C++ Builder

El IDE (Entorno de desarrollo integrado)

C++ Builder es una aplicación Windows que proporciona un entorno de desarrollo visual para construir aplicaciones Windows que integra diferentes aspectos de la programación en un entorno unificado e integrado.

1. Diseñador de formularios

Es una ventana cuadriculada sobre el cual se disponen los componentes para diseñar las ventanas que formarán la aplicación.

2. Editor de código

Un tipico de texto multiventana para ver y editar el código de la aplicación

3. Paleta de componentes

Los componentes constituyen los bloques básicos sobre los que se construyen aplicaciones Windows con C++ Builder basadas en la VCL (Visual Component Library) y son la base de la VCL.

4. Diseñador de formularios

Mediante el uso del ratón podemos colocar componentes de la paleta de componentes en el área de diseño. Lo único que tenemos que hacer es:

1. Elegir el componente en la paleta de componentes
2. Pinchar el componente para seleccionarlo
3. Pinchar en el formulario
4. Automáticamente la prueba se pondrá Anal

4. Inspector de objetos

Con el inspector de objetos podemos editar los componentes de una aplicación [illegible] (propiedades) y funcionalidad (eventos a los que puede responder).

6. Editor de código

Permite editar el código de una aplicación de una forma cómoda. Admite colores [illegible] y distintos archivos abiertos simultáneamente. Se divide en dos: inspector de clases y administrador de proyectos

7. Administrador de proyectos

Es únicamente un organizador de archivos. Tipos de archivo que pueden formar parte de un proyecto/grupo de proyectos: EXT, BPR, CPP, H, OBJ, EXE, TDS, DFM, RES, DSK, BPG, HPP.

Arrays (Vectores) y cadenas

Los Arrays son variables estructuradas, donde cada elemento se almacena de forma consecutiva en memoria.

Las cadenas de caracteres son declaradas en C como arrays de caracteres y permiten la utilización de un cierto número de notaciones y de funciones especiales.

Arrays de una dimensión

El número de componentes "n" es la dimensión del arrays. De igual manera que en matemáticas decimos que "A" es un vector de dimensión "n".

tipo nombre [n] ; } El formato para declarar un array
donde: n >= 1

nombre [i], } Para acceder a un elemento de la array
donde: 0 <= i < n

Por ejemplo. A[4]

```
for (i=0; i<4; i++)
    A[i] = i;
```

Arreglo de dos dimensiones

También se le denomina como matriz y es interpretado como un arreglo de dimensión "f", donde cada componente es un array de dimensión "c".

int nombre [f] [c]...; } Declaración de una arreglo multidimensional
donde [f, c ...] >= 1;

nombre [i] [j]; } Para acceder a un elemento
donde: 0 <= i < f, 0 <= j < c;

Cadena de caracteres

Una cadena en C es un array de caracteres de una dimensión que termina con el carácter especial '\0' (cero)

El formato para declarar una cadena es:

char nombre [n];

donde: n >= 1 y representa la longitud - 1 real de la cadena

Ejemplo

char cadena [5]

En casos específicos se puede utilizar varias formas de inicialización

char cadena [] = "Hola"

char cadena [] = {'H', 'o', 'l', 'a', 0};

Sin especificar el tamaño de la cadena, o especificando el tamaño

char cadena [5] = "Hola"

Las estructuras

Una estructura define un nuevo tipo de datos que está compuesto por una colección de datos de tipo existente. Las estructuras son muy útiles ya que permiten definir tipos de datos que se ajustan con exactitud a las necesidades de representación de cualquier entidad.

La sintaxis de definición de una estructura es:

```
struct nombre {
tipo 1 campo 1;
tipo 2 campo 2;
...
tipo campo;
};
```

Estructuras unidadas

Es aquella que incluye un campo de tipo estructura. Por ejemplo:

```
#include <iostream>
#include <string>
using namespace std;
  struct Persona {
        string nombre;
        int edad;
  };
```

```
int main() {
    Persona p1 = {"Luis", 18};
    Persona p2 = p1;
    p2.edad = 19;
    cout << "Datos:\n";
    cout << "Nombre: " << p2.nombre << endl;
    cout << "Edad: " << p2.edad << endl;
    return 0;
}
```

Fecha / /

double z = 3 * max(a, a - 1);

cout << "z vale " << z << endl;

return 0;

}

Funciones void y sin parámetros

Se suele utilizar el término función void para designar una función que no tiene valor de retorno. C++ obliga a interactuar un tipo de retorno, las funciones void utilizan el tipo especial void como tipo de retorno.

Si una función void no utiliza ninguna sentencia return, la ejecución de la función termina cuando se ejecuta la última instrucción de la función.

Variables globales y locales

La pila

El ámbito de una variable local abarca desde su lugar de definición en la función hasta el final del bloque más interno en que ha sido definida.

Las variables locales de una función se almacenan en la pila. La pila es una zona de memoria contigua donde se almacenan las variables locales de las funciones y alguna otra información, como la dirección de memoria de retorno de una función.

```
#include <iostream>
using namespace std;
double maximo (double x, double y ){
    double m;
    if (x > y)
    m = x;
    return m;
}
int main () {
   double a, b;
   cout << "Introduzca dos numeros: ";
   cin >> a >> b;
   cout << "El maximo es: " << maximo (a, b) << endl;
   cout << "El maximo de " << a << " y 10 es: " << maximo (a, 10) << endl;
```

```
a = 1;
double z = 1 + maximo ( a , a-1);
cout << "z vale: " << z << endl;
return z;
}
```

Variables globales

A una variable definida fuera del cuerpo se le denomina variable global. El ámbito de una variable global abarca desde el lugar de su definición hasta el final del fichero en que se ha definido, pudiendo ser utilizada en cualquier función situada en dicho ámbito

```
#include <iostream>
const int TAM = 4;
float v [TAM] = (5.2f, 5.4f, 3, 2); // vector global
float maximo;
void max () {
    maximo = v [0];
    for (int i = 1; i < TAM; i++)
        if (v[i] > maximo)
}

int main () {
int v = 4; // estas variables ocultan el vector global v
max (); // calcula el maximo del vector y guardalo en maximo
std::cout << "Maximo: " << maximo << " v: " << v << endl;
return 0;
}
```

Tema:

Fecha

Paso de variables

El paso de parametros por value consiste en copiar el contenido de la variable que queremos pasar a otra dentro del ambito local de la subrutina, consiste en copiar el contenido de la memoria del argumento que se quiere pasar a otra direccion de memoria, correspondiente dentro del ambito de dicha subrutina.

```
#include <iostream>
using namespace std;
int porValor (int a)
{
  a= a + 10;
  return a;
}
int main () {
    int f,a;
    a=4;
    f= porValor (a);
    cout << "Variable original" << a << endl;
    cout << "Valor de variable porValor" << f << endl;
    return 0;
}
```

Nº Tema:

Fecha / /

Paso por referencia

El paso de parametros por referencia consiste en proporcionar a la subrutina a la que se le envie como argumento la dirección de memoria del dato. En este caso se tiene un unico valor referenciado desde dos puntos diferentes.

```
#include <iostream>
using namespace std;
void porReferencia (int a, int &b, int &c) {
    b = a + 5;
    c = a + 10;
}
int main() {
int x, y, a;
    a = 4, x = 0, y = 0;
    cout << "Valor de variable porReferencia (x): " << x << endl;
    cout << "Valor de variable por Referencia (Y): " << y << endl;
    porReferencia (a, x, y);
    cout << "Variable Original" << a << endl;
    cout << "Valor de variable por Referencia (x)" << x << endl;
    cout << "Valor de variable por Referencia (Y)" << Y << endl;
    return 0;
}
```