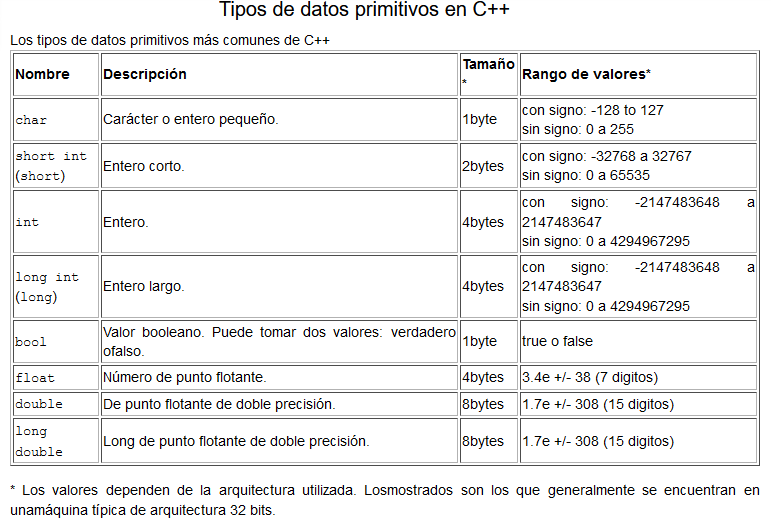
Tipos primitivos de C++ y rangos



Puntero dirección de memoria

Un puntero es una variable que almacena la dirección de memoria de un objeto. Los punteros se usan ampliamente en C y C++ para tres propósitos principales:

* para asignar nuevos objetos
* para pasar funciones a otras funciones

Detalles al crear y usar punteros en C++

* El tipo de dato del apuntador debe coincidir con el de la variable cuya posición en memoria apuntan. En el ejemplo vemos que tanto variable como apuntador son enteros.
* Siempre que queremos usar el apuntador debemos anteponer el asterisco (\*) para indicar que usaremos el valor en la posición de memoria apuntada.
* De no usar el asterisco el comportamiento sería impredecible. Estaremos haciendo uso de la dirección de memoria más no del valor almacenado en ésta.
* Después de usar un puntero, especialmente si trabajamos con arreglos o matrices, es MUY recomendable liberar la memoria utilizada con la función delete (tal como en el ejemplo)
* Un puntero o apuntador puede ser de cualquier tipo de dato, inclusive los podemos usar con tipos complejos.

Referencia

Una referencia de un objeto no es un objeto, en el sentido que no tiene su propio espacio de almacenamiento como ocurre con los punteros, y en consecuencia no pueden realizarse con ellas muchas de las operaciones que se relacionan con objetos.

Link interesante <https://www.genbeta.com/desarrollo/punteros-y-referencias>

Codigo

para recorrer en iteración los elementos de matrices u otras estructuras de datos

Int InsertarInicio(int v);//No retornan nada

void InsertarFinal(int v);

void InsertarPos (int v, int pos);

void EliminarInicio();

void EliminarFinal();

void EliminarPos(int pos);

bool ListaVacia() { return primero == NULL; } //retorna True o False

void Imprimir();

void Borrar(int v);

void Mostrar();

void Siguiente();

void Primero();

void Ultimo();

void BorrarFinal();

void BorrarInicio();

void borrarPosicion(int pos);

void Sumanum(int num);

int largoLista();// retorno un valor numerico

private:

pnodo primero;

};

lista::~lista()

{

pnodo aux;

while(primero) {

aux = primero;

primero = primero->siguiente;

delete aux;

}

actual = NULL;

}

int lista::largoLista(){

int cont=0;

pnodo aux;

aux = primero;

if(ListaVacia()){

return cont;

}else{

while(aux!=NULL){

aux=aux->siguiente;

cont++;

}

return cont;

}

}

Lista Lista;

Lista.InsertarInicio(20); T

Lista.InsertarInicio(2); F

void lista::InsertarInicio(int v) parámetro 20

Lista inicia primero null

{

if (ListaVacia()) T

primero = new nodo(v);

**Lista Vacia**

primero null

20

**Lista NO Vacia**

else

primero=new nodo (v,primero);

}

primero

2

20

null

**Ejemplo de instrucciones aux (apuntador)**

primero->valor=primero->valor+2; correcta

primero->siguiente=primero->siguiente->siguiente; correcta

pnodo aux=primero; correcta

aux=aux->siguiente; correcta

aux=aux->siguiente correcta null

void lista::InsertarFinal(int v)

Lista Lista;

Lista.InsertarFinal(10); T

Lista.InsertarFinal(11); F

{

if (ListaVacia())

primero = new nodo(v);

**Lista Vacia**

primero null

10

else

{ pnodo aux = primero;

while ( aux->siguiente != NULL)

aux= aux->siguiente;//dentro del while

aux->siguiente=new nodo(v);

} aux->valor=2 aux->siguiente=nodo 20

}

**Lista NO Vacia**

aux aux

primero

null null Inserto 10

10

20

2

aux

primero aux aux aux

Inserto 11

20

2

10

Null null

11

void lista::InsertarPos(int v,int pos)

Lista Lista;

Lista.InsertarPos(5,3); T

Lista.InsertarPos(6,4); F

{

if (ListaVacia())

primero = new nodo(v);

**Lista Vacia**

primero null

10

else{

**Pos==1 Equivale a inserter al inicio**

if(pos <=1){

// Primera Opcion

pnodo nuevo = new nodo(v); 5,pos=1

nuevo->siguiente= primero;

primero= nuevo;

primero

5

nuevo null

primero

5

//Segunda Opción

InsertarInicio(5,1) llamar al método InsertarInicio(v)

}

**Pos igual a otra posición**

else{\*\*\*\*\*\*\*\*

pnodo aux= primero; //declara y inicializa

int i =2;

while((i != pos )&&(aux->siguiente!= NULL))

{

i++;

aux=aux->siguiente;

}

pnodo nuevo= new nodo(v);

nuevo->siguiente=aux->siguiente;( 1)

aux->siguiente=nuevo;

}

}}

i=2 3 4 pos=4

aux

primero aux

20

10

2

11

aux

6

nuevo

5

i=2 3 pos=3 aux

primero aux

20

10

2

11

nuevo null

5