Week 2

Muhao Chen

Email: muhaochen@ucla.edu

Outline

- Variables and operators
- Undefined behaviors
- If-else (conditional statements)
- Loops
- Characters
- String

Variables and operators

Variables and Operators

- A typical variable declaration:

  int cs = 400;

- Type:
  - int, double, char, bool, etc..
- E.g.
  - int a = 1;   //a is 1
  - int a = 2;   //a is 2
  - double a = 1;   //a is 1.0
  - double a = 1.1;   //a is 1.1
  - **int a = 2.0;**   **//a == ?**
  - //a is 2
  - **int a = 2.8;**   **//a == ?**
  - //a is 2 (round down)

Variables and Operators

- Identifier: Name of variable
  - Starting with a letter or an underscore
  - The rest part must be letters, digits or underscores
  - Case sensitive
  - Reserved word can not be used.

• **int a;**	• **int 2a;**	**X**
• **int _a;**	• **int while;**	**X**
• **int a_;**	• **int a.b;**	**X**
• **int a2;**		

- **int apple;**
- **int Apple;**
- **Int APPLE;**
- **Int aPPlE;**

Identifiers of four different variables (<-)

Variables and Operators

- Define variables
  - Some people use:

```
double cpt_tax(int price)
{
    double tax_rate = 0.0975;
    double tax;
    tax = price * tax_rate;
    return tax;
}
```

- Some people use:

```
double cpt_tax(int a)
{
    double b = 0.0975;
    double c;
    c = a * b;
    return c;
}
```

Which one is better?

Names should be chosen carefully, so that program is readable.

Variables and Operators

```
int ______(int _)
{
  double __ = 0.1;
  double _____;
  _____ = _ * __;
  return _____;
}
```

Variables and Operators

- What does the assignment operator (i.e. =) do?
  - Assign the value of the right-hand side expression to the left-hand side variable
  - The right-most "=" has the highest priority.

```
int x;
int y;
x = 5;
y = 5;
```

```
int x;
int y;
x = (y = 5);
```

```
int x;
int y;
x = y = 5;
```

```
int x = 5;
int y = 5;
```

```
int x=5, y=5;
```

```
int a = 2;       //assignment works only where it is
int b = a + 2;    //b is now 4
a = 3;            //b will not change, b is still 4
```

Variables and Operators

- Arithmetic operators (+, -, *, /, %)
  - a = 11 * 3;   //a == 33
  - b = 11(2 + 3);   // error

Variables and Operators

- Compound assignment (+=, -=, *=, /=, %=)
  - a -= 5;   What does this mean?
  - a = a - 5

- Examples

```
count += 2;
total -= discount;
bonus *= 2;
time /= rushFactor;
amount *= c1 + c2;
```

❑ which are equivalent to:

```
count = count + 2;
total = total – discount;
bonus = bonus * 2;
time = time / rushFactor;
amount = amount * (c1 + c2);
```

Variables and Operators

- Incremental Operators (++, --)
  - int i=1;
  - i++;

What does i++ mean?
What's the equivalent arithmetic expression?

```
i=i+1;
i+=1;
```

What about ++i?

Variables and Operators

- ++i first increases the value of i, and then returns the increased value.

```
i = 1;
j = ++i;

// i is 2, j is 2
```

```
i+=1;
j=i;
```

- i++ returns the (initial) value first, then do increment.

```
i = 1;
j = i++;

// i is 2, j is 1
```

```
j=i;
i+=1;
```

Variables and Operators

- int a=5, b, c;
- b = ++a; //a is 6 here. b is also 6.
- c = b++; //c is 6 here. Then b becomes 7.

- //what are the values of a, b, c?

a is 6, b is 7, c is 6.

Undefined behaviors

Undefined behavior

- What is undefined behavior?
  - The undefined behavior is the result of **executing computer code** that does not have a **prescribed behavior** by the language specification the code adheres to.
  - If any step in a program’s execution has undefined behavior, then **the entire execution is meaningless**

Undefined behavior

- E.g. Uninitialized variables

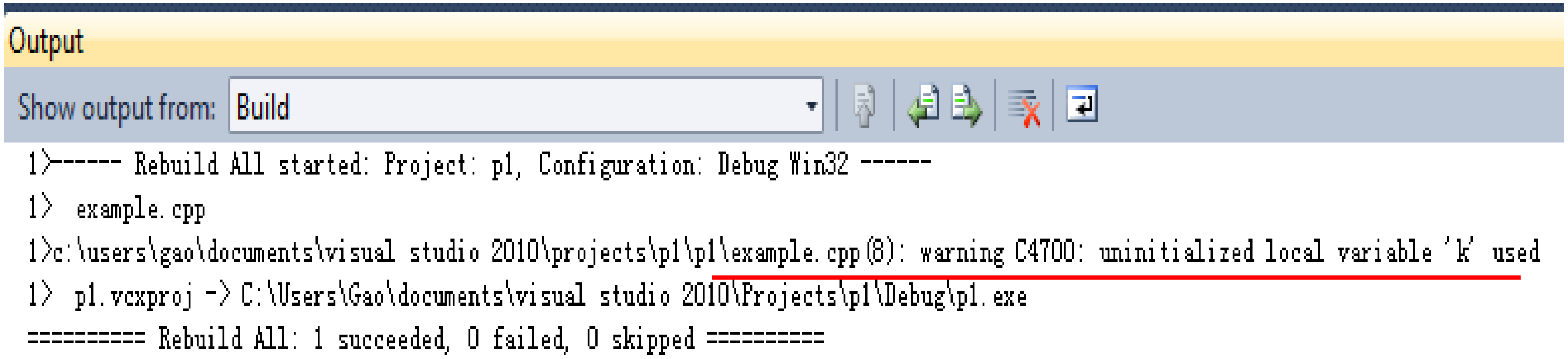
  what will happened?
  - might cause runtime error (Visual C++ debug mode)
  - or might have different values each time you run the program
  - NEVER assume that an uninitialized (int, double) variable will be 0

```
double k;
double e = 2 * k;
cout << e;
```

Undefined behavior

- When you try to run it in the debug mode.

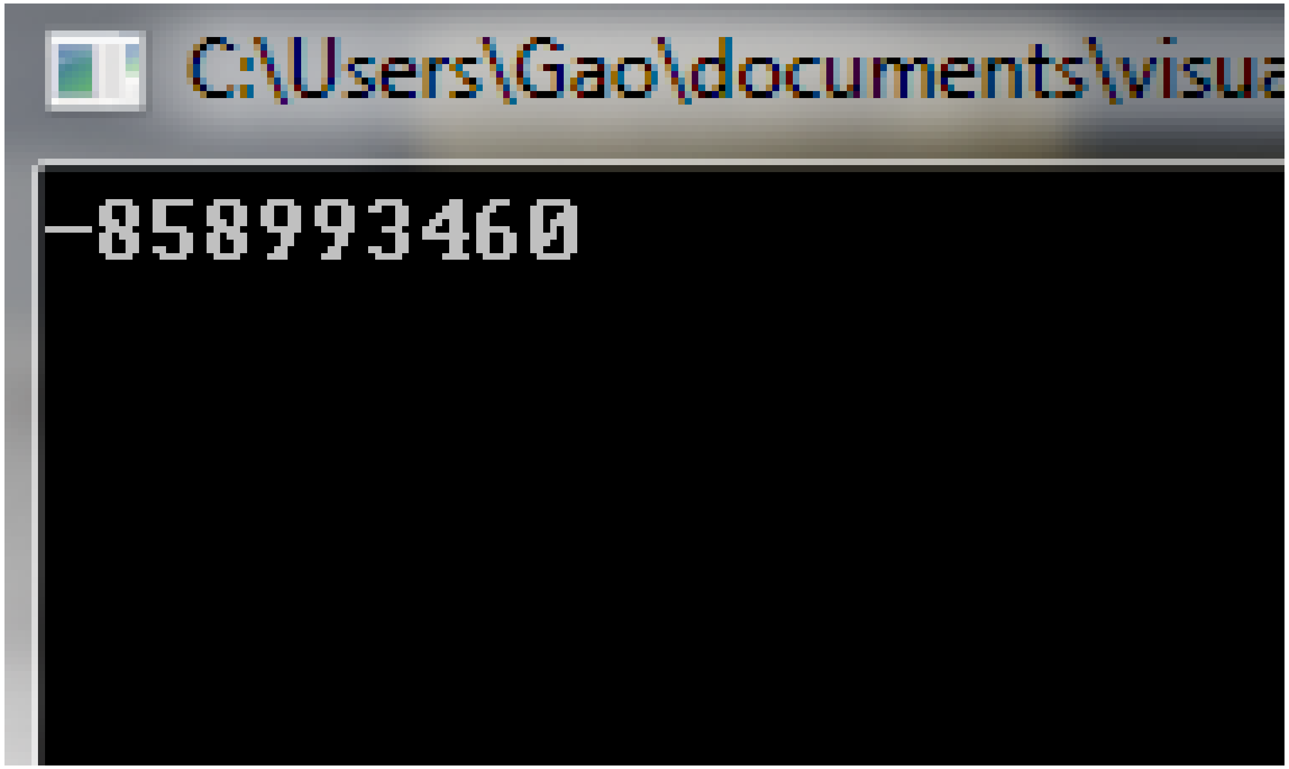
```
double k;
double e = 2 * k;
cout << e;
```



```
1>------ Rebuild All started: Project: p1, Configuration: Debug Win32 ------
1>  example.cpp
1>c:\users\gao\documents\visual studio 2010\projects\p1\p1\example.cpp(8): warning C4700: uninitialized local variable 'k' used
1>  p1.vcxproj -> C:\Users\Gao\documents\visual studio 2010\Projects\p1\Debug\p1.exe
========== Rebuild All: 1 succeeded, 0 failed, 0 skipped ==========
```

Undefined behavior

- It’s also possible that you will see this (in release mode).



```
C:\Users\Gao\documents\visua
```

```
-858993460
```

If-else (conditional statements)

If-else

- When our programs need to deal with different choices under different conditions.
  - e.g. If it is sunny, I go swimming, otherwise(else) I stay at home.
  -
    ```
    if(it is sunny tomorrow)
                I go swimming;
    else
                I stay at home;
    ```

If-else

Format:

- If (boolean expression) //bool in brackets

```
        statement;
  else
        statement;
```

- Note that the braces { }, are required when you have multiple statements

- Need brace {} for a block of multiple statements:

```
If (Boolean expression) {
    statement1;
    statement2;
    …
}
else {
    …
}
```

else is optional. If you don’t want to do anything inside “**else**”, just omit it.

If-else

Put another if-statement inside a if-statement

What's the output?

- 6

```
int a = 4, b = 4;
if (a == 4)
  if (a == b)
    a++;

if(a!=b)
  a++;
cout << a << endl;
```

If-else

- Caution
  - 1. Removing {} causes ‘if’ to affect **only one statement**.

```
int a = 8;
if (a == 4)
   a++;
a /= 4;
// a is 2
```

```
int a = 8;
if (a == 4) {
   a++;
   a /= 4;
} // a is 8
```

  - 2. Variables defined inside a branch if-else scope cannot be seen from the outside of the scope.

If-else

What's the result?

```
int main()
{
  int a = 4;
  if (a == 4)
    int b = 5;
  cout << b << endl;
}
```

- compile error
  error: 'b' was not declared in this scope
- b is declared in the if **scope**, it cannot be seen/used **outside the scope**.

Loop

While loop

When a procedure needs to be processed repeatedly.

```
while (boolean expression) {
    statements;
}
```

Loop condition

Loop body

While loop

- //compute n!=n*(n-1)*…*1

```
unsigned int n, i=1, result=1;
cin>>n;
while (i <=n) {
    result *= i++;
}
cout<<result;
```

do-while loop

- Another form:

```
do {
    statements;
} while (boolean expression);
```

- Difference?
- while : check the boolean expression before executing the loop body.
- Do .. while: execute the loop body once before checking the boolean expression.

while... and do ... while

```
int main()
{
  int n=3, i=4, result=1;
  while (i <= n) {
      result *= i;
      i++;
  }
  cout<<result;
}
```

```
int main()
{
  int n=3, i=4, result=1;
  do {
      result *= i;
      i++;
  } while (i <=n);
  cout<<result;
}
```

- n=3, i=4
- **i <=n is false. Not entering loop**
- result is 1

Check the loop condition before exec loop body

- n=3, i=4
- execute loop body first. result is 1*4=4. i++;   // i becomes 5
- **i <=n is false. End loop**
- result is 4

Exec loop body before checking the loop condition

While loop

- **Watch out for infinite loop.**

```
int n=3, i=1, result=1;
  while (i <= n) {
      result *= i;
      //i++;
}
```

- **Make sure that the loop must reach some condition to jump out**

While loop

- A programmer (David) went to the grocery store.
- Before he left, his wife said, “while you see watermelons, take one.”
- Then David never came back.

for loop

```
for ( init; condition; increment )
{
   statement(s);
}
```

1. The **init** step is executed first, and only once. This step allows you to declare and initialize any loop control variables.
2. Next, the **condition** is evaluated. This allows you to decide when to terminate the loop.
3. After the body of the for loop executes, the flow of control jumps back up to the **increment** statement. Then go back to **2**.

for loop

- n!:

```
int main()
{
  int n, result=1;
  cin >> n;
  if (n <= 0) cout << 0;
  else {
      for (int i=2; i<=n; ++i) {
        result *= i;
      }
  cout << result;
  }
}
```

for loop

- These are equivalent:

```
for (int i=2; i<=n; ++i) {
  result *= i;
}
```

```
int i=2;
for (; i<=n; ++i) {
  result *= i;
}
```

```
int i=2;
for (; i<=n;) {
  result *= i;
  ++i;
}
```

```
int i=2;
for (;;) {
  if (i>n)
     break;
  result *= i;
  ++i;
}
```

for loop

- Equivalence of for loop and while loop

```
for ( init; condition; increment )
{
   statement(s);
}
```

```
init;
while (condition)
{
   statement(s);
   increment
}
```

```
for (int i=2; i<=n; ++i)
{
   result *= i
}
```

```
int i=2
while (i<=n)
{
   result *= i;
   ++i;
}
```

for loop

- Question: can we find a equivalent do-while loop for a for-l}oop?

```
for ( init; condition; increment )
{
   statement(s);
}
```

```
init;
do {
   if (!condition)
      break;
   statement(s);
   increment;
}
while (True);
```

```
init;
if (condition) {
do {
statement(s);
   increment;
}
while (condition);
}
```

Char and String

Char in C++

- Character type **`char`** is encoded using an integer representation of **1 byte** (i.e. ASCII)
- Range (0~255)
- ASCII is the encoding schema
  - Examples
    - `' '` is encoded as 32 `'+'` is encoded as 43
    - `'A'` is encoded as 65 `'Z'` is encoded as 90
    - `'a'` is encoded as 97 `'z'` is encoded as 122

Char in C++

- Arithmetic and relational operations are defined for characters types
  - **'a' < 'b'** is true
  - **'4' > '3'** is true
  - **'6' <= '2' is false**
  - **‘F’ – 5 is ‘A’**
  - **‘x' + (‘A’ – ‘a’) is ‘X’**
  - **‘Y’ – (‘Z’ – ‘z’) is ‘y’**
  - **‘a’ – 32 is ‘A’**

Lower case letters is actually greater than its upper cases (-32)

Char in C++

- Explicit (literal) characters within single quotes
  - **`'a','D','*'`**
- Special characters - delineated by a backslash \
  - Two character sequences (escape codes)
  - Some important special escape codes
    - **`\t`** denotes a tab
    - **`\n`** denotes end-of-line
    - **`\\`** denotes a backslash
    - **`\'`** denotes a single quote
    - **`\"`** denotes a double quote
    - **`\0`** 0, end of string (NULL)

Char in C++

- #include<cctype> provides several useful functions for char, e.g.:
  - isdigit(char c): Is c a digit?
  - islower(char c): Is c lower case?
  - isupper(char c): Is c upper case?
  - isalpha(char c): Is c alphabetic?
    - Yes->return true, No->return false
  - tolower(char c): Convert c to lower case
  - toupper(char c): Convert c to upper case

Char in C++

- Example

```
// This program demonstrates some of the character testing
// functions.
#include <iostream.h>
#include <ctype.h>

void main(void)
{
 char input;
 cout << "Enter any character: ";
 cin >> input;
 cout << "The character you entered is: " << input << endl;
 cout << "Its ASCII code is: " << int(input) << endl;
```

String in C++

- String is a class in C++;
  - Class:
    - We will learn Class in detail in later classes.
    - Similar to a data type, but more powerful than a data type, e.g. it can define its own functions and attributes.
    - A string stores a sequence of characters stored in consecutive memory spaces
    - **A string is terminated by a null('\0') character.**
  - To use string, we need to add
    - #include<string>

String in C++

- Size() and Random accessing characters of a string
- For example: string s = “ab cd”;
  - s consists of 5 characters: ‘a’, ‘b’, ‘ ’, ‘c’, ‘d’;
  - We can use s.size() to get the number of characters in s, i.e. 5. (‘\0’ does not count for the size() of a string)
  - We can use s[i] to access the **(i+1)-**th character in s, e.g. s[1] = ‘b’. (i = 0 .. s.size()-1)
    - Type of s[i] is **char**
  - **Using s[i] such that i is greater than s.size()-1 is an undefined behavior**

Input a string to cause an undefined behavior

```
cin >> s;
…
for (int k=0; k<s.size(); k++) {
    if(s[k] == ‘H’) {
        if(s[k+1] == ‘E’)
            countHE++;
    }
}

//SHELLFISH
```

Thank you!

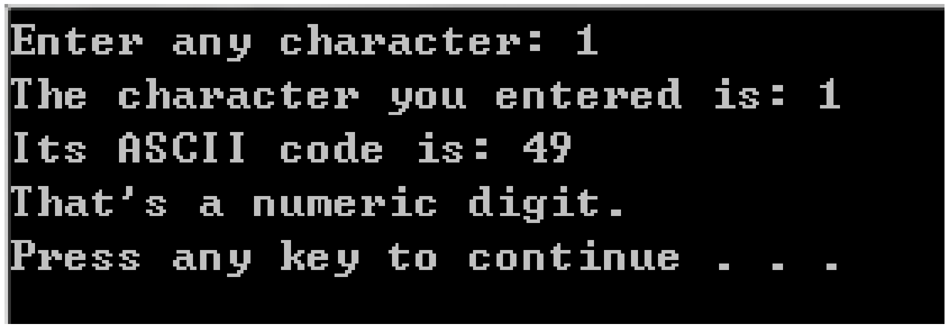
Char in C++

- Example (cnt.)

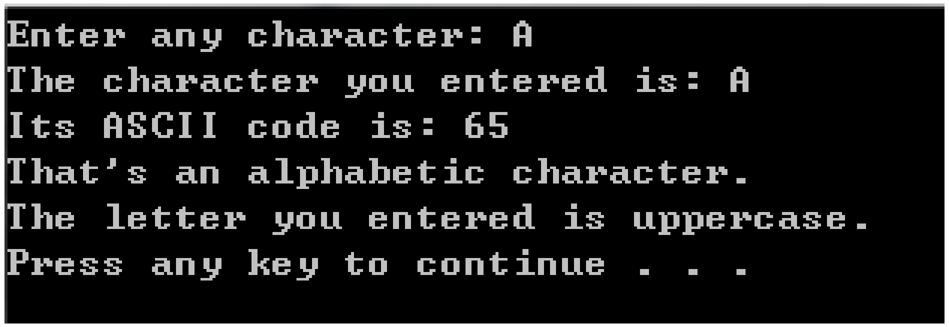
```
  if (isalpha(input))
      cout << "That's an alphabetic character.\n";
  if (isdigit(input))
      cout << "That's a numeric digit.\n";
  if (islower(input))
      cout << "The letter you entered is lowercase.\n";
  if (isupper(input))
      cout << "The letter you entered is uppercase.\n";
  if (isspace(input))
      cout << "That's a whitespace character.\n";
}
```

Char in C++

- Example (cnt.)
  - Input: 1
  - Input: a



```
Enter any character: 1
The character you entered is: 1
Its ASCII code is: 49
That's a numeric digit.
Press any key to continue . . .
```



```
Enter any character: A
The character you entered is: A
Its ASCII code is: 65
That's an alphabetic character.
The letter you entered is uppercase.
Press any key to continue . . .
```