Objectius de la pràctica:
- Edició i compilació de programes en C/C++ en l'entorn Dev-C++.
- Conèixer l'estructura general bàsica d'un programa en C/C++.
- Funcionament de les funcions bàsiques d'entrada i eixida sense format en C++ (açò és, *cin* i *cout*).
- Configuració i treball amb l'entorn de treball Orwell Dev-C++.

Al llarg de l'assignatura treballarem amb el programari Orwell Dev-C++, el qual ofereix un entorn gràfic de desenvolupament que permet editar i compilar programes en C/C++. Orwell Dev-C++ es pot descarregar i instal·lar des de la següent adreça: http://sourceforge.net/projects/orwelldevcpp/ .

1. Executa l'aplicació Dev-C++ i, des del menú "Fitxer", crea un "Nou > Fitxer Font" i escriu el següent programa en C++:

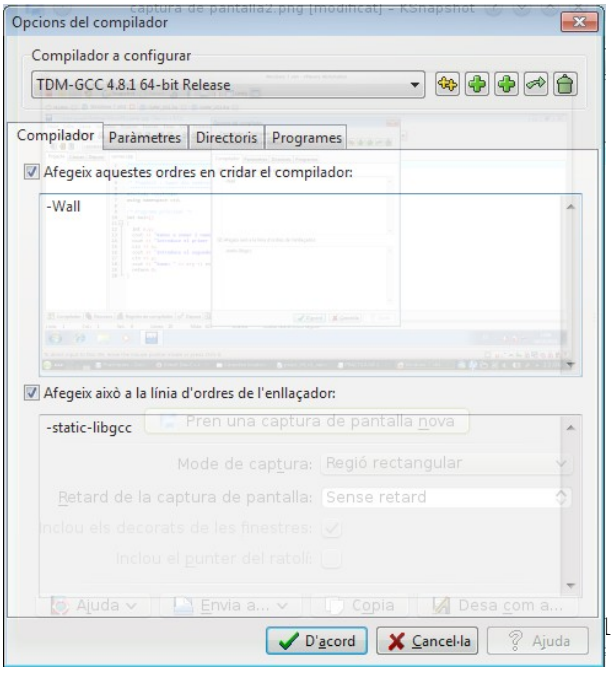
```
/*************************************************************/
/* Programa Sumar                                            */
/* Autor : Nom i Cognoms                                     */
/* Propòsit : Sumar dos nombres                              */
/*************************************************************/
#include <iostream>
using namespace std;
/* Programa principal */
int main()
{
  int x,y;
  cout << "Anem a sumar 2 nombres enters " << endl;
  cout << "Introdueix el primer enter: ";
  cin >> x;
  cout << "Introdueix el segon enter: ";
  cin >> y;
  cout << "Suma= " << x+y << endl;
  return 0;
}
```

Guarda (açò és, "Desa") el fitxer amb el nom "sumar.cpp". Si haguérem editat el programa amb un editor de text extern, hauríem primer de punxar "Fitxer > Obre..." per a carregar-lo i poder treballar amb ell.

2. Realitza les següents tasques:

Compila el programa (des del menú "Executa > Compila"). En acabant, si no hi ha errors, tria l'opció "Executa" del mateix menú "Executa".

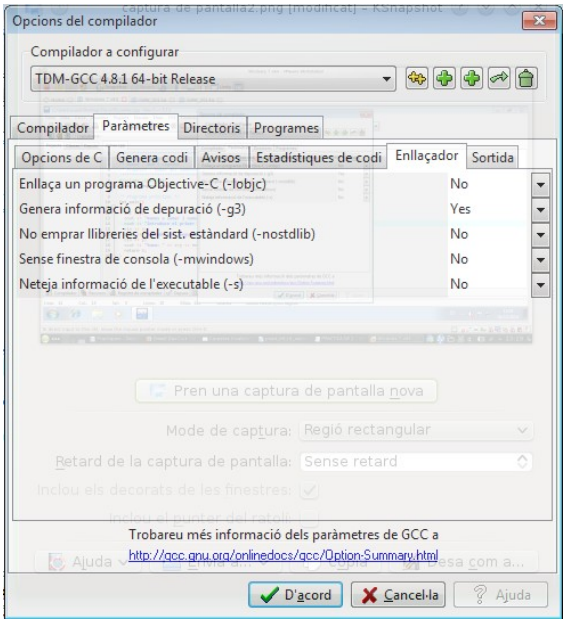
Dins del menú "Eines", obri "Opcions del compilador" i veuràs la següent finestra.



Assegura't que sempre estiga seleccionada l'opció "Afegir aquestes ordres en cridar el compilador" i escriu en el quadre de sota "-Wall".

Aquesta ordre permet veure tots els avisos (warnings) del compilador quan compiles un programa.

En el quadre de diàleg anterior, tria la pestanya "Paràmetres" i, en l'apartat "Enllaçador", assegura't que l'opció "Genera informació de depuració" està activada.



Si has hagut de fer algun dels canvis anteriors, torna a compilar el programa.

Cerca l'ordre "Depura" entre les icones del menú i comprova com s'executaria un programa pas a pas.

Activa un punt d'interrupció en el programa (breakpoint) i comprova el seu funcionament en el mode de depuració.

Tot açò aprofita per a saber perquè el teu programa no fa el que tu esperaves d'ell o bé per a millorar-lo.

3. Des del Dev-C++, escriu el següent programa i desa'l amb el nom "programa2.cpp":

```
/*********************************************************/
/* Programa Programa 2                                   */
/* Autor : Nom i Cognoms                                 */
/* Propòsit : Hola                                       */
/*********************************************************/
#include <iostream>

{
  int x,y;
  cout << "Hola món"
  return 0;
};
```

Quan el compiles comprovaràs que té errors. Fixa't en el programa anterior ("sumar.cpp") per a corregir-los.

Resum

- Edició del codi font del programa.
- Compilació (no genera un fitxer executable sinó un fitxer objecte) i enllaçat (link) del programa amb la biblioteca estàndard de C++ (genera un fitxer executable).
- Execució del programa.

4. L'asptecte del C/C++

Dels exemples anteriors podem ressaltar diverses coses:

- És fàcil d'entendre i d'imaginar què farà un programa quan s'execute.

- Hom pot distingir un conjunt de paraules amb significat especial per al compilador com ara: include, void, main i cout. A aquestes paraules les anomenem "paraules reservades".
- Hi ha un conjunt de símbols estranys com ara: #, /*, */, <, >, {, }, ;, (,).

Els elements més bàsics del C/C++

Dins de cada programa, fins i tot del més senzill, hi ha una sèrie d'elements que estan sempre presents.

Element	**Propòsit**
`#include <iostream>`	Inclou la llibreria estàndard d'entrada/eixida (iostream).
`main`	Nom de la funció principal. La seua presència és obligatòria en qualsevol programa C i l'execució del programa comença sempre des d'ella.
`void`	Tipus de dades general.
`( )`	Parèntesis que acompanyen les funcions. Dins es col·loca un conjunt d'arguments
`/* */`	Símbols de començament i finalització de comentari. Els comentaris són ignorats pel compilador i s'empren per a ajudar a clarificar el codi del programa.
`//`	Símbol de comentari d'una línia en C++. Ens indica que des del punt on apareix fins el final de la línea hi ha un comentari.
`;`	El punt i coma marca el final d'una ordre o sentència.
`{ }`	Les claus agrupen un bloc de sentències que pertanyen a una funció o a una sentència de control.

El C és un llenguatge estructurat

Quan pretenem resoldre un problema, hi ha diverses maneres de fer-ho: ho podem fer d'una forma organitzada o, ans al contrari, de qualsevol manera. Organitzar un programa amb una estructura clara i modular sol tindre molts avantatge. Per exemple, el programa es podrà comprendre millor, serà més fàcil de depurar, de mantenir i d'actualitzar. Un programa no estructurat, o poc estructurat, duu a la situació contrària, açò és: és difícil de comprendre i de modificar.

El llenguatge C és un llenguatge estructurat que es caracteritza per l'ús de blocs de codi. Un bloc no és més que un conjunt de sentències relacionades de forma lògica.

Tipus de blocs

Els tres tipus de blocs bàsics són:

- El bloc seqüencial: És el més senzill i està format per un conjunt de sentències que s'executen una rere altra de forma seqüencial.
- El bloc repetitiu: Està format per un conjunt de sentències que s'executen una o més vegades abans de continuar amb l'execució d'altres blocs.

- El bloc selectiu: Està format per un conjunt de sentències que s'executen sols quan es compleix una condició. Dins del bloc es permet, a més a més, la definició d'un conjunt de sentències alternatiu, les quals s'executaran quan no es complisca la condició anterior.

A sovint agrupem un o més d'aquests blocs per formar **mòduls** independents, anomenats **funcions**, que estan dissenyats per a fer una tasca específica. La funció és la unitat fonamental d'un programa C i, per tant, tot programa en C s'estructura com una col·lecció de funcions.

Qüestió d'estil:

Existeix una infinitat de criteris per mantenir una programació ben estructurada, quasi tants com programadors, però a continuació recomanarem algunes regles que poden ajudar en aquestes sessions de laboratori:

Pel que fa a la capçalera del programa: És adequat que les primeres línies d'un programa es dediquen a un bloc de comentaris on aparega el nom dels programadors, la data de la darrera actualització i un comentari sobre l'objectiu del programa.

Tocant als blocs. Cal deixar una separació d'una o més línies en blanc entre blocs diferents, així com afegir un xicotet comentari al començament de cada bloc que explique quin és el seu objectiu.

També és essencial sagnar, en columnes diferents, les instruccions d'un bloc contingut entre claus, per a fer palès on es troba l'inici i l'acabament del bloc.

Referent a les funcions: Les funcions han d'aparèixer separades unes d'altres mitjançant línies en blanc i, a més a més, abans de la capçalera de la funció cal posar un comentari que explique la tasca que duu a terme, el significat dels paràmetres que rep i el valor que retorna.

Estructura general d'un programa en C: Tot programa s'haurà d'adaptar, més o menys, a la següent estructura simplificada:

```
Includes de funcions de llibreria.
Definicions de Constants.
Declaracions de variables GLOBALS.
Declaració dels prototips de les funcions.
Capçalera de la Funció 1
    Declaració de variables de la Funció 1
    Bloc de sentències de la Funció 1
      .....
      .....
Capçalera de la Funció N
     Declaració de variables de la Funció N
     Bloc de sentències de la Funció N
```

Fixa't com les funcions no es poden definir unes dins d'altres, encara que sí es poden cridar entre ells i fins i tot a sí mateixes.

Convé fer especial esment en què tot programa C conté una funció anomenada “main”, que serà la funció principal encarregada d'iniciar l'execució del programa.

A continuació reproduïm un exemple senzill de programa en C ben estructurat:

```
/* Programa realitzat per Pepet.
Ultima actualització: 1 d'octubre de 2014.
Objectiu: Aquest és un programa d'exemple senzill i il·lustratiu d'un programa
ben estructurar en C. El programa serveix per a calcular l'àrea d'un cercle */

// Bloc d'includes de les llibreries externes
#include <iostream>
#include <stdlib.h>
using namespace std;

/* Bloc de definició de constants */
#define PI 3.141592

// Bloc de declaració de variables globals
float y;

// Bloc de definició de prototips de funcions
float area(float);

/* Ara ve la definició del conjunt de funcions que formen el programa */

/* Funció que calcula l'àrea d'un cercle de radi x */
/* Paràmetres: x - radi del cercle */
/* Retorna: el valor de l'àrea */

float area (float x)                              /* capçalera de la funció */
{
  // Bloc de declaració de variables locals
  float resultat;

  /* Bloc d'instruccions de la funció àrea */
  resultat=PI*x*x;       /* guarda en resultat el producte de Pi per x per x */
  return(resultat);      /* valor retornat per la funció */
}

/* Funció principal */
int main (void)
{
  /* Bloc de declaració de variables locals a la funció main */
  float z;

  /* Bloc seqüencial de presentació */
  cout << "Aquest programa és un exemple senzill";
  cout << "d'un programa ben estructurat";
  cout << endl << "Calcula l'àrea d'un cercle"<< endl;

  /* Bloc repetitiu d'entrada de dades.
     Es repeteix fins que l'usuari introduïsca un valor positiu */
  do
  {
    cout << "Introdueix el radi del cercle " << endl;
    cin >> z;
  } while (z<0);

  /* Bloc seqüencial per a calcular l'àrea i mostrar-la per pantalla */
```

```
  y = area(z);                                    /* Cridada a la funció area */
  cout << "L'àrea del cercle és " << y << endl;

  system("pause");
  return 0;
}
```

5. Introducció a les funcions d'entrada i eixida estàndard de C++

Un programa en C++ pot realitzar operacions d'entrada i eixida de diverses maneres: amb la vella biblioteca de C que proporciona el fitxer de capçalera “stdio.h” (aquest tipus de funcions d'entrada i eixida requereixen el concepte de punter) o amb la nova biblioteca de C++ que requereix incorporar el fitxer “iostream”.

A continuació descrivim els fluxos d'informació. Un flux d'entrada no és més que la sèrie d'entrades que alimenten un ordinador perquè el programa les use i un flux d'eixida és el conjunt d'informació que el programa genera.

En aquesta secció suposarem que les entrades provenen del teclat i que les eixides s'envien a la pantalla d'un terminal.

Eixides amb cout:

`cout << “He comprat dolços”;`	Mostra la cadena escrita entre cometes.
`cout << “He comprat ” << nDolcos << “ dolços;`	Mostra les cadenes amb el valor de lavariable nDolcos intercalat.
`cout << “Hola \n Hola \n”;`	Mostra per pantalla: Hola Hola Fa un bot de línia entre els dos Hola.
`cout << “Hola” << endl << “Hola” << endl;`	Mostra per pantalla: Hola Hola Fa un bot de línia entre els dos Hola.
`cout << “Hola\tHola” << endl;`	Mostra per pantalla: Hola Hola Deixa una tabulació entre els dos Hola.

En els exemples anteriors hem emprat els caràcters '\n' i '\t', que s'anomenen seqüències d'escapament. Les seqüències d'escapament comencen amb el símbol '\' i s'acompanyen d'un caràcter determinat, sense espais que els separe. La seqüència d'escapament '\n' indica a l'ordinador que bote a una nova línia en el dispositiu d'eixida i la seqüència '\t' que bote a la següent tabulació horitzontal.

Recorda com també havíem vist en els exemples anteriors una altra forma d'enviar un bot de línia amb la paraula reservada “**endl**”.

Entrades amb cin:

`cin >> num;`	Llegeix una dada introduïda pel teclat i la guarda en la variable "num".

Resum de les funcions d'entrada/eixida (associades a iostream)

La funció d'eixida bàsica que emprarem serà cout, junt amb l'operador d'inserció <<

cout << "El preu total és: " << (preu1 + preu2);

Dins de la cadena podrem afegir seqüències d'escapament com:

Caràcter	Acció
`\n`	Bot de línia (equivalent a endl)
`\t`	Tabulador
`\\`	Barra invertida
`\"`	Cometes dobles

La funció d'entrada serà cin, amb l'operador d'inserció però en la direcció contrària.

cin >> n_empleats;