Vector Graphic

**Due Time:** 23.59, Jan 28 2017 **Earnings:** 4% of your final grade

***NOTE: The code in this assignment must be your own work. It must not be code taken from another student or written for you by someone else, even if you give a reference to the person you got it from (attribution); if it is not entirely your own work it will be treated as plagiarism and given a fail mark, or less.***

**Purpose:** You are to write the code in Visual Studio 2013 for a simple C/C++ language console application that holds the data of a Vector Graphics application (there is no actual graphics in the assignment) using dynamic memory allocation for its data. This will give you an opportunity to review material that has already been taught in earlier courses and get up to speed in C programming that will be needed for C++. ***You will be shown how to set up a Visual Studio project in the lab.***

Note that the assignment code from the CLib example from “Thinking in C++” (the course book) uses C/C++ code like cout and new and delete – but in the code you write you use only C code like you have been taught in previous courses. Part of the code is shown on the next page; it is also on the Web Site in three text files that you can copy and paste. You **MUST** use this code **without modification (not a single character changed): no code added or removed, no macros, no defines and no statics**. Your task is to implement (after the main() function), using C, only the functions that are declared in the ass0.cpp file and not add any new ones. All your code is in the file ass0.cpp.

In this assignment, when the application is running the user can

* Add a new Graphic Element (together with its lines) to the Vector Graphic
* Print all the details of the Graphic Elements in the Vector Graphic

**An example of the output of the running application is given at the end. Yours must work identically and produce identical output.**

Note the following:

* The file you submit must be named ass0.cpp,
* the code from the book uses C++ new and delete for dynamic memory management,
* You can only use functions like strlen() and strcpy() or similar etc. from the standard C library to handle strings (you cannot use the C++ string class – that will be used in later assignments),
* When the application terminates it releases **all** dynamically allocated memory so it does not have a resource leak (or you lose 30%).

See the Marking Sheet for how you can lose marks, but you will lose at least 60% if:

1. you change the supplied code in any way at all (not a single character) - no code added (except header includes) or removed, no macros, no defines, no statics and no additional functions,

2. it fails to build in Visual Studio 2013,

3. It crashes in normal operation (such as printing from an empty list etc.),

4. it doesn’t produce the example output.

Part of the code is shown on the next page. You MUST use this code **without modification.** Your task is to add the implementation of the functions that are declared using the style of the posted Submission Standard. All the code is in a single file named ass0.cpp.

**What to Submit :** Use Blackboard to submit this assignment as a zip file (**not** RAR) containing only the single source code file (ass0.cpp). ***If you are taking both CST8219 and CST8233, you need only submit to CST8219***. The name of the zipped folder **must** contain your name as a prefix so that I can identify it, for example for CST8219 using my name the file would be tyleraAss0CST8219.zip. It is also vital that you include the Cover Information (as specified in the Submission Standard) as a file header in your source file so the file can be identified as yours. Use comment lines in the file to include the header. ***Before you submit the code, check that it builds and executes in Visual Studio 2013 as you expect - if it doesn’t build for me, for whatever reason, you get a deduction of at least 60%. Also make sure you have submitted the correct file – if I cannot build it because the file is wrong or missing from the zip, even if it’s an honest mistake, you get 0.***

There is a late penalty of 25% per day. Don’t send me the file as an email attachment – it will get 0.

***Example code (also in a text file on the web site). Don’t change it.***

Vector Graphic is a Stash of Graphic Elements, each of which is a Stash of Lines between Points:

|  |
| --- |
| //: C04:CLib.h  // Header file for a C-like library  // An array-like entity created at runtime  typedef struct CStashTag {  int size; // Size of each space  int quantity; // Number of storage spaces  int next; // Next empty space  unsigned char\* storage;// Dynamically allocated array of bytes  } CStash;  void initialize(CStash\* s, int size);  void cleanup(CStash\* s);  int add(CStash\* s, const void\* element);  void\* fetch(CStash\* s, int index);  int count(CStash\* s);  void inflate(CStash\* s, int increase);  ///:~ |

|  |  |
| --- | --- |
| // ass0.cpp  #define \_CRT\_SECURE\_NO\_WARNINGS  #define \_CRTDBG\_MAP\_ALLOC // need this to get the line identification  //\_CrtSetDbgFlag(\_CRTDBG\_ALLOC\_MEM\_DF|\_CRTDBG\_LEAK\_CHECK\_DF); // in main, after local declarations  //NB must be in debug build  #include <crtdbg.h>  #include <stdio.h>  #include "CLib.h"  enum{ RUNNING = 1 };    struct Point  {  int x, y;  };  struct Line  {  Point start;  Point end;  };  struct GraphicElement  {  enum{ SIZE = 256 };  char name[SIZE];  CStash Lines; // a Stash of Lines  };  struct VectorGraphic  {  CStash Elements; // a Stash of GraphicElements  };  void AddGraphicElement(VectorGraphic\*);  void ReportVectorGraphic(VectorGraphic\*);  void CleanUpVectorGraphic(VectorGraphic\*);  VectorGraphic Image;    int main()  {  char response;  \_CrtSetDbgFlag(\_CRTDBG\_ALLOC\_MEM\_DF | \_CRTDBG\_LEAK\_CHECK\_DF);  // it's a Stash of GraphicElements initialize(&(Image.Elements),sizeof(GraphicElement));  while (RUNNING)  {  printf("\nPlease select an option:\n");  printf("1. Add a Graphic Element\n");  printf("2. List the Graphic Elements\n");  printf("q. Quit\n");  printf("CHOICE: ");  fflush(stdin);  scanf("%c", &response);  switch (response)  {  case '1':AddGraphicElement(&Image); break;  case '2':ReportVectorGraphic(&Image); break;  case 'q':CleanUpVectorGraphic(&Image); return 0;  default:printf("Please enter a valid option\n");  }  printf("\n");  }  return 0;  } | //: C04:CLib.cpp {O}  // Implementation of example C-like library  // Declare structure and functions:  #include "CLib.h"  #include <iostream>  #include <cassert>  using namespace std;  // Quantity of elements to add  // when increasing storage:  const int increment = 100;  void initialize(CStash\* s, int sz)  {  s->size = sz;  s->quantity = 0;  s->storage = nullptr;  s->next = 0;  }  int add(CStash\* s, const void\* element)  {  if(s->next >= s->quantity) //Enough space left?  inflate(s, increment);  // Copy element into storage,  // starting at next empty space:  int startBytes = s->next \* s->size;  unsigned char\* e = (unsigned char\*)element;  for(int i = 0; i < s->size; i++)  s->storage[startBytes + i] = e[i];  s->next++;  return(s->next - 1); // Index number  }  void\* fetch(CStash\* s, int index)  {  // Check index boundaries:  assert(0 <= index);  if(index >= s->next)  return 0; // To indicate the end  // Produce pointer to desired element:  return &(s->storage[index \* s->size]);  }  int count(CStash\* s)  {  return s->next; // Elements in CStash  }  void inflate(CStash\* s, int increase)  {  assert(increase > 0);  int newQuantity = s->quantity + increase;  int newBytes = newQuantity \* s->size;  int oldBytes = s->quantity \* s->size;  unsigned char\* b = new unsigned char[newBytes];  for(int i = 0; i < oldBytes; i++)  b[i] = s->storage[i]; // Copy old to new  delete [](s->storage); // Old storage  s->storage = b; // Point to new memory  s->quantity = newQuantity;  }  void cleanup(CStash\* s)  {  if(s->storage != 0)  {  cout << "freeing storage" << endl;  delete []s->storage;  }  } ///:~ |

Example Output:

Please select an option:

1. Add a Graphic Element

2. List the Graphic Elements

q. Quit

CHOICE: 1

Adding A Graphic Element

Please enter the name of the new GraphicElement(<256 characters): mouth

How many lines are there in the new GraphicElement? 1

Please enter the x coord of the start point of line index 0: -2

Please enter the y coord of the start point of line index 0: -2

Please enter the x coord of the end point of line index 0: 2

Please enter the y coord of the end point of line index 0: -2

Please select an option:

1. Add a Graphic Element

2. List the Graphic Elements

q. Quit

CHOICE: 1

Adding A Graphic Element

Please enter the name of the new GraphicElement(<256 characters): nose

How many lines are there in the new GraphicElement? 4

Please enter the x coord of the start point of line index 0: -1

Please enter the y coord of the start point of line index 0: 1

Please enter the x coord of the end point of line index 0: -1

Please enter the y coord of the end point of line index 0: 2

Please enter the x coord of the start point of line index 1: -1

Please enter the y coord of the start point of line index 1: 2

Please enter the x coord of the end point of line index 1: 1

Please enter the y coord of the end point of line index 1: 2

Please enter the x coord of the start point of line index 2: 1

Please enter the y coord of the start point of line index 2: 2

Please enter the x coord of the end point of line index 2: 1

Please enter the y coord of the end point of line index 2: 1

Please enter the x coord of the start point of line index 3: 1

Please enter the y coord of the start point of line index 3: 1

Please enter the x coord of the end point of line index 3: -1

Please enter the y coord of the end point of line index 3: 1

Please select an option:

1. Add a Graphic Element

2. List the Graphic Elements

q. Quit

CHOICE: 2

VectorGraphic Report

Reporting Graphic Element #0

Graphic Element name: mouth

Line #0 start x: -2

Line #0 start y: -2

Line #0 end x: 2

Line #0 end y: -2

Reporting Graphic Element #1

Graphic Element name: nose

Line #0 start x: -1

Line #0 start y: 1

Line #0 end x: -1

Line #0 end y: 2

Line #1 start x: -1

Line #1 start y: 2

Line #1 end x: 1

Line #1 end y: 2

Line #2 start x: 1

Line #2 start y: 2

Line #2 end x: 1

Line #2 end y: 1

Line #3 start x: 1

Line #3 start y: 1

Line #3 end x: -1

Line #3 end y: 1

Please select an option:

1. Add a Graphic Element

2. List the Graphic Elements

q. Quit

CHOICE: