C vs. Java

• Design Goals:

• Implications:
C vs. C++

C++ $\supseteq$ C

C $\subseteq$ C++
Primary Differences

• C does not have iostreams

• C does not have the new keyword

• C does not have implicit pointers (by-ref)
Primary Differences

• Purely* procedural

• Strict scoping requirements

• No string type, only “C-strings”
Program #1a

void main()  {
    int *p;
    *p = 4;
}

malloc

• **malloc**: memory allocator
  – `void *malloc(size_t size)`
    • Allocates memory space on the heap
      – *Contents of the memory is unknown! Don’t assume it contains zeros (0x00).*
    • Returns a pointer to the newly allocated space
    • Functionally equivalent to the C++ `new` keyword
    • Pointer must be sent to `free()` to free the memory
      – `void free(void *ptr)`

• Usage:
Pointers

• A pointer variable is an ordinary variable that contains a memory address.

```c
int *k;    double *j;    void *p;
```
void main() {
    int *p;
    *p = 4;
    printf("The value of p is: %d\n", p);
}

printf()

printf("%s %d Hi %f\n", 
    a, b, c);

%s: C-string
%d: Integer (digits)
%ld: Longs (long digits)
%f: Floating point number
%p: Pointer (0x00CB)
%c: Character
void main() {
    int a = 42;
    int *b;

    b = a;
}
Program #3

```c
const int size = 10;
void main()
{
    int **values;

    for (i = 0; i < size; i++)
    {
        for (j = 0; j < size; j++)
    }
}
```
void main() {
    char *h = "Hello";
    char *w = "World";

    char *s = h + w;

    printf("%s\n", hw);
}