From C to C++

CPSC 1070, Donald House, Clemson University
10/18/19

Compiler

c	c++
`gcc`	`g++`

Makefile

C

```
CC = gcc
CFLAGS = -g
DEPS = stack.h
OBJ = trystack.o stack.o

%.o: %.c $(DEPS)
  $(CC) $(CFLAGS) -c -o $@ $<

trystack: $(OBJ)
  $(CC) $(CFLAGS) -o $@ $^

clean:
  rm -f *.o
  rm -f *~
  rm -f trystack
```

C++

```
CC = g++
CFLAGS = -g
DEPS = stack.h
OBJ = trystack.o stack.o

%.o: %.cpp $(DEPS)
  $(CC) $(CFLAGS) -c -o $@ $<

trystack: $(OBJ)
  $(CC) $(CFLAGS) -o $@ $^

clean:
  rm -f *.o
  rm -f *~
  rm -f trystack
```

Include Files

C	C++
`#include <stdio.h>`	`#include <cstdio>`
`#include <stdlib.h>`	`#include <cstdlib>`
`#include <math.h>`	`#include <cmath>`
`#include <string.h>`	`#include <cstring>`
`#include <ctype.h>`	`#include <cctype>`

C I/O vs. Stream I/O

C

Standard open files

```
stdin, stdout, stderr

#include <stdio.h>

int n1, n2;
float fnum;

printf("%f\n", fnum);
printf("(%d, %d)\n", n1, n2);

scanf("%f", &fnum);
scanf("%d %d", &n1, &n2);

fprintf(stderr, "error message\n");
```

C++

Standard open streams

```
cin, cout, cerr

#include <iostream>
using namespace std; // or use std::

int n1, n2;
float fnum;

cout << fnum << endl;
cout << "(" << n1 << ", " << n2 <<
                ")" << endl;
cin >> fnum;
cin >> n1 >> n2;

cerr << "error message" << endl;
```

Dynamic Memory

C

Not part of the language, uses functions defined in `stdlib`

```
#include <stdlib.h>

int *table;
int n;

scanf("%d", &n);
table = (int *)malloc(n * sizeof(int));
. . .
free(table);
```

C++

Built into the language, uses operators `new` and `delete`

```
// note: no include needed

int *table;
int n;

cin >> n;
table = new int[n];
. . .
delete []table; // [] needed for array
```

enums and structs

C

enum and struct variables need the enum or struct prefix when being declared

```
enum colors{RED, GREEN, BLUE};
enum colors tint;

struct point{
  int x, y;
};
struct point pt;
```

or you could use a typedef:

```
typedef struct _point{
  int x, y;
} point;
point pt;
```

C++

enum and struct variables do not need the enum or struct prefix

```
enum colors{RED, GREEN, BLUE};
colors tint;

struct point{
  int x, y;
};
point pt;
```

function parameters

C

All parameter passing is done by value. If you need to affect the variable referred to by the actual parameter, you need to use a pointer in the function, and pass the address of the actual parameter

```
void swap(int *a, int *b){
  int temp;
  temp = *a; *a = *b; *b = temp;
}

int m, n;
. . .
swap(&m, &n);
```

C++

In C++ parameter passing is done by value by default, but you can pass by reference by annotating the parameter with an &

```
void swap(int &a, int &b){
  int temp;
  temp = a; a = b; b = temp;
}

int m, n;
. . .
swap(m, n);
```

An Example C vs. C++
Bubble Sort Program

Download from the schedule page