From C to C++

Sujoy Sinha Roy
School of Computer Science
University of Birmingham

Reference variables

- C++ supports reference variables

  (kind of variable that is many ways similar to a pointer variable)

```
int x;
int *p = &x; // p is pointer variable
int &r = x;  // r is reference variable
```

- We say that '`p` points to `x`' and '`r` aliases `x`'.

Advantage of reference variable

- Unlike a pointer variable, we may treat `r` as an `int` variable

  -- no dereferencing with * is necessary (see next example)

Function call: C vs C++ (pass by reference)

via pointers

```
void swap(int *x, int *y)
{
      int temp;
      temp = *x;
      *x = *y;
      *y = temp;
}
int main(){
      int a=4, b=5;
      swap(&a, &b);
      return 0;
}
```

via references

```
void swap(int &x, int &y)
{
      int temp;
      temp = x;
      x = y;
      y = temp;
}
int main(){
      int a=4, b=5;
      swap(a, b);
      return 0;
}
```

C supports only via pointers.

C++ supports both.

Demo: Our first C++ program
Compilation and execution

Compilation of a C++ program and its execution

Very similar to compiling a C program.
Only the compiler is g++

- Source code is written in a file with extension .cpp

Example: swap.cpp

- Compilation:

g++ swap.cpp

You can include additional compilation flags

- Execution:

./a.out

Overloaded functions

C++ program can have multiple functions with the same name.

When two or more functions have the same name, the function is said to be overloaded

```
void foo(double x){
      ...
}
void foo(int x){
      ...
}
void foo(int x, int y){
      ...
}
```

Note: C does not support overloaded functions.
So, C compiler will show compilation error.

Overloaded functions: conditions

C++ compiler identifies a function by its signature.

A signature is composed of:

1. Name
2. Number of parameters
3. Type of parameters

```
void foo();
void foo(double x);
void foo(int x);
void foo(int x, int y);
```

These functions have distinct signatures

Essential condition: the functions must have different signatures.

Quiz1

```
void foo(){
    cout << "Foo empty\n";
}
void foo(int a){
    cout << "Foo int\n";
}
void foo(float a){
    cout << "Foo float\n";
}
int foo(int a){
    cout << "Foo ret int\n";
    return a+1;
}
```

Is this code correct?

Quiz1: answer

Signature of a function includes:

1. Name
2. Number of parameters
3. Type of parameters
4. But not return type

```
void foo(){
    cout << "Foo empty\n";
}
void foo(int a){
    cout << "Foo int\n";
}
void foo(float a){
    cout << "Foo float\n";
}
int foo(int a){
    cout << "Foo ret int\n";
    return a+1;
}
```

Both have the same signature.

C++ compilation error due to ambiguity.

Structure: C vs C++

```
struct S1{
    int a, b;
    int *p1=&a;
    int *p2=&b;
};
```

C structure

```
struct S2{
    int a, b;
    int *p1=&a;
    int *p2=&b;
    int &r1=a;
    int &r2=b;

    void swap(int *p1, int *p2);
    void swap(int  r1, int  r2);
};
```

C++ structure:
all of C structure
+ reference variables
+ member functions