CSC10001 – Introduction to Programming

2nd lecture: Basic elements of C++ program

fit hcmus
Instructor: TRAN Thi Thao Nhi

October 10, 2023

Program Paradigms

```
1  //This program calculates the user's age
2  #include <iostream>
3  using namespace std;
4
5  int main(){
6      int year, age;
7      // Get the year born
8      cout << "When were you born? ";
9      cin >> year;
10     // Calculate the age
11     age = 2023 - year;
12     // Display the age
13     cout << "You are " << age << endl;
14     return 0;
15 }
```

- special characters
- key words
- preprocessor directives
- stream input/output
- variables
- data types
- arithmetic operators

Special characters

	Name	Use for
//	double slash	comment
#	pound sign	preprocessor directive
< >	brackets	#include <filename>
( )	parentheses	naming function
{ }	braces	group of statements
" "	quotation marks	string
;	semicolon	ending statement
,	comma	separating things

Key words

- lowercase (C++)
- special meaning
- intended purpose

C++ key words	
using	namespace
int	return
and	auto
bool	break
case	const
do	double
else	enum
false	float
...	

[Table 2-4, chapter 2]

#include

The #include directive causes the contents of another file to be inserted into the program

- prototype: #include <filename>
- automatically insert the contents of filename (header file)
- not C++ statements → no semicolons

func.cpp

home > nhi > teach > csc101 > func.cpp

```
1  // #include <iostream>
2  #include <string>
3  using namespace std;
4
5  int main(){
6      int year, age;
7      // Get the year born
8      cout << "When were you born? ";
9      cin >> year;
10     // Calculate the age
11     age = 2023 - year;
12     // Display the age
13     cout << "You are " << year << "\n";
14     return 0;
15 }
```

PROBLEMS OUTPUT TERMINAL ··· bash

```
func.cpp: In function 'int main()':
func.cpp:8:5: error: 'cout' was not declared in this scope
    8 |     cout << "When were you born? ";
      |     ^~~~
func.cpp:3:1: note: 'std::cout' is defined in header '<iostream
>'; did you forget to '#include <iostream>'?
    2 | #include <string>
  +++ |+#include <iostream>
    3 | using namespace std;
func.cpp:9:5: error: 'cin' was not declared in this scope
    9 |     cin >> year;
      |     ^~~
func.cpp:9:5: note: 'std::cin' is defined in header '<iostream>
'; did you forget to '#include <iostream>'?
(base) nhi@nhi:~/teach/csc101$
```

cout << cin >>

```
#include <iostream>
using namespace std; // or std::cout std::cin
```

Use the cout object to display information on the computer's screen

- **c**onsole **out**put (cmd/terminal)
- << stream insertion operator
- more than 1 item

```
cout << "You are " << age;
```

- to start a new line:
  - stream manipulator
  - escape sequence

```
cout << endl;
cout << "\n"; // "\t"
```

The cin object can be used to read data typed at the keyboard

- **c**onsole **in**put
- >> stream extraction operator
- more than 1 item (separate by space/enter)

```
cin >> firstname >> year;
```

cin: string with spaces cannot be read fully

Variable

Variables represent storage locations in the computer's memory that may hold data

```
int age = 18;
```

`int`	type of data
`age`	variable's name
`=`	assignment
`18`	value/literal/constant

→ to access the value of variable:

```
cout << " You are " << age ;
```

Named constants: *read-only* literals may be given names that symbolically represent them in a program

```
const int CURRENT_YEAR = 2023;
int age;
age = CURRENT_YEAR - year;
```

Assignment – Initilization – Scope

Assignment

```
age = 18;
```

= assignment operator
`lvalue`: left side of =: memory (age)
`rvalue`: right side of =: expression with value (18)

Initilization

```
int age = 18; // assign values when defining
```

Scope A variable's scope is the part of the program that has access to the variable

```
cout << age; // error: 'age' was not declared in this scope
{
    int age = 18;
    cout << age; // correct
}
cout << age; // error: 'age' was not declared in this scope
```

Data types

Variables are classified according to their data type, which determines the kind of information that may be stored in them

- boolean (1 byte)

```
bool check = true; // false
```

- character (1 byte)

```
char letter = 'p'; // encode by ASCII
letter = '2'; // not a number
```

- numeric
  - integer (signed / unsigned)

`short int`	2 bytes	[-32768, 32767]
`unsigned short int`	2 bytes	[0, 65535]
`int`	4 bytes	[-2.14E+9, 2.14E+9]
`long` / `long long`	8 bytes	[-9.2E+18, 9.2E+18]

  - floating point (with precision)

`float`	4 bytes	[-3.4E+38, -3.4E-38] [3.4E-38, 3.4E+38]
`double`	8 bytes	[-1.7E+308, -1.7E-308] [1.7E-308, 1.7E+308]

Programming style

Programming style refers to the way a programmer uses identifiers, spaces, tabs, blank lines, and punctuation characters to visually arrange a program's source code

```
1 #include <iostream>
2 using namespace std;
3 int main(){
4 int a, xyz;
5 cin>>a;
6 xyz = 2023 - a;
7 cout<<"You are"<<b<<endl;
8 return 0;
9 }
```

```
1 #include <iostream>
    using namespace std;
     int main(){int year
    , age; cout << "When
     were you born? ";
    cin >> year; age =
    2023 - year; cout <<
     "You are " << age
    << endl; return 0; }
```

A program should be *[Read D.S.Malik book - chap2.part Program Style]*

- easy to follow
- consistent
- make sense to other programmers

Arithmetic Operators

unary	-	negative	`age = -(-18)`
binary	+	addition	`current_year = age + born_year`
	-	substraction	`age = 2023 - born_year`
	*	multiplication	`total = (10 + bonus) * 2 * 0.9`
	/	(integer) division	`unit_price = 21 / 10` `unit_price = 21.0 / 10`
	%	modulus	`left_over = 21 % 10`
ternary	? :	condition	`age > 17 ? "adult" : "child"`

Mathematical expressions [Appendict C/Appendict A.6 (VQHoang book)]

- when two operators share an operand, the operator with the highest *precedence* works first
- if two operators sharing an operand have the same precedence, they work according to their *associativity*

```
    a = 6 - 3 * 2 + 7 % 6
<=> a = 6 - (3 * 2) + (7 % 6)
<=> a = 1
```

Overflow and Underflow

When a variable is assigned a value that is too large or too small in range for that variable's data type → overflow/underflow

```
short test_var = 32767;
test_var = test_var + 1; //overflows

test_var = -32768;
test_var = test_var - 1; //underflows
```

- compile successfully
- but incorrect result (logical bug)
- unexpected behaviors

Type casting allows you to perform manual data type conversion

```
char letter = 'A';

cout << static_cast<int> (letter)
        << " " << (int) letter; //explicit

int number = letter; //implicit
int sum = 32 + letter; //implicit

double unit_price = 21.0 / 10;
double unit_price = 21 / 10;
int unit_price = 21.0 / 10;
```

Mathematical library

Function a "black box" that takes input (arguments) and performs a specific tasks

```
pow(3, 2)
```

`pow`	function name
`3, 2`	argument (separate by ,)

The C++ runtime library provides several functions for performing complex mathematical operations

```
#include <cmath> // for sqrt and pow function

int absolute_value = abs(-18);
double square_root = sqrt(18.0);
double base10_logarithm = log10(18.0);
```

TODO: [D.S.Malik book] chap 2, programming exercises 13

TODO

- Finish chapter 2, chapter 3, Appendict C
- Read chapter 4, part 2.4 (VQHoang book)