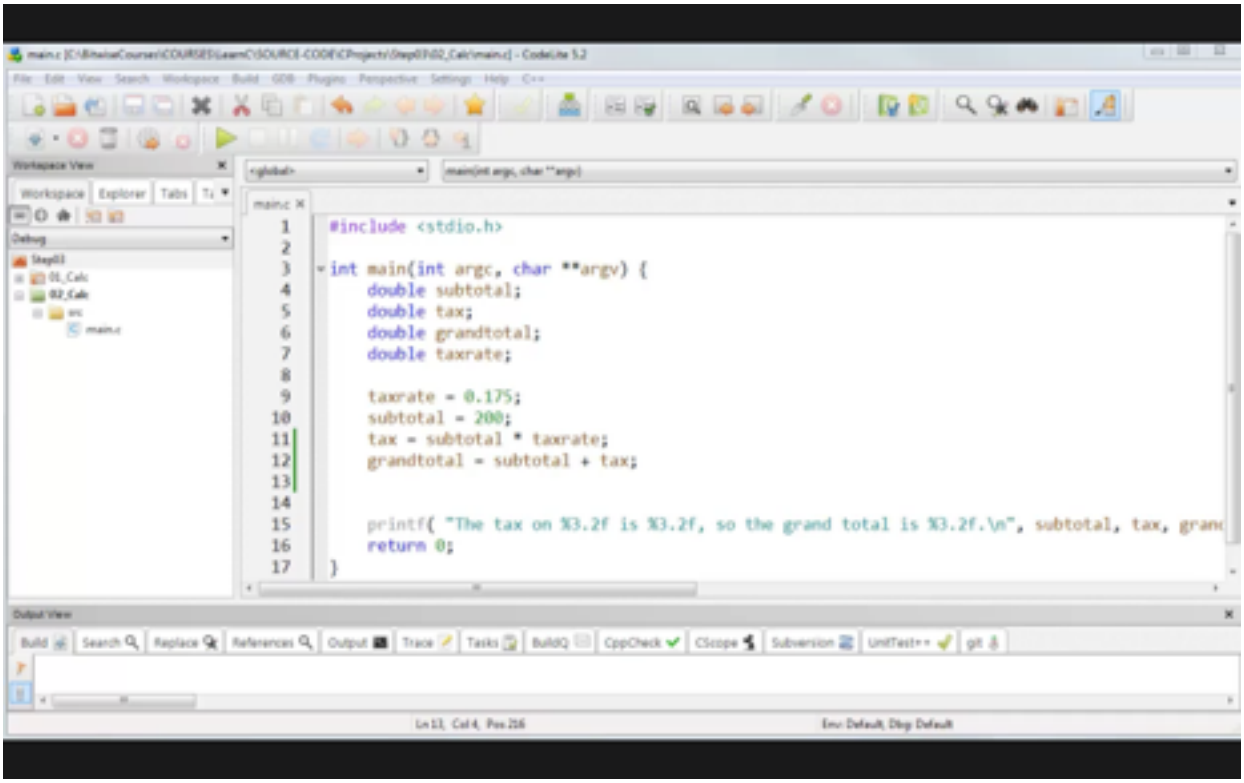
LESSON 1: Getting ready

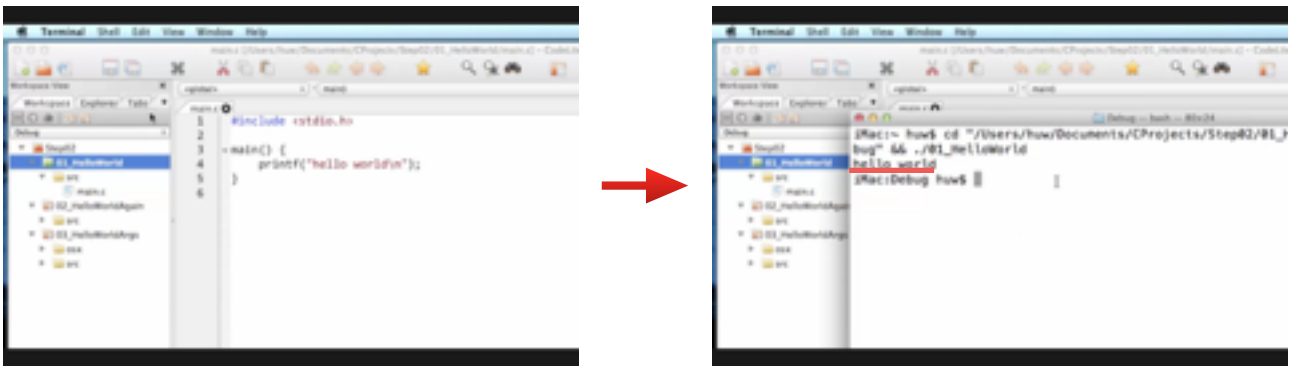
- In order to follow this course, you will need to download an IDE first. There are many IDEs available on the internet but the one I will be using for this course is called CodeLite as it is easy to install and use. You can get the IDE for both Mac and Windows from the following link http://codelite.org/.
- INTRODUCTOIN TO CODELITE:



Overview

- The very first thing you'll notice from this code is that it's coloured. This is syntax sensitive; each type of date is differently coloured, for example the names of variables are coloured in brown, the numbers are coloured in green and the types of date are coloured in blue.

LESSON 2: programming basics



Your first programme “HelloWorld

#include <stdio.h>

- “stdio.h” is a built-in feature in C that provides a number of functions or capabilities that you’ll want to use time and time again in C
- #include (hash include) just simply tells the processor to include the contents of whatever that comes after it (in this case the contents of “stdio.h)

main() (function main)

- every C programme has this, when the programme is run function main is the thing that executes first

curly arrows {}

- curly arrows mark the start and the end of a function
- between them is a block of codes

printf()

- printf() takes strings as arguments
- it prints out everything that is between the brackets
- format: printf(“strings”);
- “ “ indicates that what is between them is a string
- semicolon (;) indicates the end of a statement

comments

- sometimes the code written by you is not obvious to other programmers, so comments are used to make your codes more understandable

```
main()
{
  /* I changed the following line. */
  printf("Hello, Johan!\n");
  return 0;
}
```

/* comments */ is used to define a comment. The processor won’t take this into account at call. Comments could be anything and are only for human eyes not computers.

LESSON 3: variables, constants and types

- when you need to store some data in your programmes, you'll need to declare some **variables**
- A variable is simply a name to which some values can be assigned to. It's just like box with a label called, for example, petty cash.

```c
#include <stdio.h>

int main(int argc, char **argv) {
    double pettycash;
    double grandtotal;

    grandtotal = 500.50;
    pettycash = 10.5;
    printf("pettycash=%.2f\n", pettycash);
    pettycash = 100.25;
    printf("pettycash=%.2f\n", pettycash);
    printf("grandtotal=%.2f\n",  grandtotal);
    return 0;
}
```

Format of a variable:

type of data stored + variable name + = + data you want to store.

Types

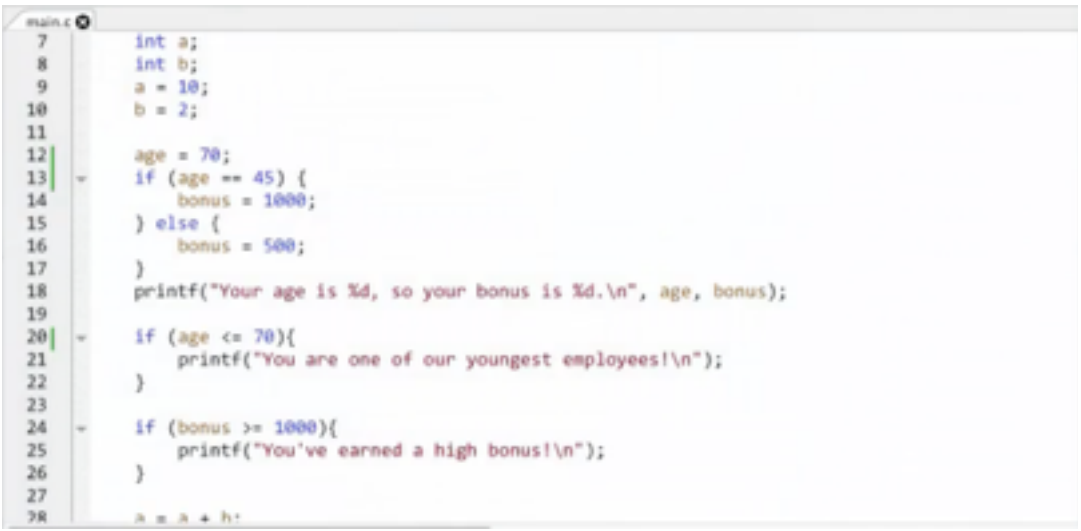
1. int type: to store integers
2. double type: to store floating-point numbers
3. char type: to store strings

Naming conventions

- variable names usually begin with a lower case letter
- spaces are not allowed, but conventionally underscores (_) or an uppercase letter is used as a replacement.

LESSON 4: Operators, tests and user input

Operators are specific symbols that are used to do operations i.e. addition, subtraction, multiplication and division.



```c
int a;
int b;
a = 10;
b = 2;

age = 70;
if (age == 45) {
    bonus = 1000;
} else {
    bonus = 500;
}
printf("Your age is %d, so your bonus is %d.\n", age, bonus);

if (age <= 70){
    printf("You are one of our youngest employees!\n");
}

if (bonus >= 1000){
    printf("You've earned a high bonus!\n");
}
```

- one equal sign (=) is used to assign a value to a variable
- two equal signs (==) are used to test a condition
- comparison operators; ==, <= and >=
- compound assignment operators; a += b -> a = a + b
- increment and decrement operators; a++, a—
- logical operators; && (and) and ll (or)

```
if (condition) {
    statement;
} else if (condition) {
    statement;
}  else {
    statement;
}
```

```c
a = a * b;
printf( "a * b  : %d\n", a );
a = 10;
a *= b;
printf( "a *= b  : %d\n", a );
a = 10;
a = a / b;
printf( "a/b  : %d\n", a );
a = 10;
a /= b;
printf( "a /= b  : %d\n", a );
a = 10;
a++;
printf( "a++ : %d\n", a );
a = 10;
a--;
printf( "a-- : %d\n", a );

showPrefixAndPostfixOps();
return(0);
}
```

```c
#include <stdio.h>

int main(int argc, char **argv) {
    int age;
    int number_of_children;
    double salary;

    age = 25;
    number_of_children = 1;
    salary = 20000.00;

    if ((age <= 30 ) && (salary >= 30000)) {
        printf( "You are a rich young person\n" );
    } else {
        printf( "You are not a rich young person\n" );
    }

    if ((age <= 30 ) || (salary >= 30000)) {
        printf( "You are either rich or young or both\n" );
    } else {
        printf( "You are not neither rich nor young\n" );
    }
```

User inputs
- scanf("%d", &n);

LESSON 5: Functions, arguments and switch

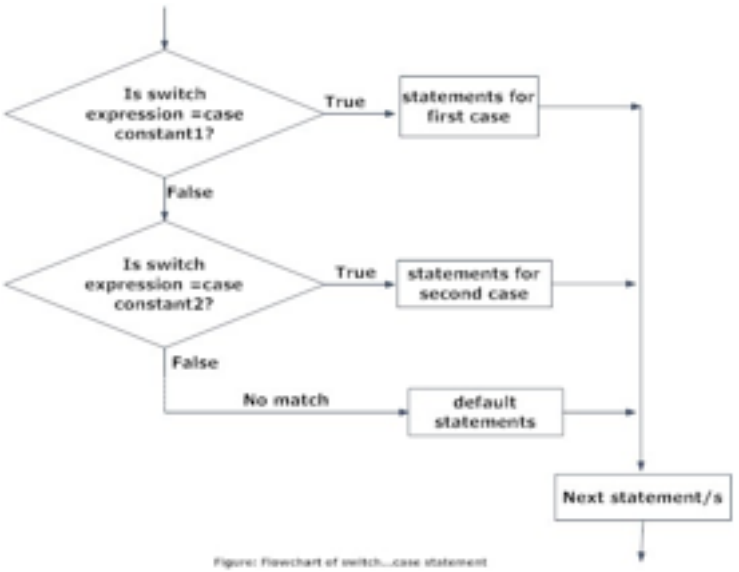
- **Functions** serve one main valuable purpose; it allows you divide your code into smaller pieces.
- To define a function:

type of the data you want to return + function name (argument type + argument name) {
    statement;
}

```c
#include <stdio.h>

int main(int argc, char **argv) {
    int age;
    int number_of_children;
    double salary;

    age = 25;
    number_of_children = 1;
    salary = 20000.00;

    if ((age <= 30 ) && (salary >= 30000)) {
        printf( "You are a rich young person\n" );
    } else {
        printf( "You are not a rich young person\n" );
    }

    if ((age <= 30 ) || (salary >= 30000)) {
        printf( "You are either rich or young or both\n" );
    } else {
        printf( "You are not neither rich nor young\n" );
    }
```

SWITCH STATEMENTS

```c
# include <stdio.h>
int main() {
    char o;
    float num1,num2;
    printf("Select an operator either + or - or * or / \n");
    scanf("%c",&o);
    printf("Enter two operands: ");
    scanf("%f%f",&num1,&num2);
    switch(o) {
        case '+':
            printf("%.1f + %.1f = %.1f",num1, num2, num1+num2);
            break;
        case '-':
            printf("%.1f - %.1f = %.1f",num1, num2, num1-num2);
            break;
        case '*':
            printf("%.1f * %.1f = %.1f",num1, num2, num1*num2);
            break;
        case '/':
            printf("%.1f / %.1f = %.1f",num1, num2, num1/num2);
            break;
        default:
            /* If operator is other than +, -, * or /, error message
            printf("Error! operator is not correct");
            break;
    }
    return 0;
}
```



http://www.programiz.com/c-programming/c-switch-case-statement

```
Enter operator either + or - or * or /
*
Enter two operands: 2.3
4.5
2.3 * 4.5 = 10.3
```

OUTPUT

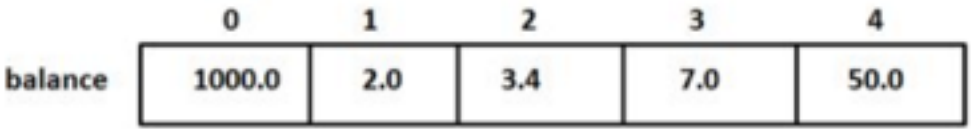
LESSON 6: Arrays, loops and break

- Arrays a kind of data structure that can store a fixed-size sequential collection of elements of the same type. An array is used to store a collection of data, but it is often more useful to think of an array as a collection of variables of the same type
- Declaring an array:

```
type arrayName [ arraySize ];
```

- Initialising an array:

```
double balance[] = {1000.0, 2.0, 3.4, 7.0, 50.0};
```



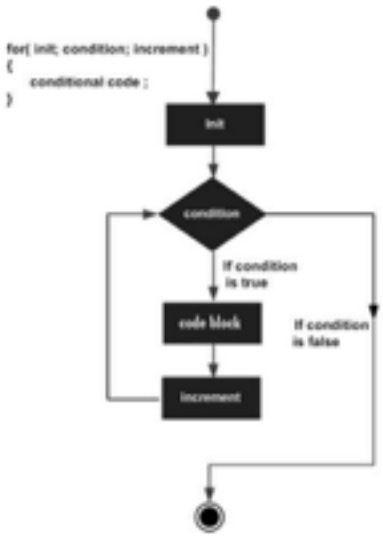
- Accessing an element:

```
double salary = balance[9];
```

for loop:

```
for ( init; condition; increment ) {
   statement(s);
}
```

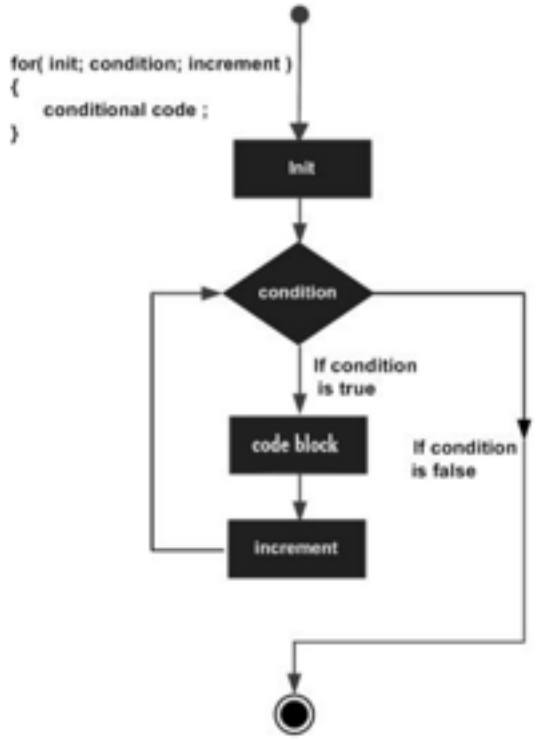
Flow Diagram



while loop:

```
while(condition) {
   statement(s);
}
```

Flow Diagram



do while loop:

```
do {
   statement(s);
} while( condition );
```

Flow Diagram