LAB2 : Operation

***Task1: sizeof()***

1. Please execute the program 4-1-1.cpp(p76) and 3-3.cpp(p64).

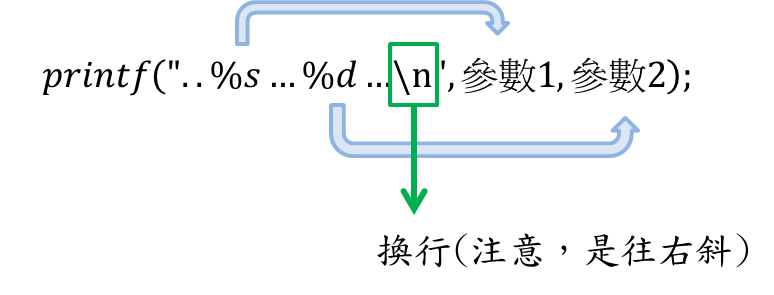
Record the size of variables on p77.

|  |  |  |
| --- | --- | --- |
| Type name | Size | Range |
| bool | 1 byte (8 bits) | Just true (1) or false (0). |
| char | 1 byte (8 bits) | signed: to  unsigned: 0 to  ASCII  (reference : <https://zh.wikipedia.org/zh-tw/ASCII> ) |
| int | 4 bytes (32 bits) | signed: to  unsigned: 0 to |
| long | 8 bytes (64 bits) | to |
| float | 4 bytes (32 bits) | +/- 3.4e +/- 38 ( 7 digits) |
| double | 8 bytes (64 bits) | +/- 1.7e +/- 308 ( 15 digits) |

1. C Language：

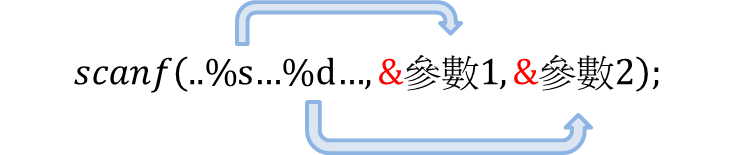
|  |  |  |  |
| --- | --- | --- | --- |
| Symbol | Description | Symbol | Description |
| %d | Interger | \n | Next line |
| %f | Floating number | \t | TAB |
| %c | Char |  |  |
| %s | String |  |  |

1. Output data :



|  |
| --- |
| //3-3-2.cpp (C )  #include <iostream>  using namespace std;  int main ()  {      int student\_id = 1124201;      char class\_name = 'B';      printf("My student ID is %d.\n I am one of the students in Daan Elect. class %c.\n", student\_id, class\_name);      system("Pause");  } |

1. Input data :



|  |
| --- |
| //3-3-2.cpp (C )  #include <iostream>  using namespace std;  int main ()  {      int num;      char school\_name[100],x;      printf("Input school name , students number and class:\n");      scanf("%s %d %c", &school\_name, &num, &x);      printf("There are %d students in %s class %c\n", num, school\_name, x);      system("Pause");  } |

Please complete the program 3-3.cpp(p64) in C Language.

|  |
| --- |
| //3-3.cpp (C )  #include <iostream>  #include <iomanip>  #include <cmath>  using namespace std;  int main ()  {      int v1=pow(2,31)-1;      char v3='A';      bool v4=true;      float v5=exp(1);      double v6=exp(1);      int num1;      float num2;      char x[20];      printf("v1(int) = %d\n",v1);      printf("v3(char) = %c\n",v3);      printf("v4(bool) = %d\n",v4);      printf("v5(float) = %f\n",v5);      printf("v6(double) = %f\n",v6);      printf("There are %d students in %s class %c\n", 40, "Daan elect.", 'B');      system("Pause");  } |

***Task2: Constants***

1. Constants : A constant is an expressions with a ﬁxed value. Kinds of constants:
2. Literals: used to express particular values within the source code; can be integers (ex: 1, -17), ﬂoating points (ex: 3.1415926535897, 1., 6.096, 3), characters (ex: ‘a’, ‘ ’, \n’), strings (ex: “6.096”, “a”), booleans (ex: true, false)
3. Deﬁned constants (#define): user-deﬁned named constants that do not require memory-consuming variables. When the preprocessor sees the #define, it will replace every subsequent occurrance of the identiﬁer in the source code.

|  |
| --- |
| //#define practice  #include <iostream>  using namespace std;  # define PI 3.14159  # define NEWLINE '\n'  int main ()  {      double r = 5.0;      double circle ;        circle = 2 \* PI \* r; // circle = 2 \* 3.14159 \* r;      cout << circle << NEWLINE ; // cout << circle << '\n';        return 0;  } |

1. Declared constants (const): user deﬁned constants with const preﬁx with a speciﬁc type that are declared the same way as variables. The value of a const variable cannot be modiﬁed. Please execute the program 3-4-1.cpp(p67).

🡺Please execute the 3-4-1.cpp(p67) in #define.

***Task3: Type Conversions***

Used for changing between data types. Type conversions are implicit when changing from smaller data type to a bigger data type or data type of same size (e.g. float to double or int to float). Type conversions usually must be explicitly stated when changing from bigger datatype to smaller datatype or when there could be a loss of accuracy (e.g. int to short or float to int), but implicitly converting from a double to an int will not generate a compiler error (the compiler will give a warning, though).

1. Please execute the 4-2-1.cpp(p79)

|  |
| --- |
| //4-2-1.cpp  #include <iostream>  using namespace std;  int main ()  {      int v1 = 2;      float v2;      double v3;        cout<<"Input a floating point number = ";      cin >> v2;      v3 = v1 + v2;      cout<<"input num + 2, Sum = " << v3 << endl;      v1=(int) v2;      cout<<"v2 = " << v2 << " ,(int) v2 = " << v1 <<endl;      v1=(int) (v2+0.5);      cout<<"v2 = "<<v2<<" ,(int) (v2+0.5) = "<< v1 <<endl;      system("Pause");  } |

1. C Language：

|  |
| --- |
| //type conversions in C  #include <iostream>  using namespace std;  int main ()  {      int num1 = 2147483647;      float num2 = 3.6;        printf("1. (3+2.5) = %f\n", (3+2.5));      printf("2. (3/2) = %d\n", (3/2));      printf("3. (3.0/10) = %f\n", (3.0/10));      printf("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");        printf("num1 = 2147483647, (float) num2 = 3.6 \n");      printf("Type Conversions: \n");      printf("1. num1 + 1 = %d + 1 = %d\n", num1, (num1 + 1));      printf("2. (float)num1 + 1 = %d + 1 = %f\n", num1, ((float)num1 + 1));      printf("3. %f \* 10 = ((int)num2 \* 10) = %d\n", num2, ((int)num2 \* 10));      printf("4. 7 / 2 = %d\n", (7 / 2 ) );      printf("5. (float) (7 / 2 ) = %f\n", (float) (7 / 2 ) );      system("Pause");  } |

Run :

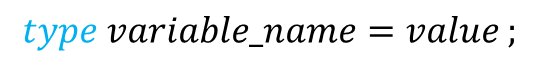
|  |
| --- |
| Type Conversions:  1. num1 + 1 = 2147483647 + 1 = -2147483648  2. (float)num1 + 1 = 2147483647 + 1 = 2147483648.000000  3. 3.600000 \* 10 = ((int)num2 \* 10) = 30  4. 7 / 2 = 3  5. (float) (7 / 2 ) = 3.000000 |

🡺Can you explain the type conversions 1~5 ?

***Task4: Operators***

1. Assignment operators :

include +=, -=, \*=, /=, %=, etc.; a += 5 is equivalent to a = a + 5, etc.



1. Arithmetic operator and combination-assignment operator:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| op. | description | example | combination | example |
| + | Addition | 6 + 4 = 10 | += | x += y; // x = x + y; |
| - | Subtraction | 6 - 4 = 2 | -= | x -= y; // x = x - y; |
| \* | Multiplication | 6 \* 4 = 24 | \*= | x \*= y; // x = x \* y; |
| / | Division | 6 / 4 = 1 | /= | x /= y; // x = x / y; |
| % | Modulo  (only for integer) | 6 % 4 = 2 | %= | x %= y; // x = x % y; |

Run program basic\_op.cpp to testify the example.

|  |
| --- |
| //basic op.  #include <iostream>  using namespace std;  int main()  {      int x = 6;      x += 3;      cout<<"6 + 4 = " << (6 + 4) << endl;      cout<<"6 - 4 = " << (6 - 4) << endl;      cout<<"6 \* 4 = " << (6 \* 4) << endl;      cout<<"6 / 4 = " << (6 / 4) << endl;      cout<<"6 % 4 = " << (6 % 4) << endl;      cout<<"float x = 6, x += 3; --> x = " << x << endl;      system("Pause");  } |

1. Increment, Decrement operators (++, --) :
2. ++a will increment a and then return the value. (so it will return one greater than the original value)
3. a++ will return the current value and then increment.
4. --a will decrement a and then return the value. (so it will return one less than the original value)
5. a-- will return the current value and then decrement.

|  |
| --- |
| //prefix and postfix increment op.  #include <iostream>  using namespace std;  int main()  {      int x = 0, y;      y = (++x) + 5;      cout<<"1.Prefix : y = (++x) + 5;\n" << "  x = " << x << ", y = " << y << endl;        x = 0;      y = (x++) + 5;      cout<<"2.Postfix : y = (x++) + 5;\n" << "  x = " << x << ", y = " << y << endl;      system("Pause");  } |

🡺Can you explain the difference?

1. Operator precedence :

|  |  |  |
| --- | --- | --- |
| 1 | () [] -> . :: | Grouping, scope, array/member access |
| 2 | ! ˜ \* & sizeof (type cast) ++ – | unary operations, sizeof and typecasts |
| 3 | \* / % | Multiplication, division, modulo |
| 4 | + - | Addition and subtraction |
| 5 | << >> | Bitwise left and right shift |
| 6 | <= >= | Comparisons: less than, etc. |
| 7 | == != | Comparisons: equal and not equal |
| 8 | & | Bitwise AND |
| 9 | ˆ | Bitwise exclusive OR |
| 10 | | | Bitwise inclusive (normal) OR |
| 11 | && | Logical AND |
| 12 | || | Logical OR |
| 13 | ?: | Conditional expression (ternary operator) |
| 14 | = += -= \*= /= %=, etc. | Assignment operators |

|  |
| --- |
| //logical op.  #include <iostream>  using namespace std;  int main(){      printf("1. 9 ^ 7 = (1001) ^ (0111) = (1110) = %d\n", 9^7);      cout << "2. !(1!=3)||1==3  : " << ( !(1!=3)||1==3 ) << endl;      cout << "3. 1!=3&&!!(1==3) : " << ( 1!=3&&!!(1==3) ) << endl;      cout << "4. !(1<3)||1>=3   : " << ( !(1<3)||1>=3 ) << endl;      cout << "5. 1<3&&!(1>=3)   : " << ( 1<3&&!(1>=3) ) << endl;      system("Pause");  } |

🡺Can you explain the outcome?

1. Ternary operator (?:) : An operator that takes three arguments and deﬁnes a conditional statement.

|  |
| --- |
| if (a > b)     result = x;  else     result = y; |

is equivalent to

|  |
| --- |
| result = a > b ? x : y; |