $2^{31} - 1$ or -2^{63} ~ $2^{63} - 1$
  - 20, 12l, 70000
- long long int
  - 8 bytes, -2^{63} ~ $2^{63} - 1$
  - E.g., 20, 12ll, 0xffll

Unsigned Int

- unsigned int
  - 4 bytes, $0 \sim 2^{32} - 1$
  - E.g., 20, 12u, 0xffu
- unsigned integer
  - There is no bit for sign of an integer
  - Used for only positive integers
  - E.g.,
    - 0000 0001: 1
    - 1000 0001: 129 (-127 in the signed int)

Overflow

- Overflow occurs when the value exceeds the range that computer can represent.
  - For example, if each value is stored using 8 bits and the first digit is for sign, that is , the range is from -128 to 127. Then adding 127 to 3 causes overflow.

```
  0 1 1 1 1 1 1 1
+ 0 0 0 0 0 0 1 1
  1 0 0 0 0 0 1 0
    -126 ??
```

```
  127
+   3
  130
```

```
int main()
{
   char c1 = 127, c2 = 3, c3;

   c3 = c1 + c2;
   printf("The sum is %d\n", c3);
   return 0;
}
```

⟹

Console program output

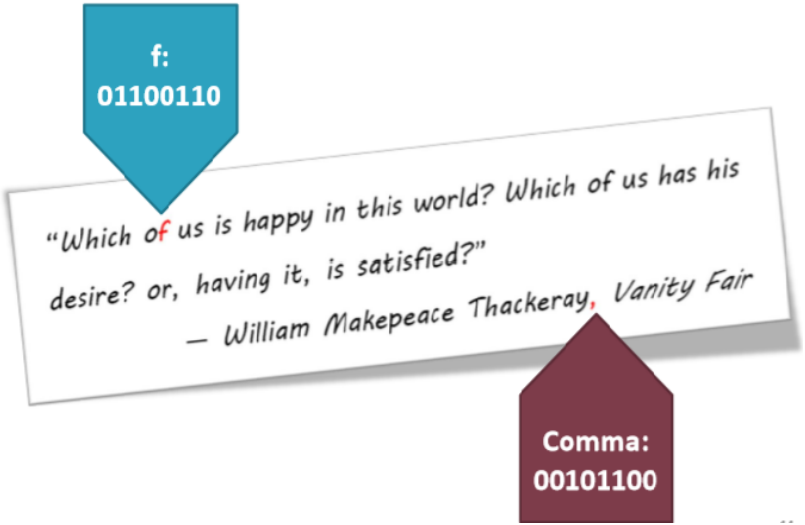
```
The sum is -126
Press any key to continue...
```

Primary Data Types

- int
  - Used to express the integer type. The biggest integer that can be expressed in a computer depends on the host computer (32 bits or 64 bits)
- char
  - Used to express the single characters. Each character corresponds to an integer between 0 and 127
- float
  - Real number (single precision float point)
- double
  - Real number (double precision float point)
- _Bool
  - A Boolean value 0 or 1

Characters

- A text document can be decomposed into chapters, paragraphs, sentences, words, and ultimately, individual characters
- Each character should be represented in machine. A character could be
  - 'a', 'b', 'c', 'd', …
  - '1', '2', '3', '4', …
  - '+', '-', '*', '/', …
  - '高', '天', '孤', '月'
- ASCII code or Unicode is used to represent characters.



ASCII Code

Char	Dec	Oct	Hex	Char	Dec	Oct	Hex	Char	Dec	Oct	Hex	Char	Dec	Oct	Hex
(nul)	0	0000	00	(sp)	32	0040	20	@	64	0100	40	`	96	0140	60
(soh)	1	0001	01	!	33	0041	21	A	65	0101	41	a	97	0141	61
(stx)	2	0002	02	"	34	0042	22	B	66	0102	42	b	98	0142	62
(etx)	3	0003	03	#	35	0043	23	C	67	0103	43	c	99	0143	63
(eot)	4	0004	04	$	36	0044	24	D	68	0104	44	d	100	0144	64
(enq)	5	0005	05	%	37	0045	25	E	69	0105	45	e	101	0145	65
(ack)	6	0006	06	&	38	0046	26	F	70	0106	46	f	102	0146	66
(bel)	7	0007	07	'	39	0047	27	G	71	0107	47	g	103	0147	67
(bs)	8	0010	08	(	40	0050	28	H	72	0110	48	h	104	0150	68
(ht)	9	0011	09	)	41	0051	29	I	73	0111	49	i	105	0151	69
(nl)	10	0012	0a	*	42	0052	2a	J	74	0112	4a	j	106	0152	6a
(vt)	11	0013	0b	+	43	0053	2b	K	75	0113	4b	k	107	0153	6b
(np)	12	0014	0c	,	44	0054	2c	L	76	0114	4c	l	108	0154	6c
(cr)	13	0015	0d	-	45	0055	2d	M	77	0115	4d	m	109	0155	6d
(so)	14	0016	0e	.	46	0056	2e	N	78	0116	4e	n	110	0156	6e
(si)	15	0017	0f	/	47	0057	2f	O	79	0117	4f	o	111	0157	6f
(dle)	16	0020	10	0	48	0060	30	P	80	0120	50	p	112	0160	70
(dc1)	17	0021	11	1	49	0061	31	Q	81	0121	51	q	113	0161	71
(dc2)	18	0022	12	2	50	0062	32	R	82	0122	52	r	114	0162	72
(dc3)	19	0023	13	3	51	0063	33	S	83	0123	53	s	115	0163	73
(dc4)	20	0024	14	4	52	0064	34	T	84	0124	54	t	116	0164	74
(nak)	21	0025	15	5	53	0065	35	U	85	0125	55	u	117	0165	75
(syn)	22	0026	16	6	54	0066	36	V	86	0126	56	v	118	0166	76
(etb)	23	0027	17	7	55	0067	37	W	87	0127	57	w	119	0167	77
(can)	24	0030	18	8	56	0070	38	X	88	0130	58	x	120	0170	78
(em)	25	0031	19	9	57	0071	39	Y	89	0131	59	y	121	0171	79
(sub)	26	0032	1a	:	58	0072	3a	Z	90	0132	5a	z	122	0172	7a
(esc)	27	0033	1b	;	59	0073	3b	[	91	0133	5b	{	123	0173	7b
(fs)	28	0034	1c	<	60	0074	3c	\	92	0134	5c	|	124	0174	7c
(gs)	29	0035	1d	=	61	0075	3d	]	93	0135	5d	}	125	0175	7d
(rs)	30	0036	1e	>	62	0076	3e	^	94	0136	5e	~	126	0176	7e
(us)	31	0037	1f	?	63	0077	3f	_	95	0137	5f	(del)	127	0177	7f

Characters and Code

- ASCII stands for American Standard Code for Information Interchange
  - Designed for English
  - 0-31: for control characters, cannot be displayed
  - Uses 8 bits to represent a character
  - E.g.,
    - 'a': 97 ('b': 98 ..... inferred from the code of 'a')
    - 'A': 65 ('B': 66 ..... inferred from the code of 'A')
    - '0': 48 ('1': 49 ..... inferred from the code of '0')
- Unicode
  - include ASCII
  - Can represent international languages
    - E.g., '高', '天', '孤', '月'

Tips: Remembering ASCII code for 'a', 'A', '0' is helpful.

char

- char
  - 1 byte, - 2^7 ~ 2^7 – 1
  - E.g., 'a', '1', '+', ' '
- unsigned char
  - 1 byte, 0 ~ 2^8 - 1
- Attention
  - '1' is different from 1
  - '+' is different from +
  - 'a' is different from a
  - 'a' is different from "a"
- Every char type value corresponds to an ASCII code

Attention: ' ' (English mode) and ' '(Chinese mode) are different. C program accepts English mode.

Primary Data Types

- int
  - Used to express the integer type. The biggest integer that can be expressed in a computer depends on the host computer (32 bits or 64 bits)
- char
  - Used to express the single characters. Each character corresponds to an integer between 0 and 127
- float
  - Real number (single precision float point)
- double
  - Real number (double precision float point)
- _Bool
  - A Boolean value 0 or 1

float

- Float
  - 4 bytes
  - E.g., 1.2, 2.5e8 (Scientific notation. 2.5×10^8)
  - Absolute value: 1.2×10^{-38} ~ 3.4×10^{38} (1.2e-38 ~ 3.4e38)
  - IEEE 754 standard: 1 bit sign, 8 bits exponent, 23 bits mantissa
- Double
  - 8 bytes
  - Absolute value: 2.2×10^{-308} ~ 1.8×10^{308} (2.2e-308 ~ 1.8e308)
  - IEEE 754 standard: 1 bit sign, 11 bits exponent, 52 bits mantissa

*Note: float or double cannot express all real numbers precisely in the range. Refer to the resources online about how float numbers are represented.

Primary Data Types

- int
  - Used to express the integer type. The biggest integer that can be expressed in a computer depends on the host computer (32 bits or 64 bits)
- char
  - Used to express the single characters. Each character corresponds to an integer between 0 and 127
- float
  - Real number (single precision float point)
- double
  - Real number (double precision float point)
- _Bool
  - A Boolean value 0 or 1

_Bool

- A Boolean value takes value 0 or 1
- It is usually used to express the comparison result
  - 0: false
  - 1: true
- It is rarely directly used in program

Identifiers (Variable Names)

- An identifier consists of a letter or underscore followed by any sequence of letters, digits or underscores
  - E.g.,
    - _Is, This_Is, A12, a23 are valid identifiers
    - X=Y, J-20, #007 are invalid identifiers
- Names are case-sensitive! The following are unique identifiers:
  - Hello, hello,
  - whoami, whoAMI, WhoAmI
- C keywords (reserved words) cannot be used as identifiers.

C Keywords

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

Class Exercises

- Are these the valid variable names?
  - _123
  - _abc
  - Example
  - Abc123
  - unsigned
  - int
  - a%b
  - 2example
  - Xx

Meaningful Identifiers

- Choose identifiers that are meaningful and easy to remember
- Good identifiers can make program readable
  - For example, *grade*, *student*, *record*, *id*, *name* are good identifiers used in a program which handles students' information.
  - *i, j, k, m, n*: usually for counting numbers
    - n1, n2 … can be used too
  - *c, ch*: usually used to store char values
  - *f*: usually used to store float numbers
  - *aaa*, *bbb*, *ccc* are not good identifiers

Attention: random naming of identifiers might cause marks lost in assignments and lab exercises.

Declaration and Assignment

- Every variable used in a program must declare its type
  - Format: TYPE variable_name _list;
  - E.g.,
    - int i;
    - float f;
    - double area;
    - unsigned int number;
    - int number, index, grade;

Declaration and Assignment

- The variables can be assigned values using the assignment operator
  - Format: variable_name = value;
  - E.g.,
    - i = 10;
    - f = 1.2;
    - area = 6.28 ;
    - area = f;

Declaration and Assignment

```
int i;
char c;
float f;

i = 28;
c = 'a';
f = 28.0;
```

declaration

assignment

Declaration and Assignment

```
int i1;
char 2c
float f;

i1 = 28.5;
2c = '*';
f = 28
```

What problems are there in the code?

Type Conversion

- C allows for conversions between the basic types, implicitly or explicitly.
- Explicit conversion uses the cast operator.

```
int x = 10;
float y = 3.14,z = 3.14;
y = (float)x;      /* y = 10.0 */
x = (int)z;        /* x = 3     */
x = (int)(-z);     /* x = -3 - rounded approaching zero */
y = x;            /* y = ??? */
```

cast operator

Implicit Conversion

- If the compiler expects one type at a position, but another type is provided, then implicit conversion occurs.

```
int x = 10;
float y = 3.14,z = 3.14;
y =(float)x;  /* y = 10.0 */
x =(int)z;    /* x = 3     */
x =(int)(-z); /* x = -3 - rounded approaching zero */
y = x;          /* y = -3.0 */
```

Implicit conversion

Implicit Conversion

- If the compiler expects one type at a position, but another type is provided, then implicit conversion occurs.
- ASCII code and character can be used alternatively.

```
char c = 'a';
int i;
i = c;    /* i = 97, ASCII code of 'a'*/
```

Type: char

Type: int

Attention: It is better to use character constant rather than integer constant for char type. E.g., c = 'a' is more readable than c = 97.

Summary

- Basic data types
- Different number systems are used in programming
- Data of different types can be converted to each other sometimes
- Meaningful identifier can make program readable
- Each character has an ASCII code