

|  |  |
| --- | --- |
| #include <iostream>  #include <math.h>  using namespace std;  int sum1(int n)  {      int S = 0;      for (int i=1; i<=n; i++)      {          S+=i;      }      return S;  } | // Sum of factorials  int sum3(int n)  {      int S = 0;      for (int i=1; i<=n; i++)      {          int tmp = 1;          for (int j = 1; j<=i; j++)          {              tmp \*=j;          }          S+=tmp;      }      return S;  } |
| // Sum of fractions  float sum2(int n)  {      float S = 0;      for (float i=1; i<=n; i++)      {          S+=1/i;      }      return S;  } | // Sum of exponential  float exponential(float x, int n)  {      float t=1;      for(int i=1; i<=n; i++)      {          t\*=x;      }      return t;  } |
| float sum4 (float x, int n)  {      float S = 0;      for (int i=1; i<=n; i++)      {          S+=exponential(x,i);      }      return S;  } | // Sum of square root loop  float sum5 (int n)  {      float S = 0;      for (int i=1; i<=n; i++)      {          S=sqrt(2+S);      }      return S;  } |

|  |  |
| --- | --- |
| // count number of digits in n  int count\_digit(int n)  {      int cnt = 0;      while(n!=0)      {          cnt++;          n=n/10;      }      return cnt;  } | float SUM(int n, float Element(float i))  {      float S = 0;      for (int i=1; i<=n; i++)      {          S+=Element(i);      }      return S;  } |
| // multiplication of digits in n  int multipli\_digits(int n)  {      int multiply = 1;      while(n!=0)      {          multiply \*= n%10;          n=n/10;      }      return multiply;  } | float fraction(float i)  {      return 1/i;  }  float factorial(float i)  {      int f = 1;      for (int j = 1; j<=i; j++)      {          f \*= j;      }      return f;  } |

[For]

|  |
| --- |
| #include <iostream>  #include <math.h>  using namespace std;  #define \_Extream -1  unsigned long Tinh(unsigned long L, unsigned long R, double P, unsigned long thuNhap)  {      unsigned long kq=0;      if(thuNhap<L) return kq;      else{          if(thuNhap<R||R == \_Extream) kq=(thuNhap-L)\*P;          else kq=(R-L)\*P;      }      return kq;  }  unsigned long TinhThue(unsigned long thuNhap)  {      unsigned long L1=4000000, L2=6000000, L3=9000000, L4=14000000, L5 = 24000000, L6=44000000, L7=84000000;      double P1=0, P2=0.05, P3=0.1, P4=0.15, P5=0.2, P6=0.25, P7=0.3, P8=0.35;      unsigned long TienThue = Tinh(0, L1, P1, thuNhap)+                  Tinh(L1, L2, P2, thuNhap)+                  Tinh(L2, L3, P3, thuNhap)+                  Tinh(L3, L4, P4, thuNhap)+                  Tinh(L4, L5, P5, thuNhap)+                  Tinh(L5, L6, P6, thuNhap)+                  Tinh(L6, L7, P7, thuNhap)+                  Tinh(L7, \_Extream, P8, thuNhap);      return TienThue;  }  int main()  {      cout << "Nhap thu nhap: ";      unsigned long thuNhap;      cin >> thuNhap;      cout << "Thue can tra: " << TinhThue(thuNhap);      return 0;  } |

[Array]\_W5\_KTLTr\_06

|  |  |
| --- | --- |
| #include<iostream>  #include<math.h>  using namespace std;  #define N 50  #define MaxRow 20  #define MaxCol 30  void arrIntInput(int a[N], int& n);  void arrIntOutput(int a[N], int& n);  void sumEvenElements(int a[N], int& n);  void multiOddPosition(int a[N], int& n);  void arr2DIntInput(int b[][MaxCol], int& m, int& n);  void arr2DOutput(int a[][MaxCol], int m, int n);  void rotateMatrix(int m, int n, int mat[][MaxCol]); |  |

|  |  |
| --- | --- |
| void arrIntInput(int a[N], int& n)  {      while(1)      {          cout << "Nhap so phan tu can dung: ";          cin>>n;          if(n<0||n>N)              cout<<"Vui long nhap lai \n";          else              break;      }      for(int i = 0; i < n; i++)      {          cout << "Nhap a[" <<i<<"]: ";          cin>>a[i];      }  } | void sumEvenElements(int a[N], int& n)  {      int Sum=0;      for (int i=0; i<n; i++)          if(a[i]%2==0)              Sum+=a[i];      cout << "\nTong cac phan tu chan: " << Sum;  }  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  void multiOddPosition(int a[N], int& n)  {      int res = 1;      for (int i=1; i< n; i++)          if(i%2 != 0)              res\*=a[i];      cout << "\nTich cac phan tu o vi tri le: "<< res;  } |
| void arrIntOutput(int a[N], int& n)  {      for(int i=0; i < n; i++)          cout << a[i] << "\t";  } |

|  |  |
| --- | --- |
| void arr2DIntInput(int b[][MaxCol], int& m, int& n)  {      cout << "\nNhap so dong: ";      cin >> m;      cout << "Nhap so cot: ";      cin >> n;      for (int i=0; i<m;i++)      {          for (int j=0; j<n; j++)          {              cout << "Nhap a[" << i << "][" <<j<<"]: ";              cin >> b[i][j];          }      }  }  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  void arr2DOutput(int a[][MaxCol], int m, int n)  {      for (int i=0; i<m; i++)      {          for (int j=0; j<n; j++)              cout << a[i][j] << "\t";          cout << "\n";      }  } | void rotateMatrix(int row, int col, int mat[][MaxCol])  {      int firstEle = mat[0][0];      for (int i=0; i<col-1; i++)      {          mat[0][i]= mat[0][i+1];      }      for (int i=0; i<row-1; i++)      {          mat[i][col-1] =  mat[i+1][col-1];      }      for (int i=col-1; i>0; i--)      {          mat[row-1][i]=mat[row-1][i-1];      }      for (int i=row-1; i>1; i--)      {          mat[i][0]=mat[i-1][0];      }      mat[1][0]=firstEle;      cout << "Rotate Matrix: \n";      arr2DOutput(mat, row, col);  }  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  int main()  {      cout << "--- BAI TAP 1 ---\n";      int arr[N], m;      arrIntInput(arr, m);      arrIntOutput(arr, m);      sumEvenElements(arr, m);      multiOddPosition(arr, m);      cout << "\n\n--- BAI TAP 2 ---";      int a[MaxRow][MaxCol], row, col;      arr2DIntInput(a, row, col);      arr2DOutput(a, row, col);      rotateMatrix(row, col, a);      return 0;  } |

[RECURSION]\_W6

|  |  |
| --- | --- |
| long numOfEven (long a[], int n);  void printNegPos (long a[], int n);  bool isAscendingArr (long a[], int n);  long numOfEven (long a[], int n)  {      int res;      if (n<=0)          return 0;      if (a[n-1]%2==0)          res = 1;      else          res = 0;      return res + numOfEven(a, n-1);  }  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  void printNegPos (long a[], int n)  {      if (n<=0)          return;      if (a[n-1] < 0)          cout << "\t" << n-1;      printNegPos (a, n-1);  } | bool isAscendingArr (long a[], int n)  {      if (n<2)          return 1;      if (a[n-1] > a[n-2])          return isAscendingArr(a, n-1);      else          return 0;  }  int main()  {      long a[] = {-6, -2, -1, 2, 8, 9};      int n = sizeof(a)/sizeof(a[0]);      cout << "1. Number of Even Elements: " << numOfEven(a, n) << endl;      cout << "2. Position of negative element: ";      printNegPos(a, n);      if (isAscendingArr(a, n)==0)          cout << "\n3. It is NOT an ascending array";      else          cout << "\n3. It is an ascending array";      return 0;  } |

[Pointer Variable]\_W10

|  |  |
| --- | --- |
| #include <stdio.h>  #include <stdlib.h>  void InputArray\_1D(int\*& a, int& n);  void OutputArray\_1D(int\* a, int n);  void FreeArray\_1D(int\*& a);  void InputMatrix(int\*\*& mat, int &n);  void OutputMatrix(int\*\* mat, int n);  void FreeMatrix(int\*\*& mat, int n);  void InputArray\_1D(int\*& a, int& n)  {      scanf("%d", &n);      a = (int\*)malloc(n\*sizeof(int));      if(a==NULL) return;      for (int i=0; i<n; i++)      {          printf("a[%d]: ", i);          scanf("%d", &a[i]);      }  }  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  void OutputArray\_1D(int\* a, int n)  {      for(int i=0; i<n; i++)      {          printf("%d \t", a[i]);      }  }  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  void FreeArray\_1D(int\*& a)  {      free(a);  } | void InputMatrix(int\*\*& mat, int &n)  {      scanf("%d", &n);      mat = (int\*\*)malloc(n\*sizeof(int\*));      if(mat==NULL) return;      for (int i=0; i<n; i++)      {          mat[i] = (int\*)malloc(n\*sizeof(int));          for (int j = 0; j<n; j++)          {              printf("mat[%d][%d]: ", i, j);              scanf("%d", &mat[i][j]);          }      }  }  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  void OutputMatrix(int\*\* mat, int n)  {      for(int i = 0; i<n; i++)      {          for(int j = 0; j<n; j++)          {              printf("%d \t", mat[i][j]);          }          printf("\n");      }  }  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  void FreeMatrix(int\*\*& mat, int n)  {      free(mat);  } |

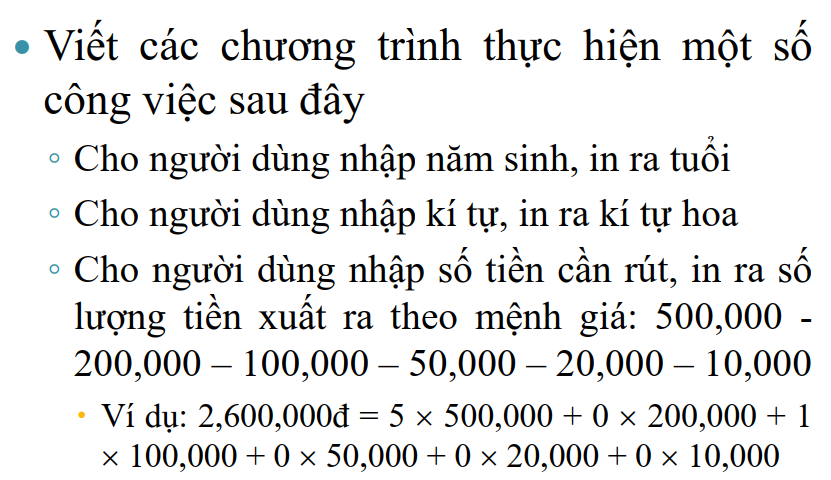
|  |
| --- |
| int main()  {      int n;      int\* a = NULL;      printf("Enter number of array element: ");      InputArray\_1D(a, n);      printf("\nPrint Array: \n");      OutputArray\_1D(a, n);      FreeArray\_1D(a);      // OutputArray\_1D(a, n);      int\*\* mat = NULL;      int m;      printf("\n\nEnter number of matrix element: ");      InputMatrix(mat, m);      printf("\nPrint Matrix: \n");      OutputMatrix(mat, m);      FreeMatrix(mat, m);      // OutputMatrix(mat, m);      return 0;  } |

[Pointer]\_W11

|  |  |
| --- | --- |
| #include <stdio.h>  #include <stdlib.h>  #include <iostream>  #include <cstring>  using namespace std;  int headSize = sizeof(int);  int memSize(int nItem)  {      return headSize + nItem\*sizeof(float);  } | float\* data\_addr(void\* origin)  {      return (float\*)((char\*)origin + headSize);  } |

|  |  |  |  |
| --- | --- | --- | --- |
| void\* origin\_addr(void\* aData)  {      if (aData != NULL)          return (char\*)aData-headSize;      return NULL;  }  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  void set\_nItem(float\* aData, int nItem)  {      \*((int\*)origin\_addr(aData)) = nItem;  }  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  float\* floatArrResize(float\* aData, int nItem)  {      int sz = memSize(nItem);      float\* anew = NULL;      void\* originAddr = NULL;      if(aData != NULL)          originAddr = origin\_addr(aData);      anew = (float\*) realloc(originAddr, sz);      if(anew != NULL)      {          if(aData == NULL)          {              memset(anew, 0, sz);          }          aData = data\_addr(anew);          set\_nItem(aData, nItem);      }      return aData;  }  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  int get\_nItem(float\* aData)  {      return \*((int\*)origin\_addr(aData));  } | | int floatArrSize(float\* aData)  {      if(aData != NULL)      {          return get\_nItem(aData);      }      return 0;  }  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  int floatArrPushback(float\*\* aData, float x)  {      int n = floatArrSize(\*aData);      float\* anew = floatArrResize(\*aData, n+1);      if(anew != NULL)      {          anew[n] = x;          \*aData = anew;          return 1;      }      return 0;  }  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  float\* floatArrInput()  {      float\* a = NULL, x=0;      while (cin>>x)      {          floatArrPushback(&a, x);      }      cin.clear();      return a;  }  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  void floatArrOutput(float\* arr)  {      int n = floatArrSize(arr);      for (int i=0; i<n; i++)      {          cout << arr[i] << "\t";      }  } | |
| void floatArrFree(void\* aData)  {      if(aData != NULL)      {          free(origin\_addr(aData));      }  } | int main()  {      float\* B = NULL;      B = floatArrInput();      cout << "\nOutput: ";      floatArrOutput(B);      floatArrFree(B);      return 0;  } | |

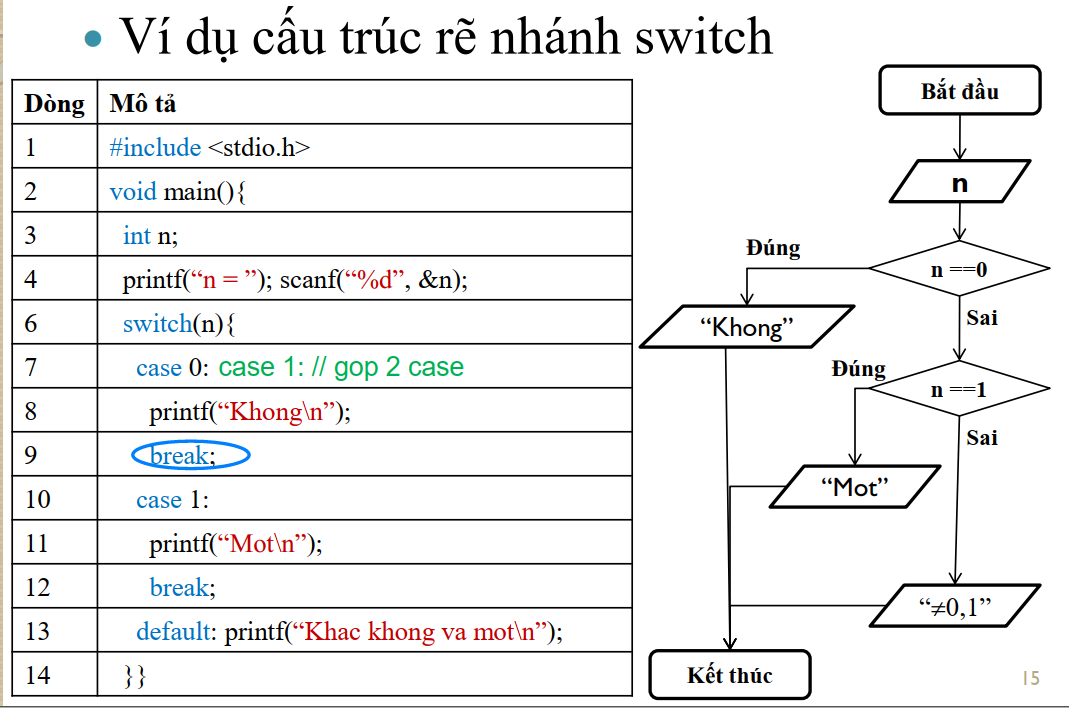
[Lab1]\_String

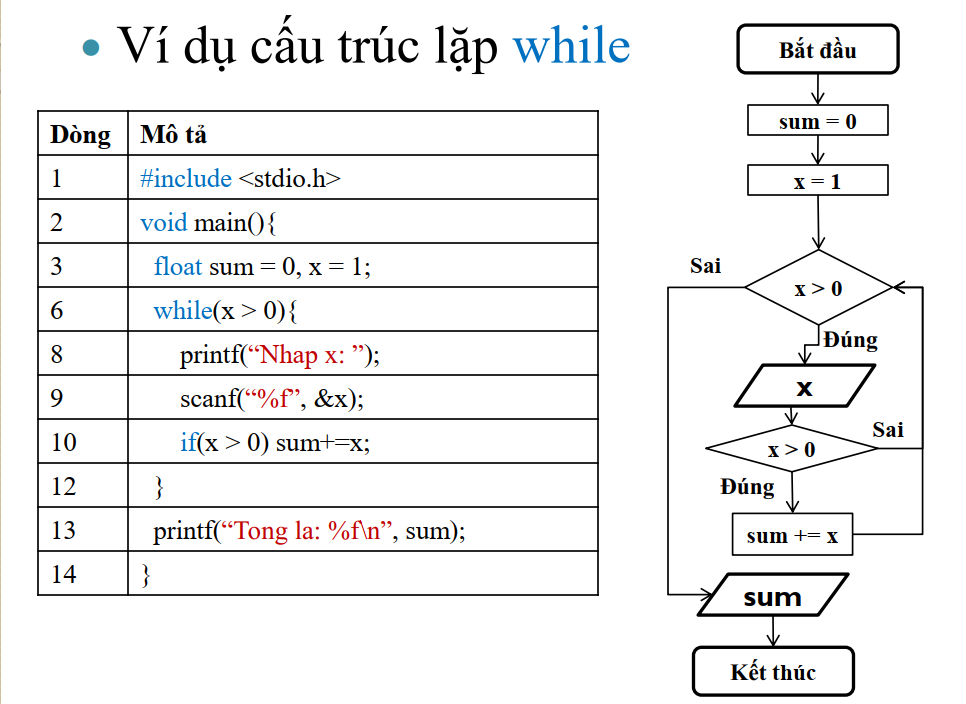


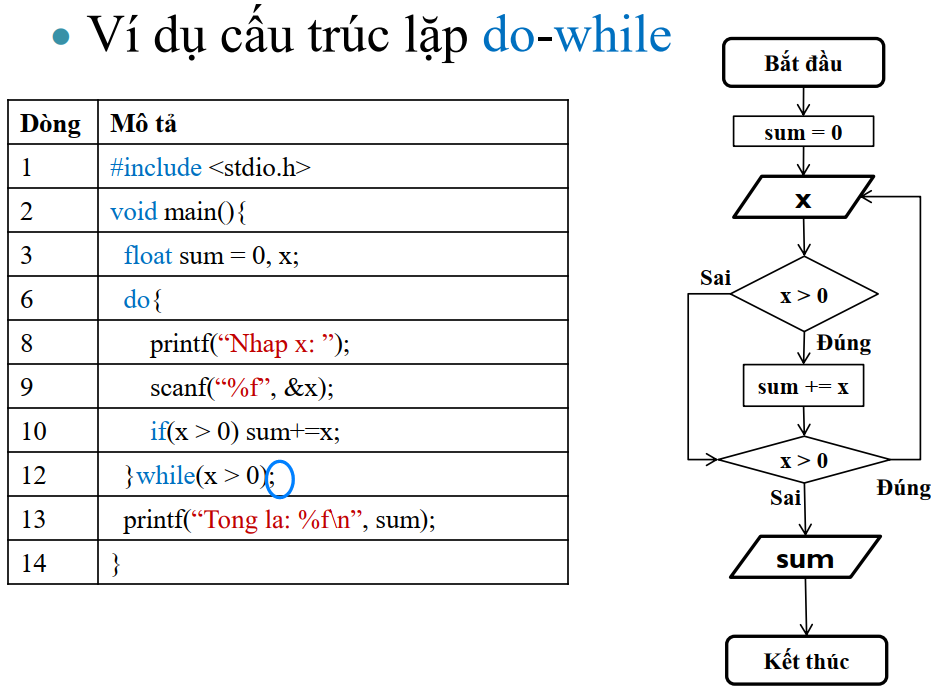
|  |  |
| --- | --- |
| #include <iostream>  #include <ctime>  #include<chrono>  #include<string>  #include<vector>  #include<sstream>  using namespace std;  int calculateAge(int yearborn)  {      auto curTime = chrono::system\_clock::now();      time\_t currentTime = chrono::system\_clock::to\_time\_t (curTime);      stringstream ss(ctime(&currentTime));      string tmp;      vector<string> element;      while(getline(ss,tmp,' '))      {          element.push\_back(tmp);      }      int res = stoi(element[4]) - yearborn;      return res;  } | char uppercase (char c)  {      if ('a' <= c && c <= 'z')      {          c = c - ('a'-'A');      }      return c;  }    \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  vector <int> withdraw (int m, vector<int> denomination)  {      vector <int> res;      int vecSz = denomination.size();      for (int i = 0; i < vecSz; i++)      {          int tmp = (int) (m/denomination[i]);          res.push\_back(tmp);          m -= denomination[i]\*tmp;      }      return res;  } |

|  |
| --- |
| int main()  {      // --- Task 1: Input Year Born => Output: Age      cout << "1. Input the year that you was borned: ";      int n;      cin >> n;      cout << "=> Your age is: " << calculateAge(n) << endl;      // --- Task 2: Input Charater => Output: Uppercase      cout << "2. Input a charater: ";      char c;      cin >> c;      cout << "=> Uppercase: " << uppercase (c) << endl;      // --- Task 3: Input Amount Withdraw => Output: Quantity for each denomination      cout <<"3. How much you want withdraw from ATM: ";      int m;      cin >> m;      vector <int> denomination {500000, 200000,100000, 50000, 20000, 10000};      vector <int> res = withdraw(m, denomination);      int resSz = res.size();      cout << "=> " << m << " = ";      for (int i = 0; i < resSz; i++)      {          if (i != resSz - 1)              cout << denomination[i] << " x " <<res[i] << " + ";          else              cout << denomination[i] << " x " <<res[i] << endl;      }      return 0;  } |

**[SLIDE]**

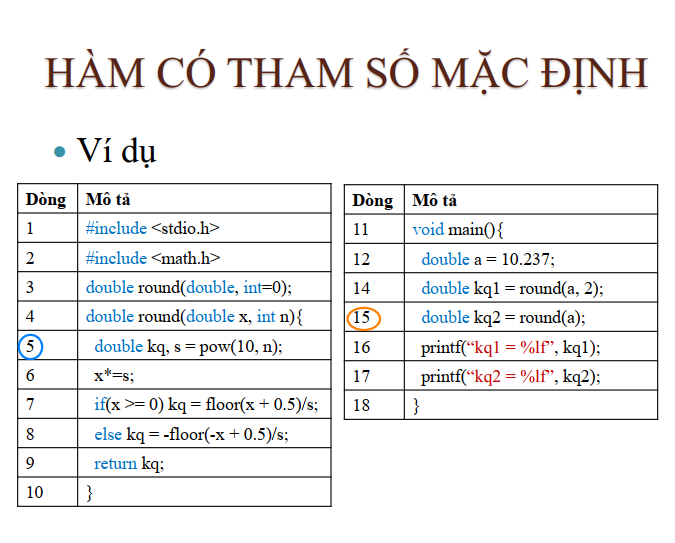
****

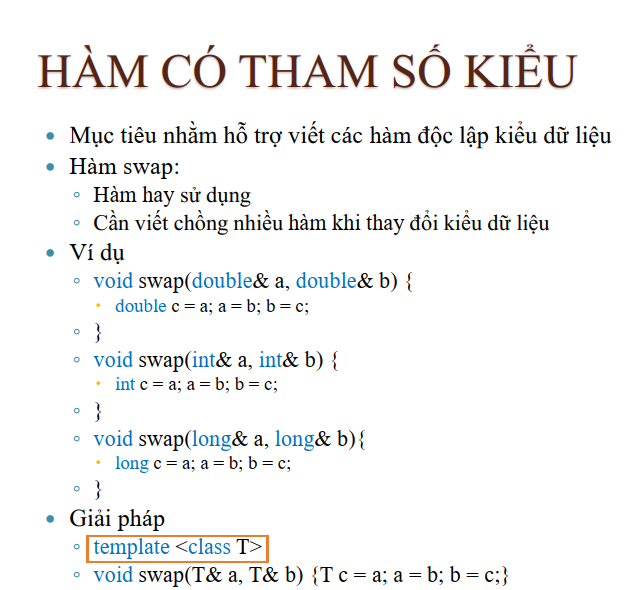
****

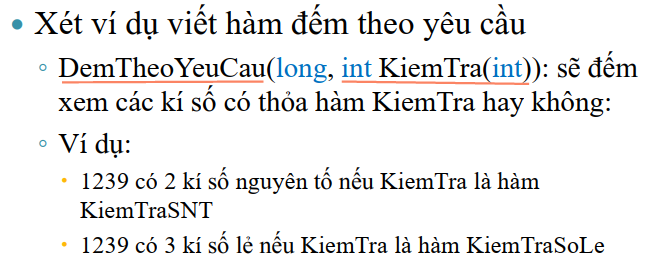
****

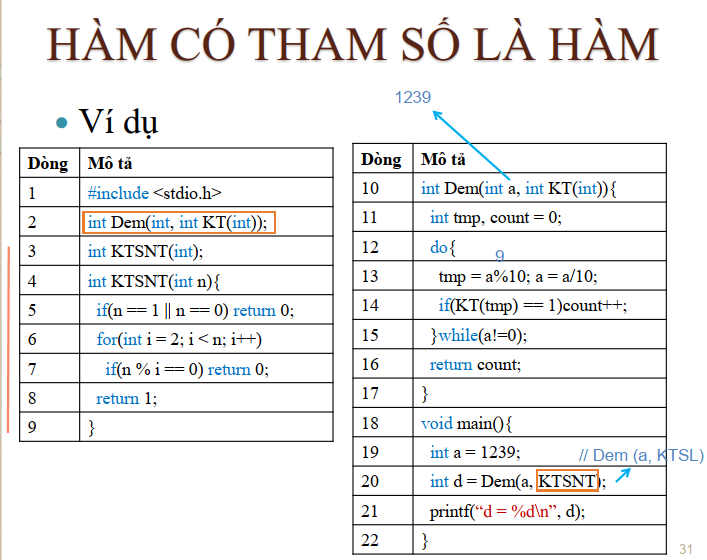
**<math.h>**

|  |  |
| --- | --- |
|  |  |

****

****

****

****