NU pre

Friday, January 6, 2023 1:33 PM

-interpreted vs compiled for java
JDK - convert plain text files(.java) to Java bytecode files(.class)
Java JVM - takes java byte files(.class) and convert them to machine code for machine you are currently on
javac

-compile/link/execute for C
-compiler itself is a program
gcc helloworld.c -o helloworld
./helloworld

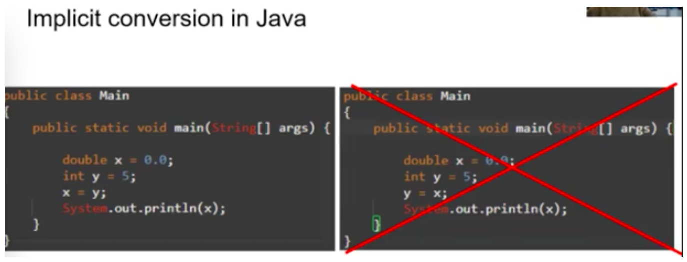
-debug in java

- convention error
  readable
- logic error
- syntax error : google the error

- isolate problem, start at first error message, lots of print statements, get to know your debugger, simplify your code, clear understanding

-runtime error
-debugging C code

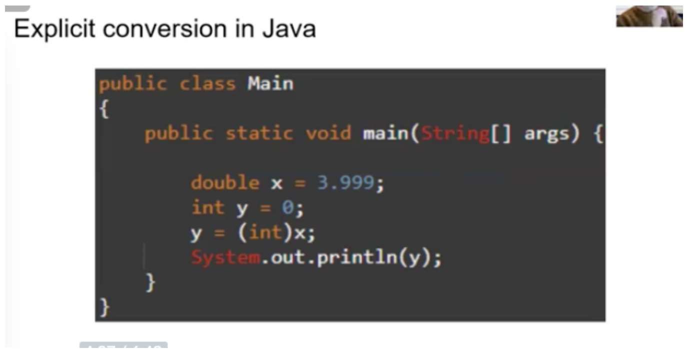
-types and typing in Java
-boolean(1bit), char(2 byte), byte, short(2 byte), int(4 byte), long(8 byte), float(4 byte), double(8 byte)



Implicit conversion in Java

```
public class Main
{
    public static void main(String[] args) {

        double x = 0.0;
        int y = 5;
        x = y;
        System.out.println(x);
    }
}
```

```
public class Main
{
    public static void main(String[] args) {

        double x = 0.0;
        int y = 5;
        y = x;
        System.out.println(x);
    }
}
```



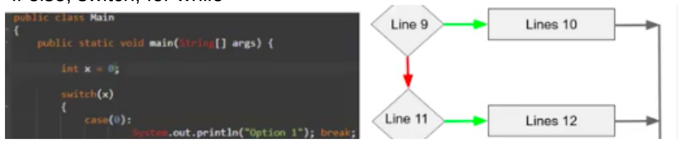
Explicit conversion in Java

```
public class Main
{
    public static void main(String[] args) {

        double x = 3.999;
        int y = 0;
        y = (int)x;
        System.out.println(y);
    }
}
```

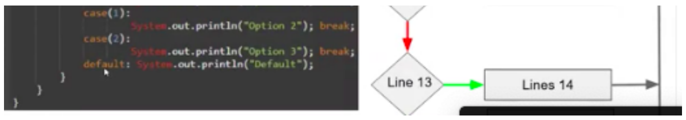
something is going to lost

-C, static vs dynamic typing
-C typing

-control in Java
-if else, switch, for while



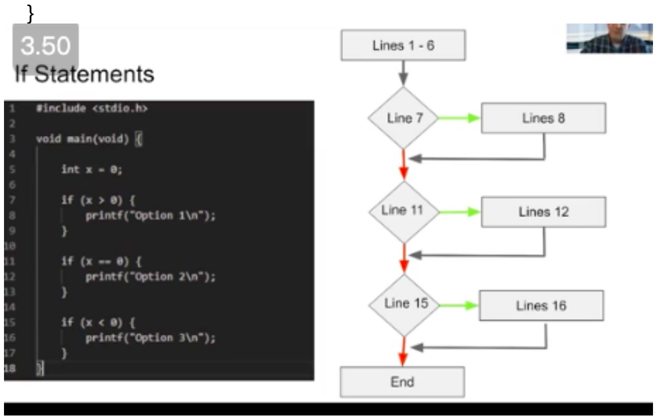
```
public class Main
{
    public static void main(String[] args) {

        int x = 0;

        switch(x)
        {
            case(0):
                System.out.println("Option 1"); break;
```



```
            case(1):
                    System.out.println("Option 2"); break;
            case(2):
                    System.out.println("Option 3"); break;
            default: System.out.println("Default");
        }
    }
}
```

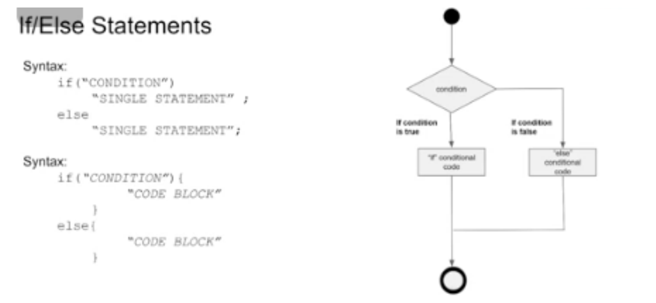
-Control in C
-selection: if, if/else
-iteration
-unconditional

-if("..."){
 "...";
 }



3.50
If Statements

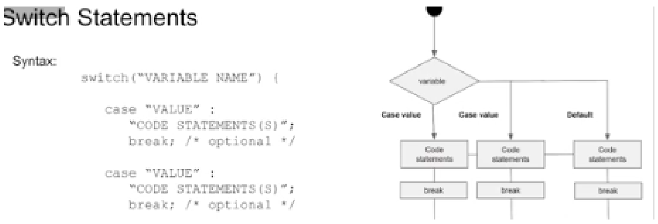
```
#include <stdio.h>

void main(void) {

    int x = 0;

    if (x > 0) {
        printf("Option 1\n");
    }

    if (x == 0) {
        printf("Option 2\n");
    }

    if (x < 0) {
        printf("Option 3\n");
    }
}
```



If/Else Statements

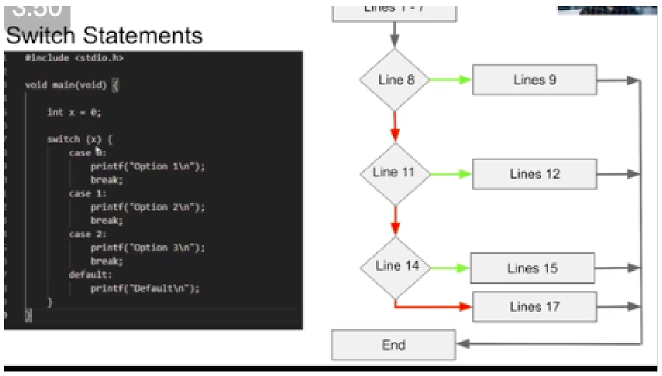
Syntax:
```
    if("CONDITION")
        "SINGLE STATEMENT" ;
    else
        "SINGLE STATEMENT";
```

Syntax:
```
    if("CONDITION"){
        "CODE BLOCK"
    }
    else{
        "CODE BLOCK"
    }
```



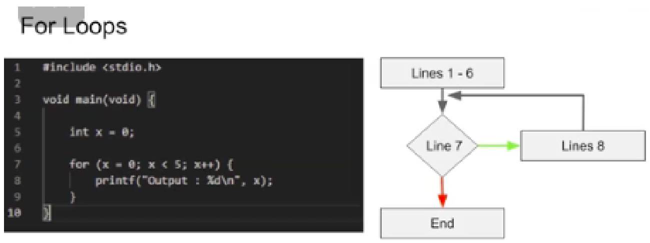
Switch Statements

Syntax:
```
    switch("VARIABLE NAME") {

        case "VALUE" :
            "CODE STATEMENTS(S)";
            break; /* optional */

        case "VALUE" :
            "CODE STATEMENTS(S)";
            break; /* optional */
```



3.50
Switch Statements

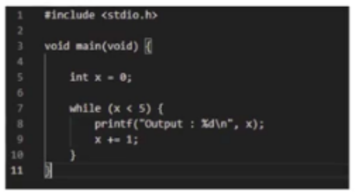
```
#include <stdio.h>

void main(void) {

    int x = 0;

    switch (x) {
        case 0:
            printf("Option 1\n");
            break;
        case 1:
            printf("Option 2\n");
            break;
        case 2:
            printf("Option 3\n");
            break;
        default:
            printf("Default\n");
    }
}
```



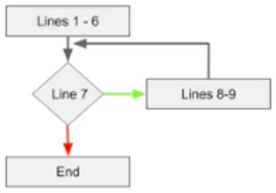
For Loops

```
#include <stdio.h>

void main(void) {

    int x = 0;

    for (x = 0; x < 5; x++) {
        printf("Output : %d\n", x);
    }
}
```

While Loops



```c
#include <stdio.h>

void main(void) {

    int x = 0;

    while (x < 5) {
        printf("Output : %d\n", x);
        x += 1;
    }
}
```



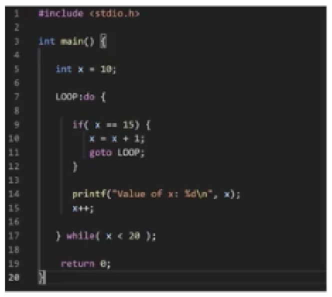
6.00

Do-While Loops

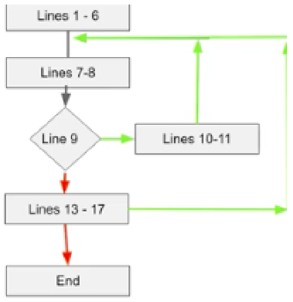
Syntax:

```
do {
    "CODE BLOCK"
} while("CONDITION")
```

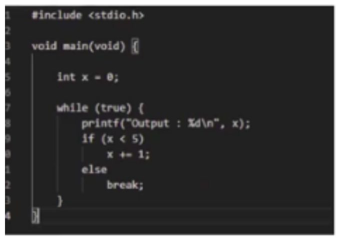
Goto



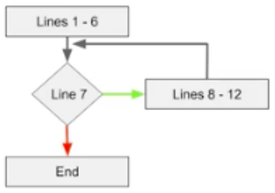
```c
#include <stdio.h>

int main() {

    int x = 10;

    LOOP:do {

        if( x == 15) {
            x = x + 1;
            goto LOOP;
        }

        printf("Value of x: %d\n", x);
        x++;

    } while( x < 20 );

     return 0;
}
```



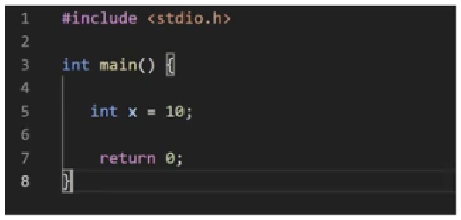
Break



```c
#include <stdio.h>

void main(void) {

    int x = 0;

    while (true) {
        printf("Output : %d\n", x);
        if (x < 5)
            x += 1;
        else
            break;
    }
}
```



Return



```c
#include <stdio.h>

int main() {

    int x = 10;

     return 0;
}
```

-pointer and memory visualization in C
-pointers
sizeof() function

-pointers

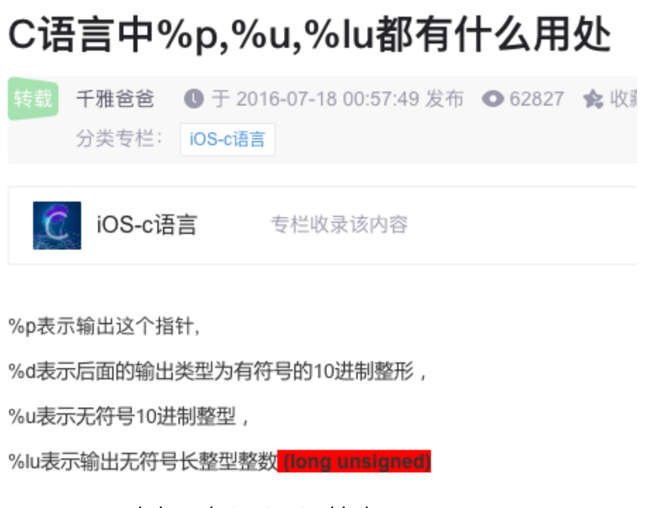
- are variables, stored in memory, size of () function
- * operator reference
  & operator address
- relative addressing
- pointer arithmetic / substraction

- *arr is arr[0]
- scanf, printf
- pass array to functions

- pointers

```
// what is a pointer
// what is a variable - a variable is a named location in memory store address of other variable
// variable name really masks the address reference by the variable
#include <stdio.h>
void introPointer(void);
void pointerMath(void);
int main(void){
}
void introPointer(void){
    int a = 7;
    printf("a is %d Reference address %p", a, &a);
    printf("Size in bytes for int: %lu", sizeof(int));
    // pointers
    int * pA = &a; // 0 is only value assign to pointer
    printf("value in mesmory: %p reference address: %p", pA, &pA);
    // dereferencing a pointer
    // *pA = 10;
    printf("value in mesmory: %d reference address: %p", *pA, pA);
    *pA = *pA + 10;
    printf("New value in a: %d \n", a);
    return ;
}
void pointerMath(void){
    int b = 8;
    int * pB = &b;
    printf("value in mesmory: %d reference address: %lu\n", *pB, pB);
    printf("value in mesmory: %d reference address: %lu\n", *pB+1, pB+1);
    printf("Size in bytes for int: %lu \n", sizeof(int));")
    return;
}
```

-char is 1 bit
-long is 8 bit
-int is 4 bit


C语言中%p,%u,%lu都有什么用处

转载 千雅爸爸 于 2016-07-18 00:57:49 发布 62827 收藏

分类专栏： iOS-c语言

iOS-c语言 专栏收录该内容

%p表示输出这个指针,

%d表示后面的输出类型为有符号的10进制整形，

%u表示无符号10进制整型，

%lu表示输出无符号长整型整数 (long unsigned)

-memory can only be substract not added

-Pointer Arithmetic

```
#include <stdio.h>
void introPointer(void);
void pointerMath(void);
void pointerArray(void);

int main(void){
}
void introPointer(void){
    int a = 7;
    printf("a is %d Reference address %p", a, &a);
    printf("Size in bytes for int: %lu", sizeof(int));
    // pointers
    int * pA = &a; // 0 is only value assign to pointer
    printf("value in mesmory: %p reference address: %p", pA, &pA);
    // dereferencing a pointer
    // *pA = 10;
    printf("value in mesmory: %d reference address: %p", *pA, pA);
    *pA = *pA + 10;
    printf("New value in a: %d \n", a);
    return ;
}
void pointerMath(void){
    int b = 8;
    int * pB = &b;
    printf("value in mesmory: %d reference address: %lu\n", *pB, pB);
    printf("value in mesmory: %d reference address: %lu\n", *pB+1, pB+1);
    printf("Size in bytes for int: %lu \n", sizeof(int));")
    return;
}

void pointerArray(void){
        // an array is a contigious memory that stores homogenous collection2
}
```