Lingkungan Pemrograman C++

Pertemuan 2

Outline

- C++ Programming Environtment
  - Writing Your First Program
  - Using Arguments with Functions
  - Storing Variables in Memory
  - Assigning Values to Variables
  - Using Type Definitions
  - Expressions
  - Prefix and Postfix Operators
  - **If**-**Else** Conditional Statements
  - Compound **If** Statements

Compiler & Linker

- Programs C++ dikembangkan dari 2 perangkat yang disebut dengan *compiler* dan *linker*.
- C**ompiler** merubah program C++ menjadi bentuk yang dapat dieksekusi. Compiler dapat menerjemahkan program (*source code*) menjadi bentuk yang dapat dibaca oleh mesin (*machine code*). Compiler akan menghasilkan *object file*.
- L**inker** yang akan mengubah *object file* agar dapat dijalankan di mesin.

Lingkungan Pemrograman C++

- Ada beberapa lingkungan pemrograman yang dapat digunakan untuk membuat program dengan bahasa C++. Diantaranya GCC (the GNU Compiler Collection), Microsoft Visual C++, NetBeans dan Code::Blocks. Dalam praktikum ini digunakan GCC sebagai compilernya.
- GCC atau GNU C Compiler peralatan pemrograman yang *open source* yang mendukung pengembangan aplikasi C++. GCC sangat populer baik di mesin Linux, Windows dan Mac OS. GCC bekerja dengan a command-line, dimana kita dapat memberikan perintah agar compiler dan linker dapat menghasilkan program yang dapat dieksekusi. GCC compiler dapat melakukan kompilasi (*compile*) sekaligus mengintegrasikan (*link*) dengan library dalam satu langkah.

Langkah Membuat Program C++

- Berikut ini adalah langkah-langkah dalam membuat program C++ secara bertahap:
  - Buat file *source code* dengan editor teks.
  - Gunakan compiler untuk mengkonversi *source code* menjadi *object file*.
  - Gunakan *linker* untuk menghubungkan *object file* dan setiap *library* yang diperlukan untuk menghasilkan program *executable*.

```
g++ program.cpp -o executable.exe
```

  - Ketik nama *executable* untuk menjalankannya.

```
./executable.exe
```

Variable Types

Type	Size	Values
`unsigned short`	2 bytes	0 to 65,535
`short`	2 bytes	–32,768 to 32,767
`unsigned long`	4 bytes	0 to 4,294,967,295
`long`	4 bytes	–2,147,483,648 to 2,147,483,647
`int`	4 bytes	–2,147,483,648 to 2,147,483,647
`unsigned int`	4 bytes	0 to 4,294,967,295
`long long int`	8 bytes	-9.2 quintillion to 9.2 quintillion
`char`	1 byte	256 character values
`bool`	1 byte	true or false
`float`	4 bytes	1.2e–38 to 3.4e38
`double`	8 bytes	2.2e–308 to 1.8e308

Relational Operators

Name	Operator	Sample	Evaluates
Equals	`==`	`100 == 50;`	false
		`50 == 50;`	true
Not equal	`!=`	`100 != 50;`	true
		`50 != 50;`	false
Greater than	`>`	`100 > 50;`	true
		`50 > 50;`	false
Greater than or equals	`>=`	`100 >= 50;`	true
		`50 >= 50;`	true
Less than	`<`	`100 < 50;`	false
		`50 < 50;`	false
Less than or equals	`<=`	`100 <= 50;`	false
		`50 <= 50;`	true

Logical Operators

Operator	**Symbol**	**Example**
AND	`&&`	`grade >= 70 && grade < 80`
OR	`¦¦`	`grade > 100 ¦¦ grade < 1`
NOT	`!`	`!grade >= 70`

Operator Precedence

Level	Operators	Evaluation Order
1 (highest)	`( ) . [ ] fi ::`	Left to right
2	`* & ! ~ ++ – – + -`	Right to left
	`sizeof new delete`	Left to right
3	`.* fi *`	Left to right
4	`* /`	Left to right
5	`+ -`	Left to right
6	`<< >>`	Left to right
7	`< <= > >=`	Left to right
8	`== !=`	Left to right
9	`&`	Left to right
10	`^`	Left to right
11	`¦`	Left to right
12	`&&`	Left to right
13	`¦¦`	Left to right
14	`?:`	Right to left
15	`= *= /= += -= %=`	Right to left
	`<<= >>= &= ^= ¦=`	Right to left
16 (lowest)	`,`	Left to right

1. Writing Your First Program

Hello.cpp

```
#include <iostream>

int main()
{
        std::cout << "Hello world!\n";
        return 0;
}
```

Now that you've been introduced to how the process works, it's time to create your first C++ program and give the compiler a test drive. When you've finished, save the file as hello.cpp.

```
$ g++ Hello.cpp -o Hello.exe
$ ./hello.exe
```

2. Using Arguments with Functions

Calculator.cpp

```cpp
#include <iostream>

int add(int x, int y)
{
      // add the numbers x and y together and return the sum
      std::cout << “Running calculator ...\n”;
      return (x+y);
}

int main()
{
      /* this program calls an add() function to add two different
      sets of numbers together and display the results. The
      add() function doesn’t do anything unless it is called by
      a line in the main() function. */
      std::cout << “What is 867 + 5309?\n”;
      std::cout << “The sum is “ << add(867, 5309) << “\n\n”;
      std::cout << “What is 777 + 9311?\n”;
      std::cout << “The sum is “ << add(777, 9311) << “\n”;
      return 0;
}
```

3. Storing Variables in Memory

Sizer.cpp

```
#include <iostream>

int main()
{
        std::cout << "The size of an integer:\t\t";
        std::cout << sizeof(int) << " bytes\n";
        std::cout << "The size of a short integer:\t";
        std::cout << sizeof(short) << " bytes\n";
        std::cout << "The size of a long integer:\t";
        std::cout << sizeof(long) << " bytes\n";
        std::cout << "The size of a character:\t";
        std::cout << sizeof(char) << " bytes\n";
        std::cout << "The size of a boolean:\t\t";
        std::cout << sizeof(bool) << " bytes\n";
        std::cout << "The size of a float:\t\t";
        std::cout << sizeof(float) << " bytes\n";
        std::cout << "The size of a double float:\t";
        std::cout << sizeof(double) << " bytes\n";
        std::cout << "The size of a long long int:\t";
        std::cout << sizeof(long long int) << " bytes\n";

        return 0;
}
```

4. Assigning Values to Variables

Rectangle.cpp

```cpp
#include <iostream>

int main()
{
    // set up width and length
    unsigned short width = 5, length;
    length = 10;

    // create an unsigned short initialized with the
    // result of multiplying width by length
    unsigned short area = width * length;

    std::cout << “Width: “ << width << “\n”;
    std::cout << “Length: “ << length << “\n”;
    std::cout << “Area: “ << area << “\n”;
    return 0;
}
```

5. Using Type Definitions

NewRectangle.cpp

```cpp
#include <iostream>

int main()
{
      // create a type definition
      typedef unsigned short USHORT;

      // set up width and length
      USHORT width = 5;
      USHORT length = 10;

      // create an unsigned short initialized with the
      // result of multiplying width by length
      USHORT area = width * length;

      std::cout << "Width: " << width << "\n";
      std::cout << "Length: " << length << "\n";
      std::cout << "Area: " << area << "\n";
      return 0;
}
```

6. Expressions

Expression.cpp

```
#include <iostream>
int main()
{
    int x = 0, y = 72, z = 0;
    std::cout << “Before\n\nx: “ << x << “ y: “ << y;
    std::cout << “ z: “ << z << “\n\n”;
    z = x = y + 13;
    std::cout << “After\n\nx: “ << x << “ y: “ << y;
    std::cout << “ z: “ << z << “\n”;
    return 0;
}
```

7. Prefix and Postfix Operators

Years.cpp

```cpp
#include <iostream>

int main()
{
        int year = 2010;
        std::cout << "The year " << ++year << " passes.\n";
        std::cout << "The year " << ++year << " passes.\n";
        std::cout << "The year " << ++year << " passes.\n";

        std::cout << "\nIt is now " << year << ".";
        std::cout << " Have the Seattle Mariners won the World Series yet?\n";

        std::cout << "\nThe year " << year++ << " passes.\n";
        std::cout << "The year " << year++ << " passes.\n";
        std::cout << "The year " << year++ << " passes.\n";

        std::cout << "\nSurely the Mariners have won the Series by now.\n";
        return 0;
}
```

8. **If-Else** Conditional Statements

Grader.cpp

```cpp
#include <iostream>

int main()
{
        int grade;
        std::cout << "Enter a grade (1-100): ";
        std::cin >> grade;

        if (grade >= 70)
        std::cout << "\nPass\n";
        else
        std::cout << "\nFail\n";

        return 0;
}
```

9. Compound **If** Statements

NewGrader.cpp

```cpp
#include <iostream>

int main()
{
        int grade;
        std::cout << "Enter a grade (1-100): ";
        std::cin >> grade;

        if (grade >= 70)
        {
                if (grade >= 90)
                {
                        std::cout << "\nPass with an A grade\n";
                        return 0;
                }
                if (grade >= 80)
                {
                std::cout << "\nPass with a B grade\n";
                return 0;
                }
                std::cout << "\nPass with a C grade\n";
        }
        else
                std::cout << "\nFail\n";

        return 0;
}
```

Tugas (1)

- Buatlah sebuah program yang menggunakan constant untuk touchdown (6 point), field goal (3 point), extra point (1 point), dan safety (2 point). Kemuaian jumlahkan semuanya dengan urutan yang sama seperti saat mencetak skor oleh tim yang ada pada SuperBowl. Tampilkan skor akhirnya.
- Tambahkan pada program Rectangle.cpp agar dapat menentukan area luasan 3 dimensi dari segitiga yang memiliki lebar, panjang dan tinggi. Untuk menentukan luasan gunakan operator perkalian * untuk mengalikan ketiga nilai tersebut.

Tugas (2)

- Buatlah versi baru dari program NewGrader yang tidak termasuk "return statement", kecuali untuk yang final. Jalankan dengan beberapa kali percobaan nilai, sampai Anda menemukan bug dari program tersebut. Jelaskan apa yang sebenarnya terjadi?
- Buatlah sebuah program yang meminta dari user dari 1 sampai 100, tanyakan apakah passing grade berada di skala yang sama, dan laporkan bahwa user telah melewati passing grade tersebut.