

CS231 1 Final Revision

Part I: Pre-midterm

Dr. Henry Xu

The final exam

- 60% of your grade
- 2-hour, closed-book
- Covers everything we taught
- 8 questions
- Find errors in a program, write code, etc.

Before midterm:

- Basic syntax
- Conditional statements
- Looping statements
- In revision, we will mix different contents if needed

Basic syntax

Variable scope

- Global variables are NOT recommended
- Scope in user-defined functions

```
int summ(int x, int y)
{
    return x+y;
}
```

```
int main(){
    int x, y;
    cin >> x >> y;
    cout << summ(x,y);
    return 0;
}
```



int and char

- Escape sequences

'\n', '\t', '\#', '\0', '\'', '\"'

- Characters are almost the same as integers

```
char c = 'd';  
c++;  
cout << c;
```

Output is 'e'

```
char c = 'd';  
cout << (char)c+1;
```

```
char c = 'd';  
cout << c+1;
```

Output is 101

Strings (cstring)

- Strings are a special kind of arrays (will be covered in Part 2)

```
char name[] = "Henry Xu";
```

- Size is optional; string identifier is a constant pointer; etc.

Conversion between types

```
double z;  
z = 1/3;  
cout << z;
```

Output is 0

```
double z;  
z = 1.0/3;  
cout << z;
```

Output is 0.333333

```
char c = 'd';  
cout << (char)c+1;
```

Output is 'e'

Operators

- Increment and decrement operators

```
int x = 0;  
x++;  
--x;
```

- Efficient assignment operators

```
int x = 4;  
x += 1;  
x %= 2;  
cout << x;
```

Output is 1

Operators

- Logical operators (different from mathematics; commonly used in loops)

```
char x;  
cin >> x;  
if ( 'a' <= x <= 'z' )  
cout << "lowercase" << endl;
```



```
char x;  
cin >> x;  
if ( 'a' <= x && x <= 'z' )  
cout << "lowercase" << endl;
```



Operators

- Equality operator (different from the assignment operator!)

```
int x=0;  
if (x == 0)  
    cout << "false" << endl;
```

Output is false

```
int x=0;  
if (x = 0)  
    cout << "false" << endl;
```

Output is

cout formatting

- Remember to add the following:

```
#include<iomanip>
```

- Syntax:

```
double x=0.1234567;  
cout << fixed << setprecision(2) << x;
```

Output is 0.12

Conditional statements

if... else...

- The if statement can only have one statement in its body
- So it's strongly recommended to always use a compound statement

```
if (mark >= 90) {  
    cout << "Excellent!\n";  
}
```

dangling else

- The `else` part always matches the NEAREST `if`

```
if (a==1) {  
    if (b==2) {  
        cout << "***\n";  
    }  
    else {  
        cout << "###\n";  
    }  
}
```

Short-circuit evaluation

- Applies to logical AND and OR operators
- The left part is always evaluated. The right part may or may not be evaluated.
- The key is to remember the truth table for the two operators

Conditional operator

- Usually used as a concise way for expressing simple conditional statements.
- The part **before** ":" applies when the condition is **true**

```
int x, y;  
cin >> x >> y;  
int min_x = (x>y) ? y : x;  
cout << min_x;
```

Loops

while

- Basic syntax; always use the compound statement
- `do... while`: the loop body will be run for at least once

for

- Basic syntax; number of iterations

```
int sum = 0;
for (int i=0; i<10; i++) {
    sum += i;
}
```

- Always a good practice to initialise a variable before use
- Nested for loops, remember to use **different** index variables

break, continue

- break causes the control flow to exit from the **innermost** loop or switch statement
- continue causes the control flow to directly jump to the end of the **current** iteration, i.e. the start of the **next** iteration
- break, control exists from the loop; continue, control is still inside the loop, but just skip the rest of the current iteration