Tin học cơ sở 4

Data Types and Variables

Outline

- Basic built-in data types
- Variable and Constants
- Operators

Example

Khai báo 2 biến số nguyên, “a” và “b”

Nhập 2 số nguyên vào a và b

Viết các biểu thức “a”, “b” và “a-b” theo định dạng %i

```
#include <stdio.h>

int  main(void)
{
     int   a, b;

     printf(“Nhap 2 so nguyen: ");
     scanf("%i %i", &a, &b);

     printf("%i - %i = %i\n", a, b, a - b);

     return 0;
}
```

```
Nhap 2 so nguyen: 21 17
21 - 17 = 4
```

Lệnh xuất - printf

Xuất dữ liệu ra màn hình:

```
printf("%i - %i = %i\n", a, b, a - b);
```

- Các ký tự hằng được in nguyên văn
- Các ký tự định dạng được thay bằng giá trị của biểu thức tương ứng:
- %i: ký tự định dạng số nguyên kiểu int
- Các ký tự điều khiển: \n – xuống dòng; \t – dấu tab; \\ – dấu \; \“– dấu “ …
- Thư viện: `stdio.h`

Lệnh nhập - scanf

Nhập dữ liệu từ bàn phím

```
scanf("%i %i", &a, &b);
```

- Trong chuỗi định dạng chỉ có ký tự định dạng và khoảng trắng.
- Dữ liệu phải được nhập vào các biến.
- Trước tên biến phải ghi dấu & - toán tử địa chỉ. Nếu không có toán tử địa chỉ, giá trị của biến sẽ không được cập nhật
- Thư viện: `stdio.h`

Variables

- **Variables** are *containers* that hold values.
- Variables are implemented as memory cells.
- Each variable has three properties:
  - Name (used to refer to it)
  - Type (set of possible values)
  - Current value (changes as program runs)
- Variable are defined by giving name, type and optionally an initial value.

int i, j, k;

double length, width;

Names

- Names are used to identify variables and other components of C programs. They must start with a letter followed by letters, digits or both. The characters '_' may also appear in names.
- Some examples: i, k53, X, nchars, numberOfChars, number_of_chars.
- Upper and lower case are distinguished.
- Use meaningful names.
- C/C++ has reserved words (keywords) that you cannot use as names.

C keywords

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

C++ keywords

and	*and_eq*	*asm*	*auto*	*bitand*	*bitor*
bool	*break*	*case*	*catch*	*char*	*class*
compl	*const*	*const_cast*	*continue*	*default*	*delete*
do	*double*	*dynamic_cast*	*else*	*enum*	*explicit*
export	*extern*	*false*	*float*	*for*	*friend*
goto	*if*	*inline*	*int*	*long*	*mutable*
namespace	*new*	*not*	*not_eq*	*operator*	*or*
or_eq	*private*	*protected*	*public*	*register*	*reinterpret_cast*
return	*short*	*signed*	*sizeof*	*static*	*static_cast*
struct	*switch*	*template*	*this*	*throw*	*true*
try	*typedef*	*typeid*	*typename*	*union*	*unsigned*
using	*virtual*	*void*	*volatile*	*wchar_t*	*while*
xor	*xor_eq*				

Basic data types

- Integral types.
- Floating-point types.
- Boolean

Integral Types

Name	#bytes	Format	Min Val	Max Val
char	1	%c	CHAR_MIN	CHAR_MAX
unsigned char	1	%c	0	UCHAR_MAX
short	2	%hi	SHRT_MIN	SHRT_MAX
unsigned short	2	%hu	0	USHRT_MAX
int	4	%i	INT_MIN	INT_MAX
unsigned int	4	%u	0	UINT_MAX
long	8	%li	LONG_MIN	LONG_MAX
unsigned long	8	%lu	0	ULONG_MAX

The number of bytes (range of values) of each type varies between implementations.
Max/Min values are defined in limits.h

For most purposes use *int*.

int Example

```
#include <stdio.h>
#include <limits.h>

int  main()
{
     unsigned long   big = ULONG_MAX;

     printf("minimum int = %i, ", INT_MIN);
     printf("maximum int = %i\n", INT_MAX);
     printf("maximum unsigned = %u\n", UINT_MAX);
     printf("maximum long int = %li\n", LONG_MAX);
     printf("maximum unsigned long = %lu\n", big);

     return 0;
}
```

```
minimum int = -32768, maximum int = 32767
maximum unsigned = 65535
maximum long int = 2147483647
maximum unsigned long = 4294967295
```

Char example

In ra mã ASCII của ký tự

```
#include <stdio.h>
#include <limits.h>

int main()
{
    char  lower_a = 'a';
    char  lower_m = 'm';

    printf("minimum char = %i, ", CHAR_MIN);
    printf("maximum char = %i\n", CHAR_MAX);

    printf("Sau '%c' la '%c'\n", lower_a, lower_a + 1);
    printf("Ky tu in hoa '%c'\n", lower_m - 'a' + 'A');

    return 0;
}
```

Trong NNLT C, ký tự chính là số nguyên

```
minimum char = -128, maximum char = 127
Sau 'a' la 'b'
Ky tu in hoa 'M'
```

Số nguyên trong các cơ số khác

- Các hệ cơ số có thể thực hiện được: cơ số 8 (octal), cơ số 10 (decimal), cơ số 16 (hexadecimal)

Số 0: số octal

0x: số hexadecimal

```
#include <stdio.h>

int main(void)
{
    int   dec = 20, oct = 020, hex = 0x20;

    printf("dec=%d, oct=%d, hex=%d\n", dec, oct, hex);
    printf("dec=%d, oct=%o, hex=%x\n", dec, oct, hex);

    return 0;
}
```

```
dec=20, oct=16, hex=32
dec=20, oct=20, hex=20
```

Floating-Point Types

Name	#bytes	format	Max val	Min val
float	4	%f %e %g	FLT_MIN	FLT_MAX
double	8	%lf %le %lg	DBL_MIN	DBL_MAX
long double	12	%Lf %Le %Lg	LDBL_MIN	LDBL_MAX

- The number of bytes (range of values) of each type varies between implementations.
- For most purposes use *double*.
- Floating point numbers appearing in the text of a program are, by default, of type double
- Max/Min values are defined in "float.h"

Double example

```
#include <stdio.h>
#include <float.h>

int  main(void)
{
     double f = 3.1416, g = 1.2e-5, h = 5000000000.0;

     printf("f=%lf\tg=%lf\th=%lf\n", f, g, h);
     printf("f=%le\tg=%le\th=%le\n", f, g, h);
     printf("f=%lg\tg=%lg\th=%lg\n", f, g, h);

     printf("f=%7.2lf\tg=%.2le\th=%.4lg\n", f, g, h);

     return 0;
}
```

```
f=3.141600          g=0.000012          h=5000000000.000000
f=3.141600e+00      g=1.200000e-05      h=5.000000e+09
f=3.1416            g=1.2e-05           h=5e+09
f=   3.14           g=1.20e-05          h=5e+09
```

Boolean Type and void

- In C:
  - No boolean type.
  - The type *int* is used for boolean values. 0 represents *false*, all other values are considered to represent *true*.
- *void*:
  - Function that does not return any values
  - Generic type using *type casting*.

Literals

0	Integer 0	int
3.14159	Approximation to pi	double
1.414	Square root of 2	double
1.414F	Square root of 2	float
‘a’	The letter a	char
‘1’	The digit 1	char
‘\n’	Newline character	char
0x7fff	Hexadicimal number	int
10000L	Ten thousand	long
“Hello there”	string	Array of char

Constants

- Variables – their values can vary as the program executes.
- PI or speed of light is different because its value is *constant*.
- There are two ways to handle constants.

Constant Declarations

- We can declare a constant variable:

const float SPEED_OF_LIGHT = 1079252848.8;

Or using scientific notation:

const float SPEED_OF_LIGHT = 1.079e+9;

- Variables declared with the *const* qualifier cannot be altered.
- Attempting to alter a *const* variable will generate a compilation error.

Constant example

Các hằng pi, days_in_week, sunday được tạo với từ khóa const

```
#include <stdio.h>

int main(void)
{
    const long double PI = 3.141592653590L;
    const int DAYS_IN_WEEK = 7;
    const SUNDAY = 0;

    DAYS_IN_WEEK = 5;

    return 0;
}
```

Lỗi

Symbolic Constants

- Alternatively, we can define a symbolic constant at the top of the file:

#define SPEED_OF_LIGHT 1079252848.8

#define name replacementText

- The compiler's pre-processor will parse your code replacing all occurrences of name with replacementText.
- It will not make the replacement if name is inside quotes or part of another name.
- Use with caution
- Use constants to avoid burying "magic numbers" or values in the code.

Symbolic Constant Example

Tìm từ "PI", thay bằng 3.1415....

```
#include <stdio.h>

#define PI              3.141592653590L
#define DAYS_IN_WEEK    7
#define SUNDAY          0

int  day = SUNDAY;
long flag = USE_API;
```

Lưu ý: không có "=" và ";"

Không thay thế "PI"

Naming Conventions

- It is helpful to use descriptive names for your variables, which make the code easier to read and understand.
- Many library functions begin their names with an underscore '_'. For this reason, you should avoid starting functions or variable names with an underscore.
- Some like to capitalize the first letter of each word: CountWord()
- Some like to use all lowercase, with underscores: word_count
- Be consistent
- Constants are defined in full uppercase.

Assignment

- Procedural programming is about state change.
- **Assignment** is the basic state-change operation.
- Syntax: *var = expression;*

i = 0;

- If the value is not quite the right type, it may be converted.
- Variables can also be ***initialised*** when they are declared:

char c = 'A';

int x = 10;

Assignment

- Variables can be modified during a program:

y = x + 5;

x = x + 3;

- The expression to the right of = is evaluated and the result stored in the variable to the left of the =
- Variables must be assigned before they are used; otherwise they will get “garbage” values;
- Depending on implementation, “garbage” values typically have default of 0 but you should make any assumptions.

Arithmetic Operators

Operator	Use	Description
+	op1 + op2	
-	Op1 - op2	
*	Op1 * op2	
/	Op1 / op2	Divide op1 by op2. Ignore remainder for integer division
% (modulus)	Op1 % op2	Computes the remainder of dividing op1 by op2. Can't be used on float.

Ví dụ về toán tử chia "/"

- Trình biên dịch dựa vào kiểu của các toán hạng để quyết định phép chia tương ứng

"i", "j" kiểu int, "/" là phép chia lấy nguyên → k nhận giá trị 1

"f", "g" kiểu double, "/" là phép chia số thực → h nhận giá trị 1.25

Phép chia nguyên, bất kể "h" có kiểu double. Kết quả là 1.00000

```
int  main(void)
{
    int     i = 5,   j = 4,   k;
    double  f = 5.0, g = 4.0, h;

    k = i / j;
    h = f / g;
    h = i / j;

    return 0;
}
```

Increment and Decrement Operators

- Increment ++: add 1
- Decrement --: subtract 1
- Both can appear either before or after a variable. Recommend to put them after the variable.

```
                    // suppose k = 7 initially
n = k--;            // first assign k to n, then decrement k;
                    // afterwards, k = 6 but n = 7

                    // suppose k = 7 initially
n = --k;            // first decrement k, then assign k to n
                    // afterwards, k = 6 and n = 6
```

Example

Thứ tự thực hiện các toán tử ++ và -- phụ thuộc vào vị trí của chúng (trước hay sau) so với biến:

```
#include <stdio.h>

Int main(void)
{
    int   i, j = 5;
    i = ++j;
    printf("i=%d, j=%d\n", i, j);
    j = 5;
    i = j++;
    printf("i=%d, j=%d\n", i, j);

    return 0;
}
```

Tương đương:
1. j++;
2. i = j;

Tương đương:
1. i = j;
2. j++;

```
i=6, j=6
i=5, j=6
```

Assignment Operators

x += 10

is an abbreviation for

x = x + 10

We can also use

-= *= /= %=

- Has the value of the variable's value

a = b = 10;

Comparison Operators

Operator	Use	Return true if
>	Op1 > op2	op1 is greater than op2
>=	Op1 >= op2	op1 is greater than or equal to op2
<	Op1 < op2	op1 is less than op2
<=	Op1 <= op2	op1 is less than or equal to op2
==	Op1 == op2	op1 and op2 are equal
!=	Op1 != op2	op1 and op2 are not equal

Boolean operators

Operator	Use	Returns true if
&&	Op1 && op2	op1 and op2 are both true, conditionally evaluates op2
ll	Op1 ll op2	either op1 or op2 is true, conditionally evaluates op2
!	!op	op is false

Evaluating expressions

- When evaluating an expression, operators of higher precedence are evaluated first.
- For example, ‘*’ is evaluated before ‘+’.
- Use parentheses to explicitly indicate how expression should be evaluated.
- Don’t worry about using unnecessary parentheses.
- When precedence is equal, expression are evaluated left to right order, except assignment operator.

Precedence Levels

parentheses	**()**
unary operators	*++expr –expr !*
multiplicative	* / %
additive	+ -
shift	<< >>
relational	< > <= >=
equality	== !=
Bitwise operators	& ^ |
Logical operators	&& ||
Conditional	?:
Assignment	= += -= *= /= %= &= ^= | = <<= >>= >>>=

sizeof

sizeof(Obj)

- Cho biết kích thước của đối tượng theo đơn vị byte

```
#include <stdio.h>

int main(void)
{
    long   big;

    printf("\"big\" is %u bytes\n", sizeof(big));
    printf("a short is %u bytes\n", sizeof(short));
    printf("a double is %u bytes\n", sizeof (double));

    return 0;
}
```

```
"big" is 4 bytes
a short is 2 bytes
a double is 8 bytes
```

References

- [K&R], Chapter 2