Hochiminh City University of Technology
Computer Science and Engineering
[CO1027] - Fundamentals of C++ Programming

Basic components

Lecturer: Duc Dung Nguyen
Credits: 3

Outcomes

- Be able to write simple programs
- Be able to explain the source code using comments
- Be able to format the output to the console

“C makes it easy to shoot yourself in the foot; C++ makes it harder, but when you do, it blows away your whole leg.”

– Bjarne Stroustrup

Today's outline

- Program structure
- Variable and Data types
- Problem solving

Program structure

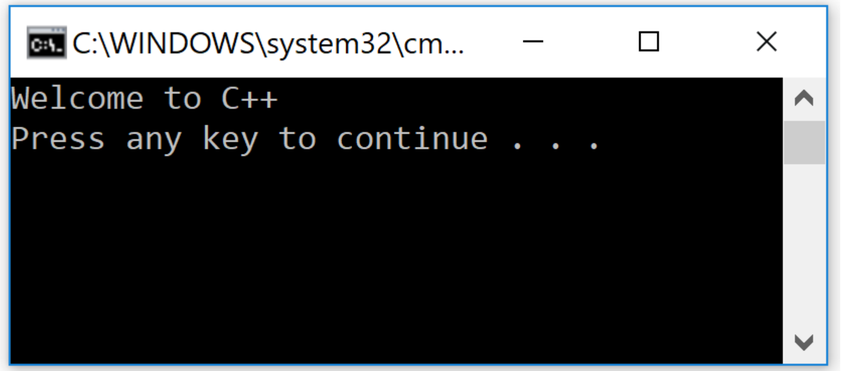
Program structure

```
#include<iostream>

int main()
{
   std::cout << "Welcome to C++!\n";
   return 0;
}
```

preprocessor directives

main() function



Program structure

- Preprocessing directive: anything begin with #
  - include: source inclusion (use libraries or additional code)
  - define, undef, etc.: constant, macros
  - pragma: specify diverse options for compiler
- main(): the entry point of our program. This is where everything start.

Program structure

- Global variables definition: these variable are visible to all classes and functions in the program
- Structure, Class or Function definition and implementation
- *Namespace*: use to group components of a module, library, or a pack of smaller libraries.

Using namespace

```
#include<iostream>
using namespace std;


int main()
{
   cout << "Welcome to C++!\n";
   return 0;
}
```

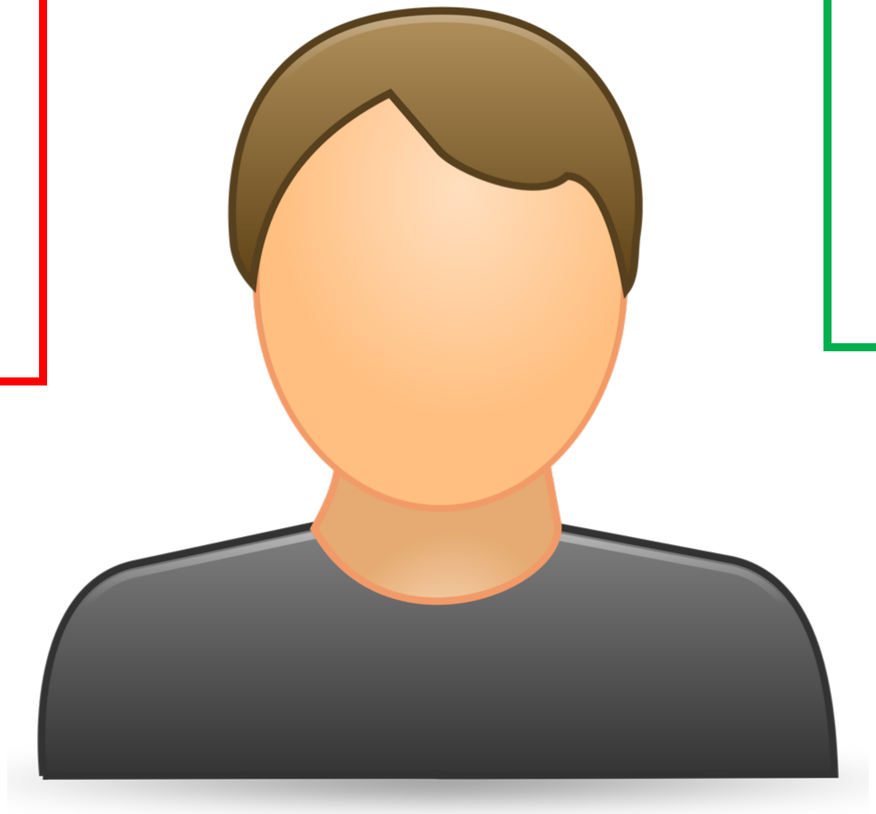
Program structure

```
#include <something>
// define data structure
// define functions
// declare global variables
// using namespace

int main()
{
   // local variables
   /* Your code should be put here.
   TO DO
   */
   return 0;
}
```

User IO

std::cout for writing to the console.



std::cin
for reading from the console

User IO

```cpp
#include <iostream>
using namespace std;

int main()
{
    int age;
    cout << "How old are you?\n";
    cin >> age;
    cout << "Your age: " << age << endl;
    return 0;
}
```

Comments

- Two types
  - Single line comments: anything after //
    - `a = 0; // set variable a to 0`
  - Block comments: anything between /* and */
    - ```
      b = 1; /* set b to 1.
      remember that the value variable b
      can be changed later */
      ```
      ```

Style guide

- There are a number of style guides available, **the best one is the one used by the people who are paying you**.
- A straightforward style guide is:
  C++ Coding Standards
- For a more detailed guideline:
  Google C++ Style Guideline

Compile the program

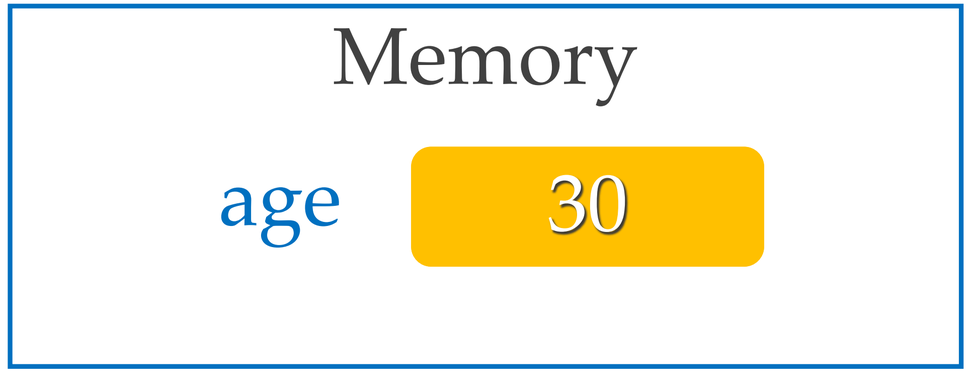
- Simplest way: using IDE
  - Visual Studio, Xcode, KDE IDE, Eclipse, etc.
- Manually: gcc, g++
  - gcc example.c
  - g++ example.cpp
- Compile and link separately
- Use other build system: e.g. CMake

Variables and Data types

Variable

```
#include <iostream>
using namespace std;

int main()
{
    int age;
    cout << "How old are you?\n";
    cin >> age;
    cout << "Your age: " << age << endl;
    return 0;
}
```



Data types

- Data types in C++ is mainly divided into two types:
  - **Primitive Data Types:** built-in or predefined data types and can be used directly by the user to declare variables. Example: int, char , float, bool etc.
  - **Abstract or user defined data type:** defined by user itself. Like, defining a class in C++ or a structure.

Primitive Built-in Types

Type	Keyword
Boolean	bool
Character	char
Integer	int
Floating point	float
Double floating point	double
Valueless	void
Wide character	wchar_t

Memory size

Type	Typical Size (in bytes)	Typical Range
char	1	-127 to 127 or 0 to 255
unsigned char	1	0 to 255
signed char	1	-127 to 127
int	4	-2147483648 to 2147483647
unsigned int	4	0 to 4294967295
signed int	4	-2147483648 to 2147483647
short int	2	-32768 to 32767
unsigned short int	2	0 to 65,535
signed short int	4	-32768 to 32767
long int	4	-2,147,483,648 to 2,147,483,647
signed long int	4	same as long int
unsigned long int	4	0 to 4,294,967,295
float	4	+/- 3.4e +/- 38 (~7 digits)
double	8	+/- 1.7e +/- 308 (~15 digits)
long double	8 or 12	+/- 1.7e +/- 308 (~15 digits)
wchar_t	2 or 4	1 wide character

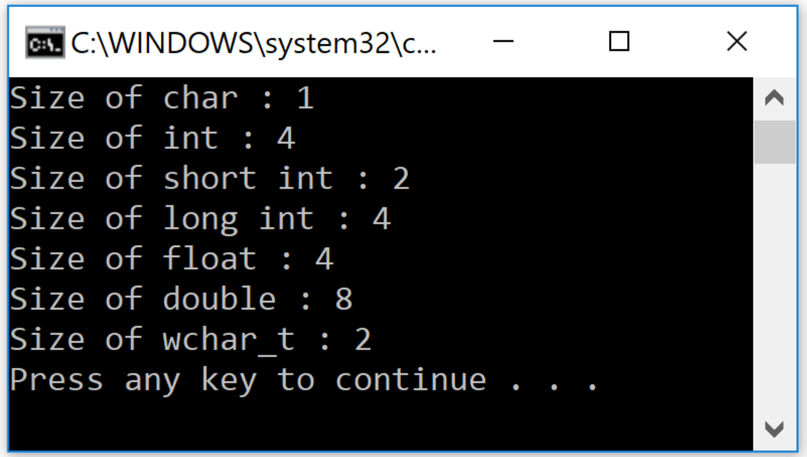
https://www.tutorialspoint.com/cplusplus/cpp_data_types.htm

Memory size

```
#include <iostream>
using namespace std;

int main() {
   cout << "Size of char : " << sizeof(char) << endl;
   cout << "Size of int : " << sizeof(int) << endl;
   cout << "Size of short int : " << sizeof(short int) << endl;
   cout << "Size of long int : " << sizeof(long int) << endl;
   cout << "Size of float : " << sizeof(float) << endl;
   cout << "Size of double : " << sizeof(double) << endl;
   cout << "Size of wchar_t : " << sizeof(wchar_t) << endl;

   return 0;
}
```



Variable declaration

- [optional] <type> <variable name>[<array declaration>][=assigned values]
  - int i, j;
  - float x = 0.5f;
  - const char* depName = “BK-CSE”;
  - volatile int noOpt = 100;
  - etc.

Variable declaration

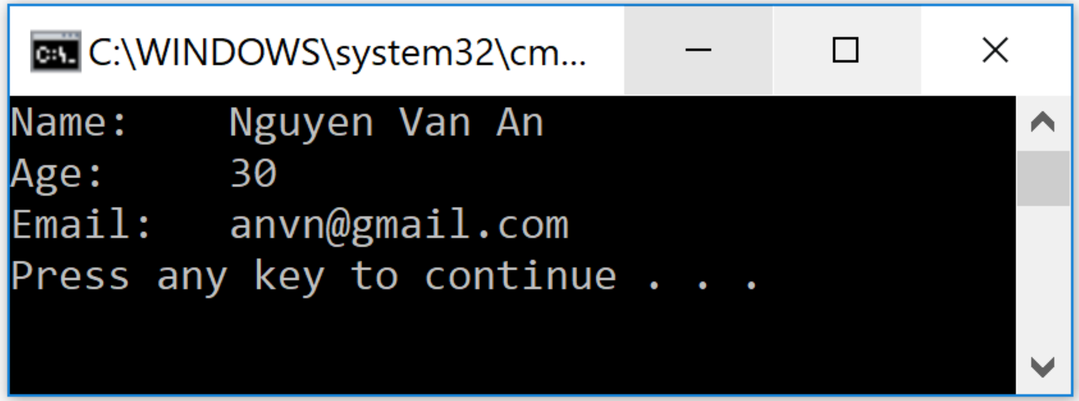
- Rules for variable name (identifiers, in general):
  - A valid identifier is a sequence of one or more letters, digits, or underscore characters (_).
  - Spaces, punctuation marks, and symbols cannot be part of an identifier.
  - Identifiers shall always begin with a letter/_.
- Case sensitive
- Reserved keywords:
  - `alignas, alignof, and, and_eq, asm, auto, bitand, bitor, bool, break, case, catch, char, char16_t, char32_t, class, compl, const, constexpr, const_cast, continue, decltype, default, delete, do, double, dynamic_cast, else, enum, explicit, export, extern, false, float, for, friend, goto, if, inline, int, long, mutable, namespace, new, noexcept, not, not_eq, nullptr, operator, or, or_eq, private, protected, public, register, reinterpret_cast, return, short, signed, sizeof, static, static_assert, static_cast, struct, switch, template, this, thread_local, throw, true, try, typedef, typeid, typename, union, unsigned, using, virtual, void, volatile, wchar_t, while, xor, xor_eq`

Output format

Output format

- Use escape sequences
  - \n for new line
  - \t for tab

```
cout << "Name:\t Nguyen Van An\n";
cout << "Age:\t 30\n";
cout << "Email:\t anvn@gmail.com\n";
```



Output format

- Use <iomanip> library (http://www.cplusplus.com/reference/iomanip/)
  - `setw(n)`: set the *field width n* to be used on output operations. By default, `setw(n)` will **justify** everything to the right.
  - Use `left, right, internal` to adjust to the side you want

```
#include <iomanip>

cout << left << setw(8) << "Name:" << "Nguyen Van An\n";
cout << left << setw(8) << "Age:" << "30\n";
cout << left << setw(8) << "Email:" << "anvn@gmail.com\n";
```

Output format

- Using cout: require “iomanip” for formatting
  - `cout.width(<output width>)`: set width of the output result, both text and number
    - `cout << setw(<output width>)`
  - `cout.precision(<number>)`: set maximum number of significant digits (set to 0 to reset this setting)
    - `cout << setprecision(<number>)`
- `cout << “*” << setw(4) << 8 << “*” << endl;`
- `cout << “*” << setprecision(4) << 12356.4 << “*” << endl;`

Output format

- Save old settings:
  - `ios::fmtflags old_settings = cout.flags();`
  - `int old_precision = cout.precision();`
- Load settings:
  - `cout.flags(new_settings);`
  - `cout.precision(new_precision);`

Output format

- Fixed format for floating point number
  - `cout.setf(ios::fixed, ios::floatfield);`
  - `cout << fixed;`
  - Reset to default: `cout.setf(0, ios::floatfield);`
- E.g.:
  - ```
    cout.setf(ios::fixed, ios::floatfield);
    cout.precision(2);
    cout << 3.14159 << “, ” << 0.8642e-3;
    ```
    ```

Problem solving

Example problem

- Write a program that solve the quadratic equation (second degree polynomial equation) of the following form:

$$ax^2 + bx + c = 0$$

- Prepare:
  - Define problem, what are criteria and constraints?
  - Gather information, explore, plan
  - Act: write the algorithm, implement code
  - Check, generalize, disseminate

Example problem

- Define problem:
  - Find solution for the quadratic equation
  - Input: constants a, b, c
  - Output: variable x
- Criteria, constraints:
  - a, b, c: integer, real, or complex numbers?
  - x: real or complex number?

Example problem

- Gather information:
  - known: constants a, b, c
  - unknown: variable x
- Explore: existed solution!
- Plan: input constants, check condition, compute solution

Example problem

- Act:
  - Write pseudocode and draw flowchart.
  - Write the program
- Check:
  - Did the program operate as designed?
  - Is the output correct w.r.t the input?
  - Did you test every cases?

Summarise

- Understand basic elements of C/C++
- Be able to read and explain the code with comments
- Know how to write clear code