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

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)， e.g., 29
    - Hexadecimal (base-16), e.g., 0x1D
    - Octal (base-8), e,g, 041

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 ~ $2^{32} - 1$
  - 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.

$$
\begin{array}{r}
0\ 1\ 1\ 1\ 1\ 1\ 1\ 1 \\
+\ 0\ \underline{0\ 0\ 0\ 0\ 0\ 1\ 1} \\
1\ 0\ 0\ 0\ 0\ 0\ 1\ 0
\end{array}
\qquad\qquad
\begin{array}{r}
127 \\
+\ \underline{\quad 3} \\
130
\end{array}
$$

-126 ??

```
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 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')

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

char

- char
  - 1 byte, $-2^7 \sim 2^7 - 1$
  - E.g., 'a', '1', '+', ' '
- unsigned char
  - 1 byte, $0 \sim 2^8 - 1$
- 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;       declaration
float f;

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

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