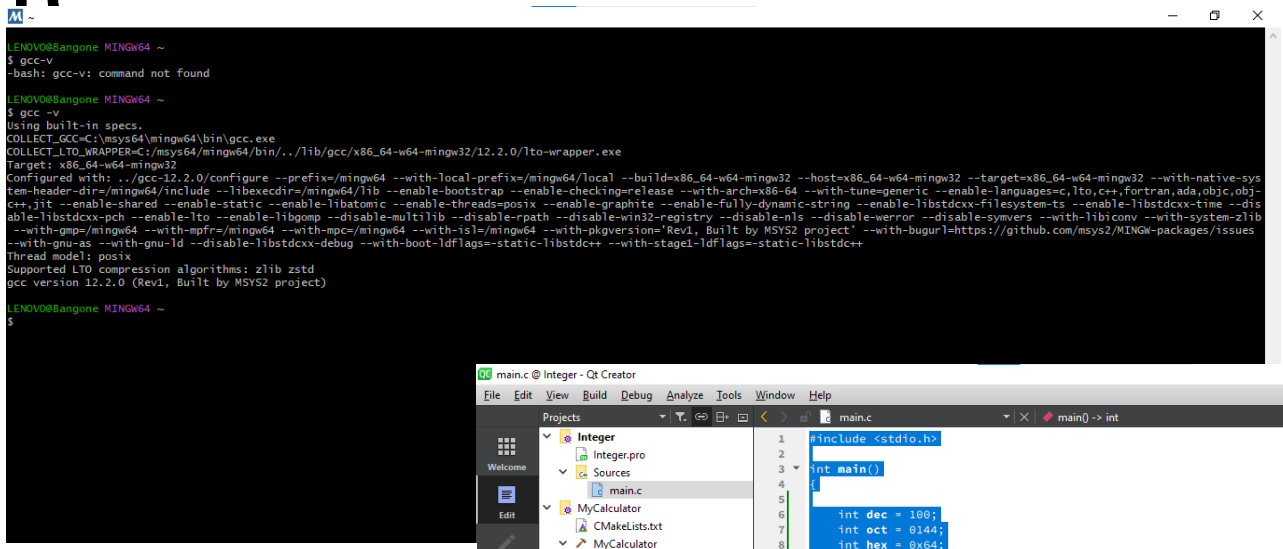
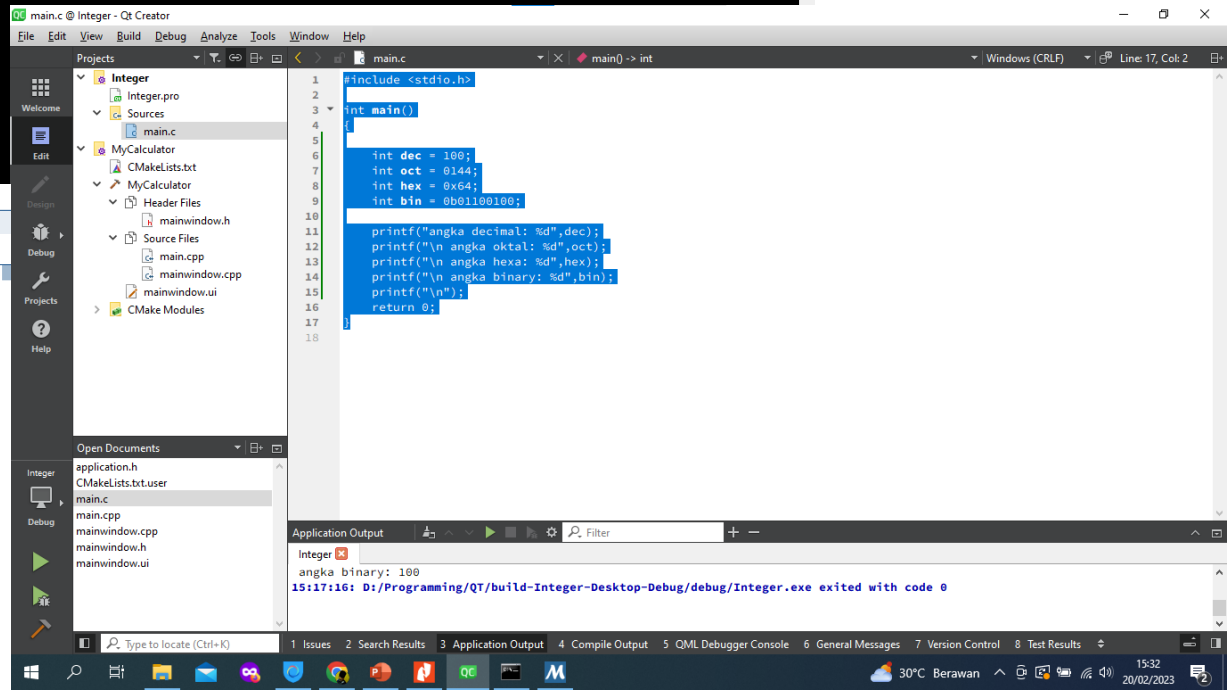
C++ Basic Variable&Collection

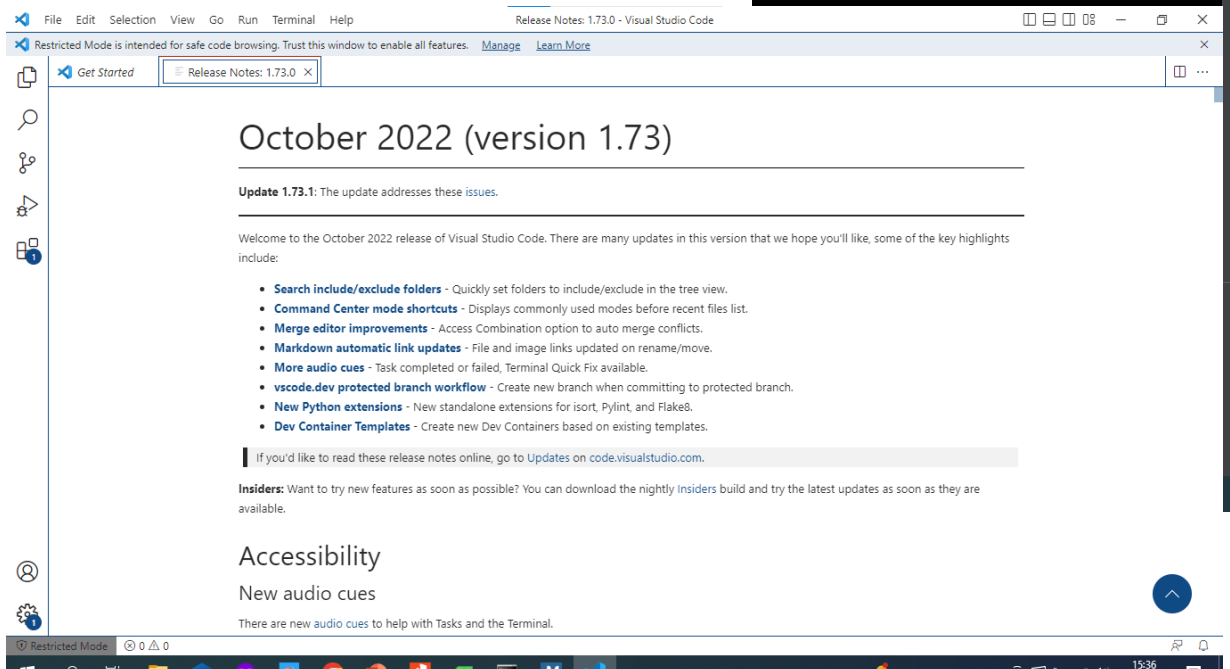
By Hermawan

Requirement

- GCC, Install MYSYS
- QT/Visual Studio







Preprocesors

- #define ZERO 0
- #include “one_file”

  #include “hitung.h”
- #include <iostream>

  include <iostream>

Basic Variable

- Integer literal (C++14)
  - Decimal

  int d = 100;
  - Oktal: 3 Bit Binary…. Max 7

  $decimal = a_{(n-1)} \times 8^{(n-1)} + ... + a_2 \times 8^2 + a_1 \times 8^1 + a_0 \times 8^0$

  int o = 0144=100;
  - Hexa: Heksadesimal 1 Byte/4 Bit Binary…. Max 15/F

  (0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F)

  Dimana A = 10, B = 11, C= 12, D = 13 , E = 14 dan F = 15

  $6*16^0 + 4*16^1 = 100$

  int x = 0x64=100;
  - Binary

  int b = 0b01100100=100;

Basic Variable

```
#include <stdio.h>

int main(){

    int dec = 100;
    int oct = 0144;
    int hex = 0x64;
    int bin = 0b01100100;
    long l = 123456789;
    float f = 100.5;
    double d  = 100000.56789;

printf("angka decimal: %d",dec);
    printf("\n angka oktal: %d",oct);
    printf("\n angka hexa: %d",hex);
    printf("\n angka binary: %d",bin);
    printf("\n angka long Int: %d",l);
    printf("\n angka float: %f",f);
    printf("\n angka double: %f",d);
    printf("\n");
    return 0;
{
```

Basic Variable

- Integer literal (C++14)
  - Decimal

  int d = 100;
  - Oktal: 3 Bit Binary…. Max 7

  $decimal = a_{(n-1)} \times 8^{(n-1)} + ... + a_2 \times 8^2 + a_1 \times 8^1 + a_0 \times 8^0$

  int o = 0144=100;
  - Hexa: Heksadesimal 1 Byte/4 Bit Binary…. Max 15/F

  (0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F)

  Dimana A = 10, B = 11, C= 12, D = 13 , E = 14 dan F = 15

  $6*16^0 + 4*16^1 = 100$

  int x = 0x64=100;
  - Binary

  int b = 0b01100100=100;

Basic Composition
Struct

```
struct Date
{
    unsigned int Year : 13;
        2^13 = 8192, enough for "year" representation for long time
    unsigned int Month: 4;
        2^4 = 16, enough to represent 1-12 month values.
    unsigned int Day: 5;
         2^5 = 32, enough to represent 32 day values.
};
```

Basic Collection

Basic Collection:

1. Static Array

   int Array[const]

2. Dynamic Array *

   int *pArray = new int[const]

3. Pointer of Object, &

   int *pArray[const]=&Object[const]

Pointer

A **pointer** is a variable that **stores** the **memory address** of another variable as its value.

```
int i = 10;
int* p = &i;
```

Pointer p stores memory address of i,

*p is i = 10

& is reference address

* Is dereference value

Pointer

```
printf("Hello World!\n");
    int i = 10;
    int* p = &i;
    int** pp = &p;
    printf("\n%d",p);
    printf("\n0x%p", p);
    printf("\n%d",*p);
    printf("\n%d",&p);
    printf("\n0x%p",&p);
    printf("\n%d",pp);
    printf("\n%d",**pp);
```

Var	Value	Address
A	10	0x0000005564bff86c
*p	1690302572	0x0000005564bff860
**pp	1690302560	0x0000005564bff858

Basic Collection Array

Basic Collection:

```
int const number=x;
    int arr[number]; // allocation stack of integer array
    int *pArr=new int[number];   // ← Allocation heap of integer array
    int *p=&arr[0];  // pointer of array index
    int *pArray[number];
    pArray[0]=&arr[0];

    cout<<"\n"<<p<<"\n"<<&arr[0]<<"\n"<<*(p+1)<<endl;
    cout<<"\n"<<*(pArray[0])<<endl;
```

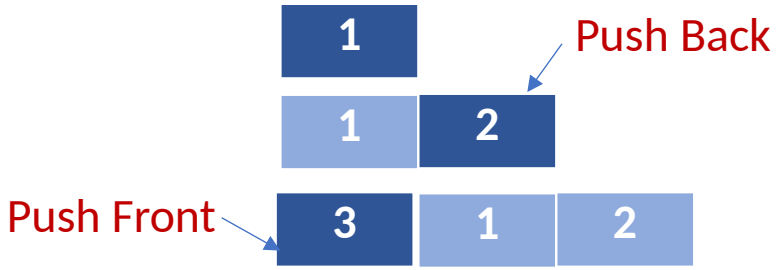
Container/Bag Collection

Bag Collection C++ STL:

1. List
2. Queue
3. Stack
4. Vector
5. Set
6. Multiset
7. Map
8. MultiMap

Container/Bag Collection
1. List

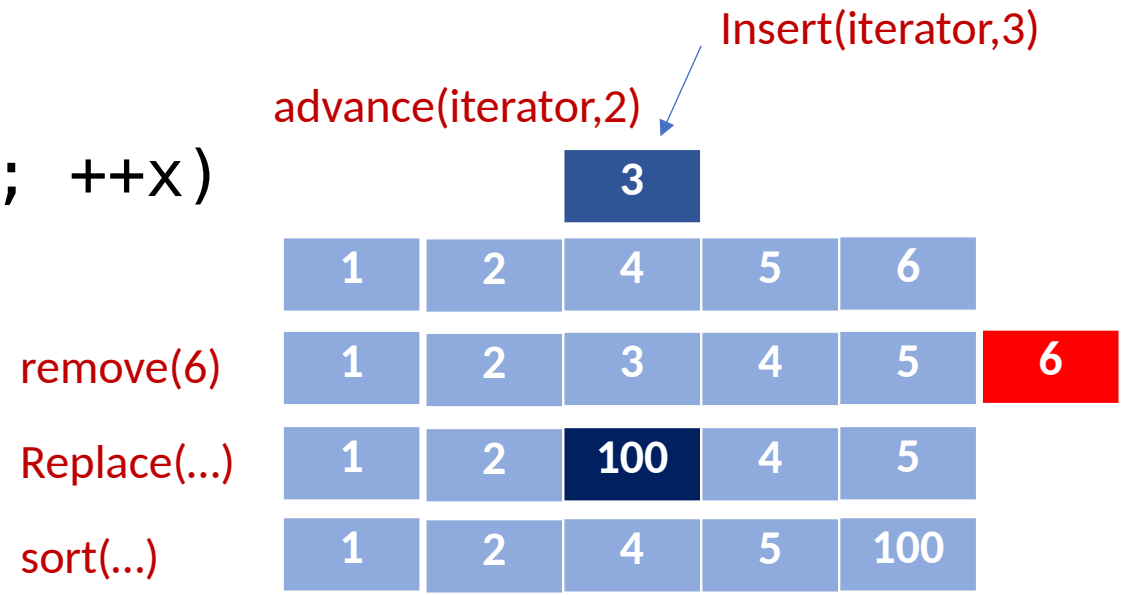
```
list<int> myList;
   myList.push_back(1);
   myList.push_back(2);
   myList.push_front(3);
   list<int>::iterator iterL;
   for(iterL=myList.begin(); iterL!=myList.end();++iterL){
       cout<<*iterL<<endl;
   }
```



Container/Bag Collection
1. List

```
list<int> xList = { 1, 2, 4, 5, 6 };
   cout<< "List: ";
   auto x = xList.begin();
   for( x; x != xList.end( ); ++x)
     cout<< *x << " ";

   list<int>::iterator it;
   it=xList.begin();
   advance(it,2);
   xList.insert( it, 3);
   xList.remove(6);
   replace(xList.begin(),xList.end(),2,100);
   xList.sort();
  cout<< "New list";
```



Container/Bag Collection
2. Queue-Stack

Other items of List,

```
queue.push();
queue.pop();
```

```
vector<int> g1;
```

Container/Bag Collection
3. Vector

Other items of List within vector data,

```
vector<type> vektor;
vektor.push_back(const);
sort(vektor.begin(),vektor.end());
```

Container/Bag Collection
3. Set

Ordered of List,

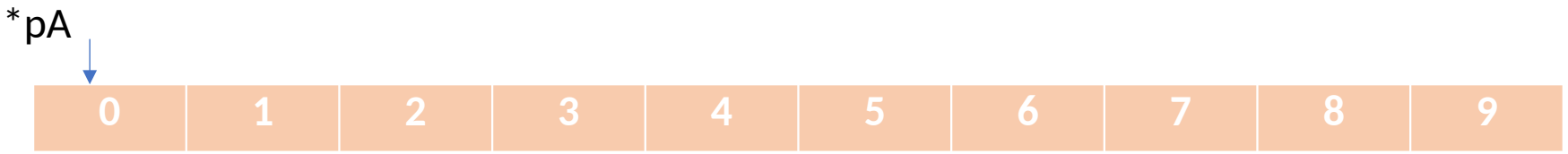
```
Always set ordered and unique data list
```

Container/Bag Collection
4. Map

Pair Set of Key and Value data,

```
map.insert(pair<int, string>(1, “data 1”));
map.insert(pair<int, string>(2, “data 2”));
map[3]=“data3”;
map<int, string>::iterator itr;
   for (itr = map.begin(); itr != map.end(); ++itr) {
    cout << '\t' << itr->first << '\t' << itr->second
         << '\n';
   }
```

Quiz…?



Diketahui Array int A[10], jika *pA adalah pointer pada &A

Tentukan cara Scan *pA sehingga dihasilkan output pA=3,4,5.

2. Jika *pA digunakan untuk men-Copy stack A[10], tentukan alokasi memori heap untuk *pA