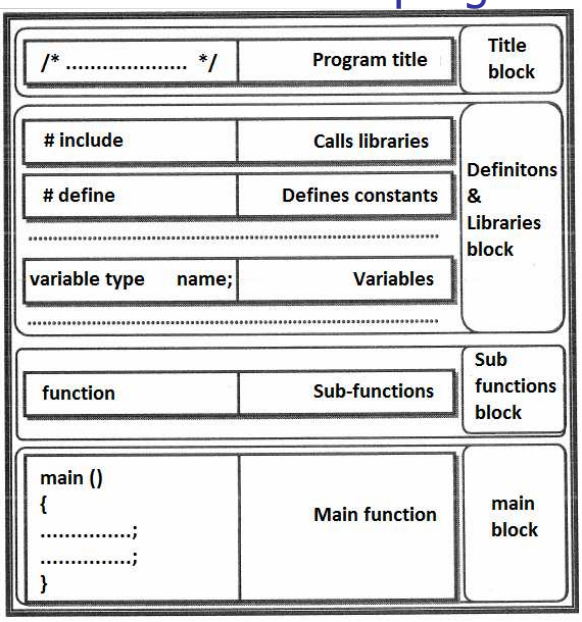
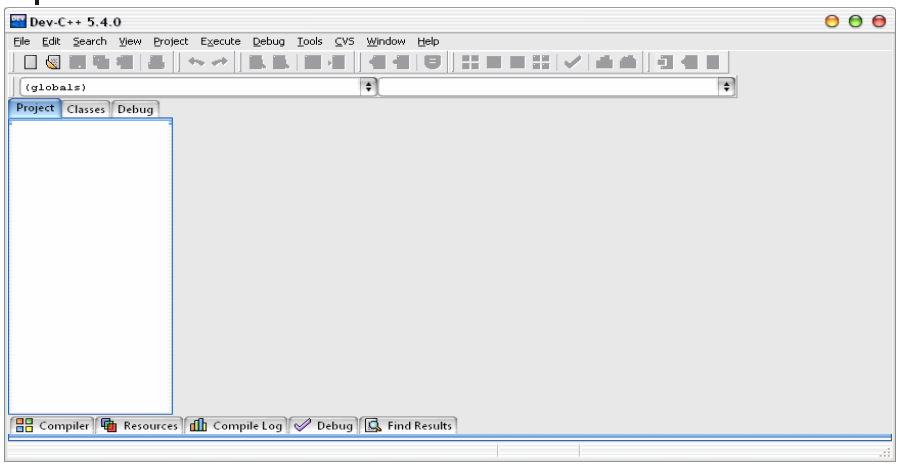
C Programming – Lecture I

Instructor Özgür ZEYDAN
http://cevre.beun.edu.tr/zeydan/

Structure of a C program

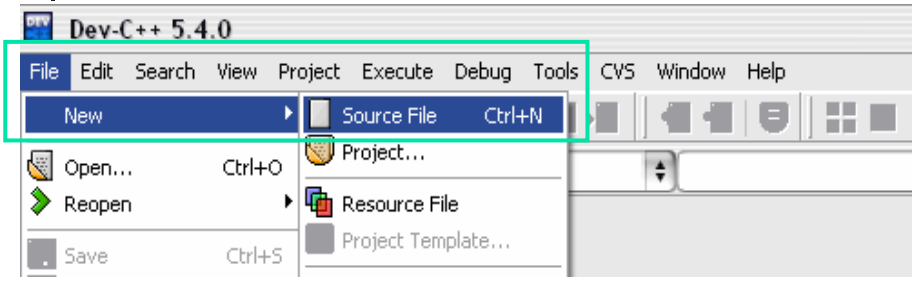


C Compiler: Dev C++

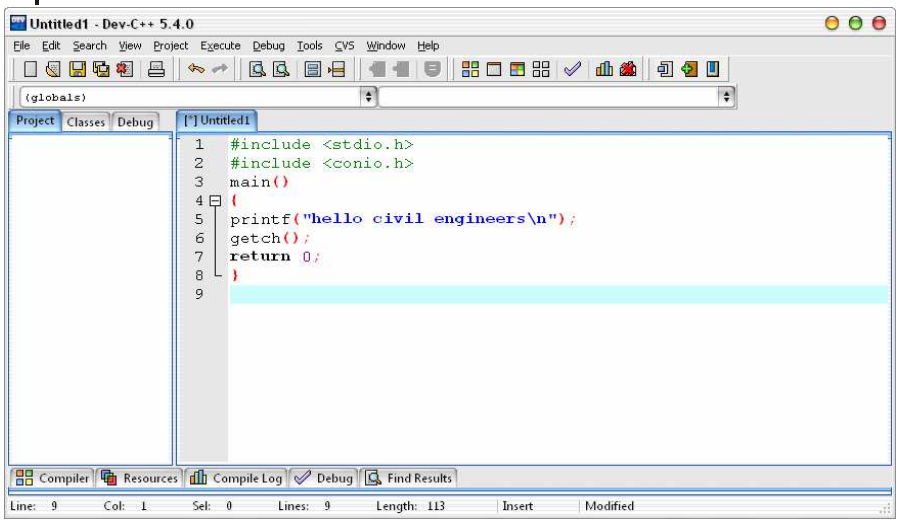


Download latest version from following link:
http://sourceforge.net/projects/orwelldevcpp/

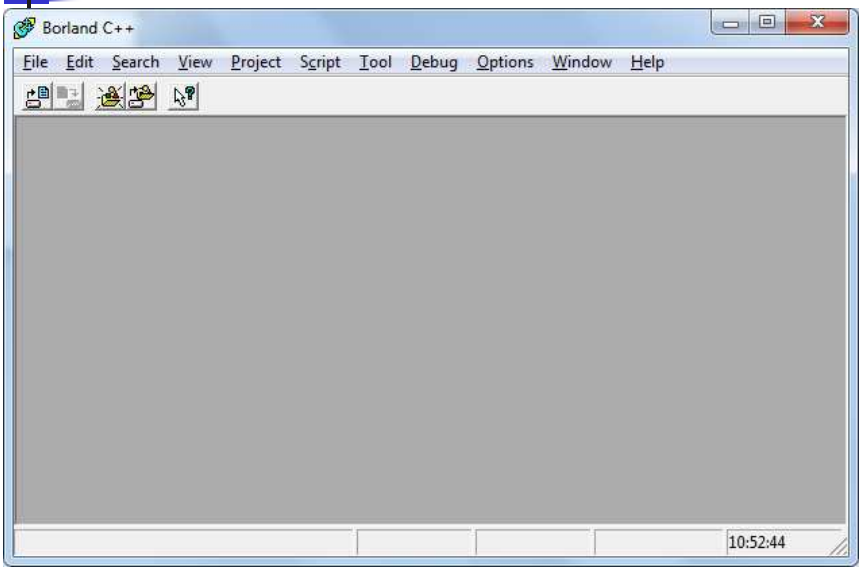
Creating a new file



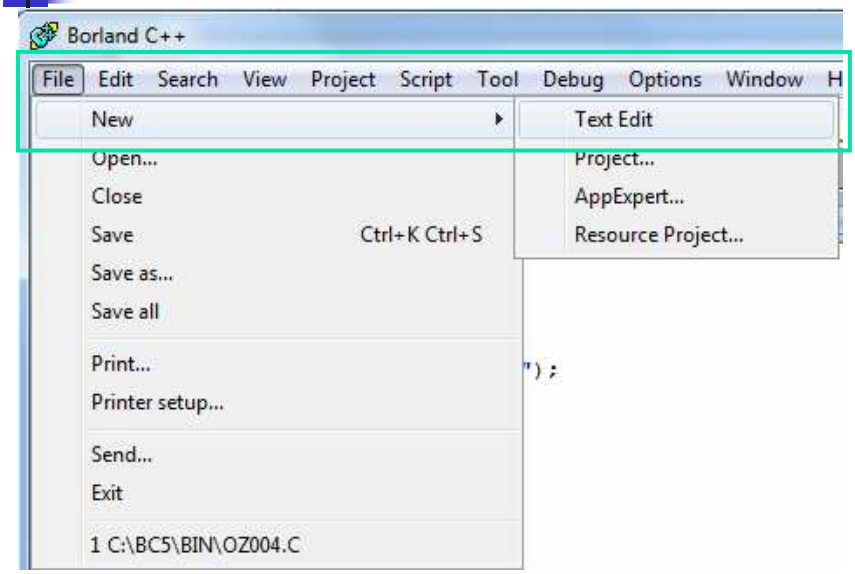
Your First C Program



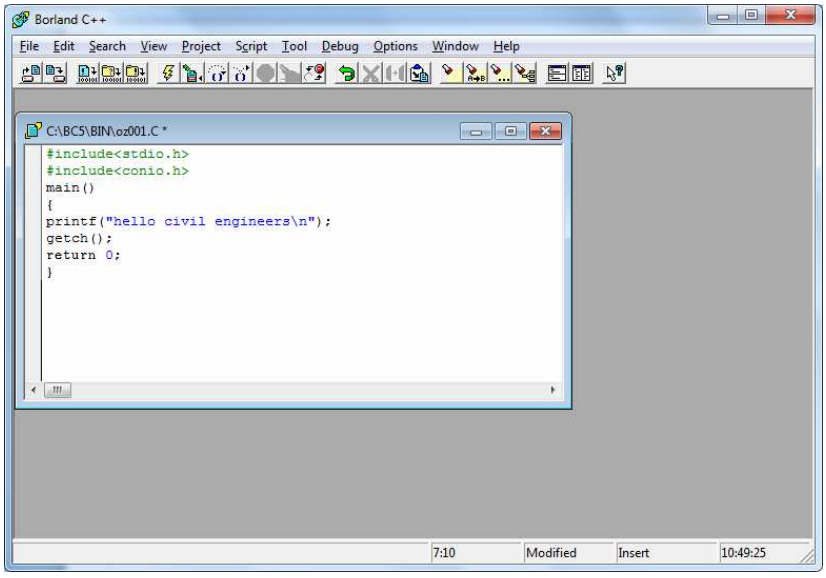
C Compiler: Borland C++



Creating a new file



Your First C Program

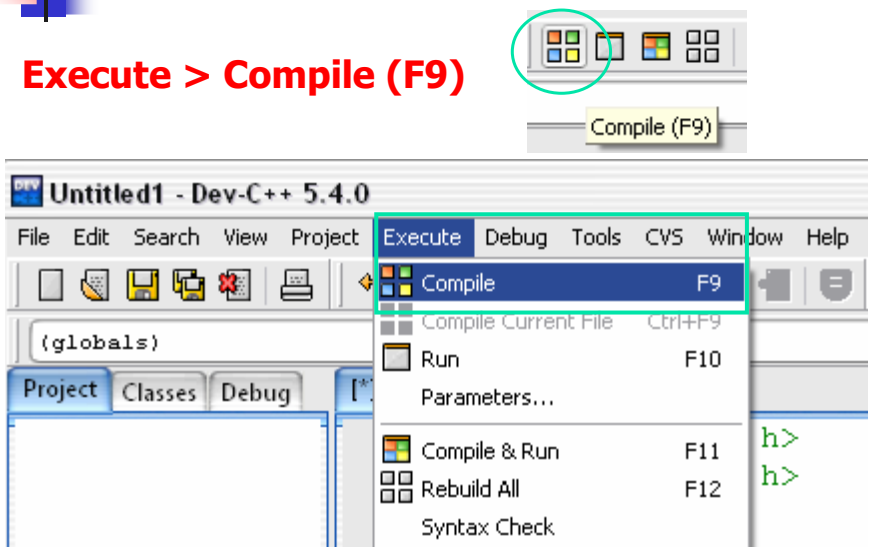


Your First C Program

```
#include <stdio.h>
#include <conio.h>
main()
{
printf("hello civil
engineers\n");
getch();
return 0;
}
```

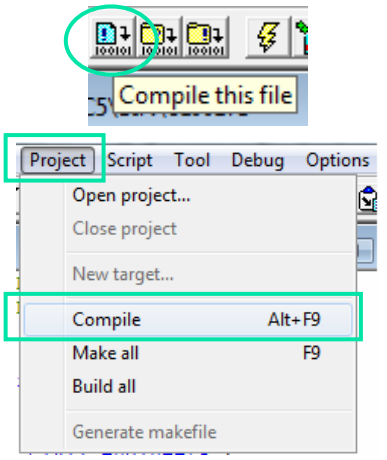
Compiling your first program

Execute > Compile (F9)

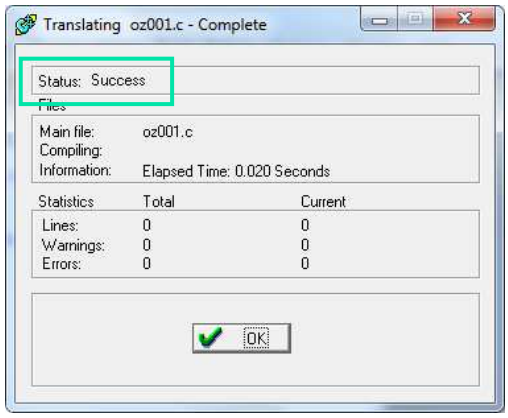


Compiling your first program

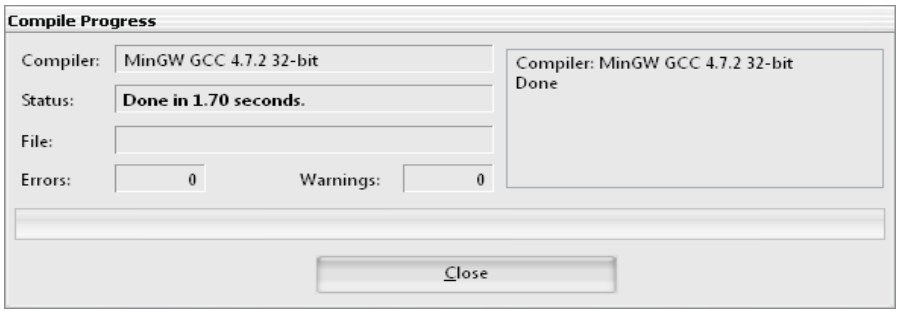
Project > Compile (ALT + F9)



Compile Screen

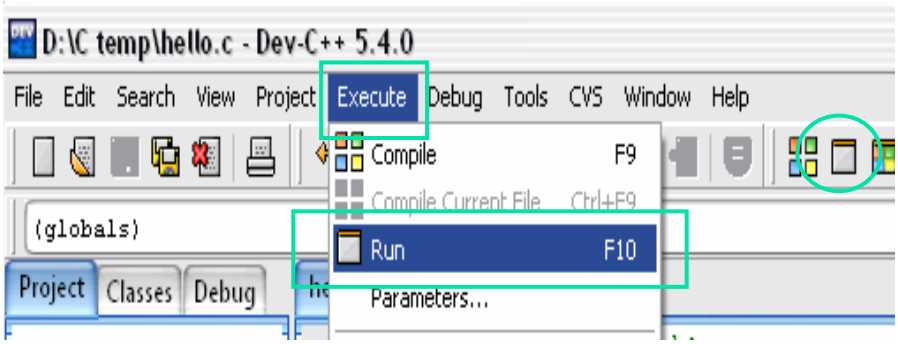


Compile Screen

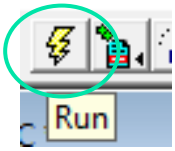


Running your first program

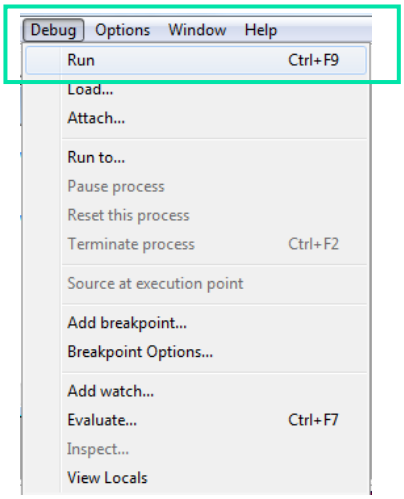
Execute > Run (F10)



Running your first program

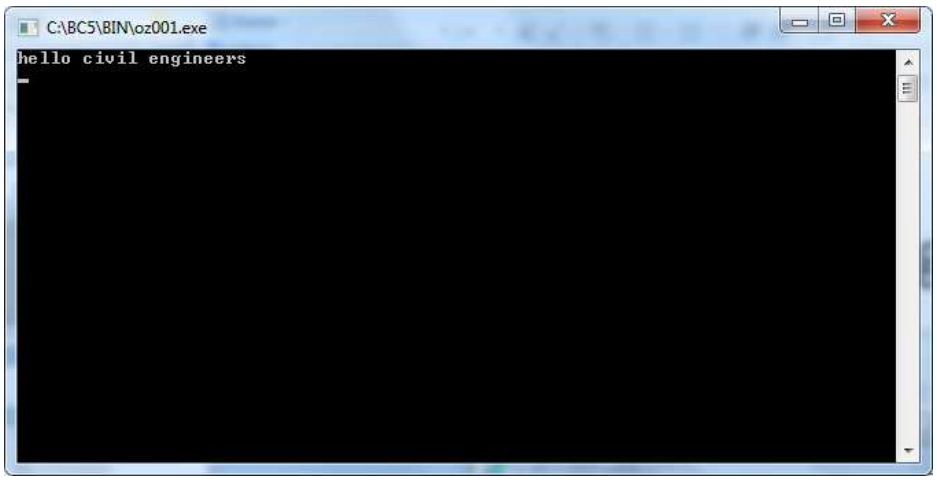


Debug > Run (CTRL + F9)



Output Screen

Output Screen



#include<stdio.h>

- This C program starts with **`#include <stdio.h>`**
- This line includes the "standard I/O library" into your program.
- The standard I/O library lets you:
  - read input from the keyboard,
  - write output to the screen,
  - process text files stored on the disk, and so on.
- It is an extremely useful library.
- C has a large number of standard libraries like stdio, including string, time and math libraries.

Headers

- .h files are called header files
- **`#include<stdio.h>`**
  - **`printf()`** → output command
  - **`scanf()`** → input command
- **`#include<conio.h>`**
  - **`getch()`** → reads key (program waits until key pressed)
  - **`clrscr()`** → clears output screen (Borland C++ compiler only)

Clearing Screen

- Borland C++ compiler
- Dev C++ IDE

```
#include<conio.h>
main()
{
clrscr();
}
```

```
#include<stdlib.h>
main()
{
system("cls");
}
```

main() function

- The line **`int main()`** declares the main function.
- Every C program must have a function named main somewhere in the code.
- At run time, program execution starts at the first line of the main function.
- In C, the **`{`** and **`}`** symbols (block delimiters) mark the beginning and end of a block of code.

printf()

- The **`printf`** statement in C allows you to send output to standard out (for us, the screen).
- The portion in quotes is called the format string and describes how the data is to be formatted when printed.
- The format string can contain:
  - string literals such as "Hello civil engineers"
  - symbols for carriage returns (**`\n`**)
  - operators as placeholders for variables.

return 0;

- The **`return 0;`** line causes the function to return an error code of 0 (no error) to the shell that started execution.
- `;` symbol is statement terminator.

Modifying first program

```
/* Programmer : Özgür ZEYDAN */
#include <stdio.h>
#include <conio.h>
main()
{
printf("hello civil engineers\n");
getch();
printf("\nhello civil engineers");
getch();
return 0;
}
```

Comments

- **`/* Programmer : Özgür ZEYDAN */`**
- Comments can be inserted into C programs by bracketing text with the `/*` and `*/` delimiters.
- Comments are useful for a variety of reasons.
- Primarily they serve as internal documentation for program structure and functionality.
- Best programmers comment as they write the code, not after the fact.

Variables

- As a programmer, you will frequently want your program to "remember" a value. For example, if your program requests a value from the user, or if it calculates a value, you will want to remember it somewhere so you can use it later. The way your program remembers things is by using variables. For example:

```
int b;
```

- This line says, "I want to create a space called b that is able to hold one integer value." A variable has a name (in this case, b) and a type (in this case, int, an integer). You can store a value in b by saying something like:

```
b = 5;
```

- You can use the value in b by saying something like:

```
printf("%d", b);
```

Variable Names

- Variable names starts with a letter or underscore "_" and followed by either letters or digits or underscore "_".
- Variable names can not start with digits or special characters.
- C is a case sensitive language. It means variables SUM is different than Sum or sum.
- You can not use Reserved Words as variable names.

Reserved Words

Keywords			
auto	**double**	**int**	**struct**
break	**else**	**long**	**switch**
case	**enum**	**register**	**typedef**
char	**extern**	**return**	**union**
const	**float**	**short**	**unsigned**
continue	**for**	**signed**	**void**
default	**goto**	**sizeof**	**volatile**
do	**if**	**static**	**while**

Example - 1

```
#include <stdio.h>
#include <conio.h>
int main() {
    int a, b, c;
    a = 5;
    b = 7;
    c = a + b;
    printf("%d + %d = %d\n", a, b, c);
    getch();
    return 0; }
```

Explanation of Example - 1

- The line `int a, b, c;` declares three integer variables named **a**, **b** and **c**. Integer variables hold whole numbers.
- The next line initializes the variable named **a** to the value 5.
- The next line sets **b** to 7.
- The next line adds **a** and **b** and "assigns" the result to **c**.
- The `printf` statement then prints the line "5 + 7 = 12."

Placeholders

- The `%d` placeholders in the `printf` statement act as placeholders for values.
- There are three %d placeholders, and at the end of the printf line there are the three variable names: **a**, **b** and **c**.
- C matches up the first %d with a and substitutes 5 there. It matches the second %d with b and substitutes 7. It matches the third %d with c and substitutes 12.
- Then it prints the completed line to the screen: 5 + 7 = 12.

Example - 2

- In previous example, user can not change the values of a and b.
- The program always use a = 5 and b = 7.
- Let us write a better program:
  - Programs asks user to write first number and then initialize the value of a to that number.
  - Programs asks user to write first number and then initialize the value of b to that number.
  - Then program calculates c as c = a + b.
  - Finally program displays the output.

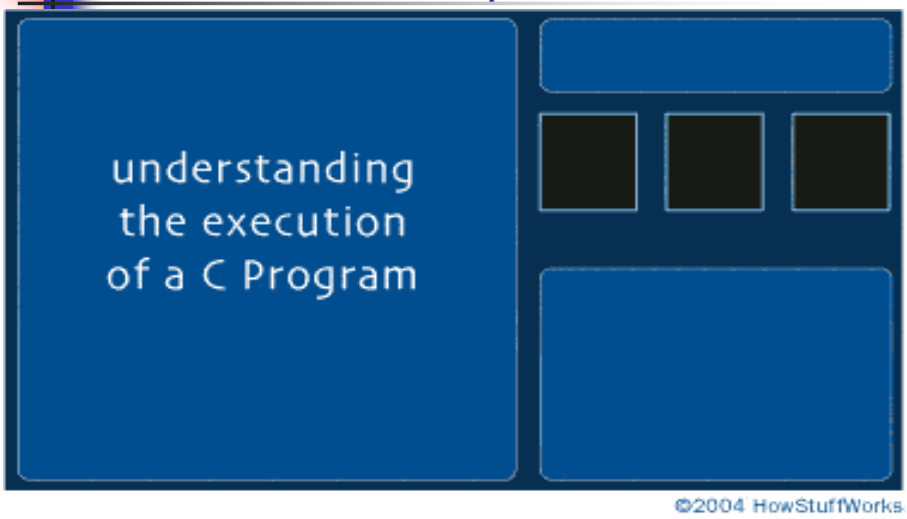
Example – 2 (Pseudocode)

Start
Use Variables: a, b and c
Display "write a number"
Read a
Display "write a number"
Read b
Calculate c = a+b
Display "c = a+b"
Stop

Example – 2 (C code)

```
#include <stdio.h>
#include <conio.h>
int main() {
    int a, b, c;
    printf("Enter the first value:");
    scanf("%d", &a);
    printf("Enter the second value:");
    scanf("%d", &b);
    c = a + b;
    printf("%d + %d = %d\n", a, b, c);
    getch();
    return 0; }
```

Example – 2



http://www.howstuffworks.com/c.htm

Scanf()

- The **`scanf` function allows you to accept input from standard in**, which for us is generally the keyboard.
- Note that scanf uses the same sort of format string as printf . Also note the **`&`** in front of a and b.
- This is the **address operator** in C: It returns the address of the variable.
- You must use the & operator in scanf on any variable of type char, int, or float, as well as structure types.
- If you leave out the & operator, you will get an error when you run the program.

Placeholders

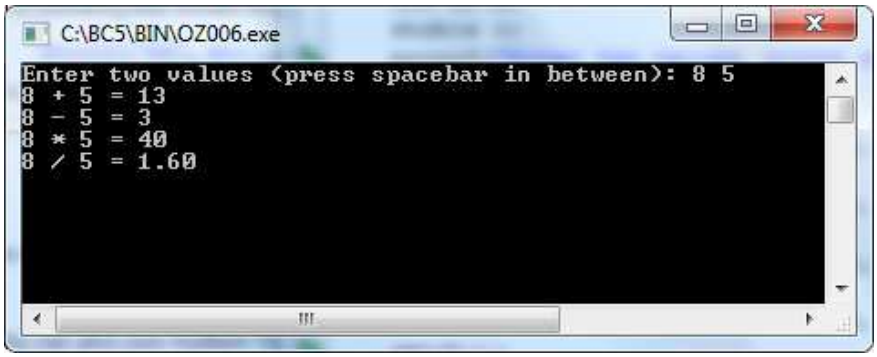
- You can **print all of the normal C types with `printf`** by using different placeholders:
  - **int** (integer values) uses **`%d`**
  - **float** (floating point values) uses **`%f`**
  - **double** (decimal values) uses **`%f`**
  - **char** (single character values) uses **`%c`**
  - **character strings** (arrays of characters) use **`%s`**
  - **double** (decimal values) uses **`%lf`** in **`scanf`**

Example – 2 (Better C code)

```
#include <stdio.h>
#include<conio.h>
int main() {
    int a, b, c;
    printf("Enter two values (press spacebar in between): ");
    scanf("%d %d", &a, &b);
    c = a + b;
    printf("%d + %d = %d\n", a, b, c);
    getch();
    return 0;
}
```

Example - 3

Write a C program that gives following output screen.
Make all necessary changes.



Example - 3

```
#include <stdio.h>
#include <conio.h>
int main() {
    int a, b;     double c;
    printf("Enter two values (press spacebar in between): ");
    scanf("%d %d", &a, &b);
    c = a + b;     printf("%d + %d = %0.0f \n", a, b, c);
    c = a - b;     printf("%d - %d = %0.0f \n", a, b, c);
    c = a * b;     printf("%d * %d = %0.0f \n", a, b, c);
    c = (double) a / b;     printf("%d / %d = %4.2f \n", a, b, c);
    getch();
    return 0;
}
```

Formatting Outputs

- Integers: **%nd**
- n: minimum width on the output where value is printed on the right.

```
a=15;
printf("%3d",a);      →    _ 1 5
printf("%5d",a);      →    _ _ _ 1 5
b=5682;
printf("%2d",b);      →    5 6 8 2
printf("%6d",b);      →    _ _ 5 6 8 2
```

Formatting Outputs

- Doubles: **%n.mf**
- n: minimum total width of the output
- m: exact width for the output of part after decimal point

c=15.648;

printf("%9.4f",c); → _ _ 1 5 . 6 4 8 0

printf("%4.2f",c); → 1 5 . 65

Example - 4

```
#include<stdio.h>
#include<conio.h>
main()
{
char me[20];
printf("Please write your name: ");
scanf("%s",&me);
printf("Nice to meet you, %s",me);
getch();
return(0);
}
```

Arithmetic Operators and Precedence

Operation	Arithmetic operator	C programming
Addition	+	a + 5
Substraction	-	b - 14
Multiplication	*	c * d
Division	/	j / i
Modulus	%	x % y

Operator	Precedence (evaluation order)
( )	First
* / %	Second
+ -	Third

Example - 5

- Remember from previous week. We have written an algorithm and drawn a flowchart for celcius to fahrenhayt conversion program.
- Now, write a C code of that program that converts F value, which is given by user, to C value.

Example - 5

```
#include <stdio.h>
#include <conio.h>
main() {
float c,f;                  /* c: celcius, f: fahrenhayt */
printf("Fahrenhayt to celcius conversion program.\n");
printf("Write F value : ");
scanf("%f",&f);
c=(f-32)*5/9;
printf("\nC value is %4.1f",c);
getch();
return(0); }
```

Homework

1. Write a C program that converts centimeters to inches and feet.

2. Write a C program that converts gallons to liters and cubic meters.

Example - 6

- Write a C program that calculates area and circumference of a circle whose radius is given by the user.
- You can define pi by writing this line immediately after header lines:

  `#define PI 3.1415962`
- $A = \pi * r^2$
- $C = 2* \pi * r$

Example - 6

```
#include<stdio.h> #include<conio.h>
#define PI 3.1415962
main()          {
int r;   double a,c;
printf("This program calculates area and circumference of a
circle.");
printf("Radius : ");    scanf("%d";&r);
a=PI*r*r;      c=2*PI*r;
printf("\nArea: %f, Circumference: %f",a,c);
getch();
return(0); }
```

Example - 7

- Write a program that asks user to write radius (r) and height (h) of a cylinder and then calculates:
  - Volume ($V = \pi r^2 h$)
  - Side area ($2\pi rh$)
  - Total area ($2\pi r^2 + 2\pi rh$)