

File Zipper Software

Table of Contents

Introduction.....	2
Background.....	2
Implementation & Testing	3
Problems and Challenges	14
Conclusion and Breakdown	15

Introduction

Our main motive was to recreate the popular application “WinRAR” which is heavily used by people on a day to day basis.

Team Members:

1. Mohammad Yehya Hayati, K213309
2. Sufyan Abdul Rasheed, K213206

Background

The main research done was on the various different data structures and how to use them in our program. The data structures used in this project were:

- Nodes
- Linked Lists
- Stacks
- MinHeap/Priority Queue
- Huffman Trees

Their uses and functionality is further discussed in the later sections. However a brief summary is given as follows:

Nodes were used for the implementation of Linked Lists and Stacks.

Linked Lists were used for storing an undetermined amount of characters/pixels.

Stacks were used as a buffer to move nodes from a Linked List to a MinHeap/Priority Queue.

MinHeaps were used to store nodes with characters/pixels with a priority set on their frequency. This was also further used as a sorting class which was implemented by the functionality of HeapSort ().

Huffman Trees were used to generate a unique prefix code for every individual character/pixel, which was then used as a replacement for their ASCII codes.

The functionality of all the above mentioned points are combined into 2 functions; zipfile () and unzipfile ().

In addition to this, classes for the storing of data were created. For example, class CharInfo was created to store character information and class BMPHeader was created to store header files of images.

Here is an overview of all the different classes and data structures.

```
1  #include <iostream>
2  #include <fstream>
3  using namespace std;
4  #pragma pack(2)
5  class BMPHeader
6  {
15 class DIBHeader
16 {
32 class CharInfo
33 {
62 template<typename T>
63 class Node
64 {
80 template<typename T>
81 class LinkedList
82 {
145 template<typename T>
146 class MinHeap
147 {
232 template<typename T>
233 class HuffmanTree
234 {
409 void ZipFile()//Main Zip Function
410 {
498 void UnzipFile()//Main Unzip Function
499 {
551 void PrintFile()//Print Contents of File
552 {
576 int main()
577 {
```

Implementation & Testing

(For the further inspection, see attached code)

First and foremost, we will look at the node class. This was needed as it acts as the baseline for all the data structures we will need to use. We made it a generic class to ensure usability of the same class for different file types. For us, we have supported 2 file types which are .txt files and 24-bit .bmp files. Here is our implementation of it.

```
62  template<typename T>
63  class Node
64  {
65      public:
66          T data;
67          Node<T> *left, *right;
68      public:
69          //Constructors
70          Node(){left = right = NULL;}
71          Node(T data){left = right = NULL; this->data = data;}
72          //Destructors
73          ~Node(){left = right = NULL;}
74          //Overloaded Operators
75          bool operator > (Node<T> &x) {return data > x.data;}
76          bool operator < (Node<T> &x) {return data < x.data;}
77          void operator = (Node<T> *x) {*data = x->data;}
78          //Member Functions
79  };
```

Next we will take a look at the class that holds all the data. This is the CharInfo class. It contains four different and equally important details about any unique character. Its character, frequency, binary code, and the length of binary code. At the time of instantiation only the character and frequency are used and the binary code and length are in play later (during encoding).

However we faced a hurdle during this process, and solved it which resulted in the auxiliary space of our program to decrease from 16 bytes to 16 bits! More on this will be discussed later under the “Problems and Challenges” heading. Furthermore, using binary literals which were introduced in C++14, introduced elements of assembly code which we learnt in COAL and made accessing variables faster for our compiler.

Another thing to note is that this class can also be used for raster/bitmap images as they are also written in a similar format, which is 3 characters represent a pixel.

Here is our implementation of it.

```

32 class CharInfo
33 {
34     private:
35         int freq;
36         char c;
37         unsigned short int BinaryCode;
38         int BinaryLength;
39     public:
40         //Constructors
41         CharInfo(){freq = 0; c = '\0'; BinaryCode = 0; BinaryLength = 0;}
42         CharInfo(int f, char c){this->freq = f; this->c = c; BinaryCode = 0; BinaryLength = 0;}
43         CharInfo(int f, char c, unsigned short int bin, int len){this->freq = f; this->c = c; BinaryCode = bin; BinaryLength = len;}
44         //Setters
45         void setBinaryCode(unsigned short int val){BinaryCode = val;}
46         void setBinaryLength(int val){BinaryLength = val;}
47         void setChar(char c){this->c = c;}
48         //Getters
49         bool getBinaryCode(int index){return (BinaryCode & (1<<index))>>index;}
50         int getBinaryLength(){return BinaryLength;}
51         char getChar(){return c;}
52         int getFreq(){return freq;}
53         //Overloaded Operators
54         bool operator > (CharInfo &x) {return freq > x.freq;}
55         bool operator < (CharInfo &x) {return freq < x.freq;}
56         void operator ++ () {++freq;}
57         void operator = (CharInfo &x) {this->freq = x.freq; this->c = x.c; this->BinaryLength = x.BinaryLength; BinaryCode = x.BinaryCode;}
58         CharInfo operator + (CharInfo &x) {return CharInfo(this->freq + x.freq, '5');}
59         bool operator == (char c) {return this->c == c;}
60         //Destructor
61     };

```

Next we will discuss a bit on image files (specifically 24-bit bitmap files). Bitmap files contain a file header which stores important data of any image. For example, it contains the type of the image file or the offset of the image. It takes a total of 14 bytes of the starting of a bmp file. However, this is a problem in itself which will be discussed later in the “Problems and Challenges” heading.

Bitmap files contain another header that is called the DIB (Device Independent Bitmap) header contains the information of the image. For example, the width*height, bits per pixel, and even color depth. This is important since we need to know the width of an image to bypass a very famous problem, of compiler self-allocation, which will be discussed later in the “Problems and Challenges” heading.

Here is our implementation of it.

```

4  #pragma pack(2)
5  class BMPHeader
6  {
7      private:
8          char HeaderField[2];
9          unsigned int size;
10         unsigned int garbage;
11         unsigned int imageOffset;
12     public:
13         unsigned int getImageOffset(){return imageOffset;}
14 };
15 class DIBHeader
16 {
17     private:
18         unsigned int hsize;
19         int width;
20         int height;
21         unsigned short int colorplanes;
22         unsigned short int bpp;
23         unsigned int compression;
24         unsigned int size;
25         unsigned int hor_res;
26         unsigned int ver_res;
27         unsigned int colorDepth;
28         unsigned int impColor;
29     public:
30         unsigned int getWidth(){return width;}
31 };

```

Then we needed a way to store all the characters from a file, but we don't exactly know how many characters there are. Therefore we needed to implement a Linked List class.

This Linked List class was made to fit the needs of a Linked List and as well as a Stack. Thus it has a particular set of functions as seen below.

```

80  template<typename T>
81  class LinkedList
82  {
83      Node<T> *head;
84      public:
85          //Constructors
86          LinkedList():head(NULL){}
87          LinkedList(T d){head = new Node<T>(d);}
88          //Destructor
89          ~LinkedList(){Deletelist();}
90          //Member Functions
91          void Deletelist()
92          {
93              //
94          }
95          void Insert(T d)
96          {
97              //
98          }
99          Node<T>* ReturnHead(){return head;}
100         void InsertChar(char ch)
101         {
102             //
103         }
104         Node<T>* Pop()
105         {
106             //
107         }
108         int SizeofList()
109         {
110             //
111         }
112     };

```

Then we needed a tree-like structure; one that can also semi-sort the nodes that we have created in the Linked List. Thus we used a MinHeap. For the sake of keeping the space complexity of our program at a minimum, we have used the shallow copy technique with removing the pointers from the Linked List to make sure there are no memory leaks.

We have also created a function that will be quite helpful when discussing the next data structure.

```

217  Node<T>* CreateHuffmanTree()
218  {
219      Node<T> *left, *right, *root;
220      while(top != 1)
221      {
222          left = ExtractRoot();
223          right = ExtractRoot();
224          root = new Node<T>(left->data+right->data);
225          root->left = left;
226          root->right = right;
227          Insert(root);
228      }
229      return ExtractRoot();
230  }

```

This small piece of code extracts the two smallest values of the MinHeap (using the logic of heap sort) and creates a new node that is the parent of

these two nodes, and then pushes the node back into the queue. This process repeats itself until there is one node left which will be the root node of our Huffman Tree.

Here is our implementation of a MinHeap.

```
145 template<typename T>
146 class MinHeap
147 {
148     Node<T> **arr;
149     int capacity;
150     int top;
151     int parent(int i) {return (i-1)/2;}
152     int left(int i) {return 2*i + 1;}
153     int right(int i) {return 2*i + 2;}
154     public:
155         //Constructors
156         MinHeap(){arr = NULL;}
157         MinHeap(int n)
158         {
159             //Destructor
160             ~MinHeap()
161             {
162                 //Member Functions
163                 int Capacity(){return capacity;}
164                 void Insert(Node<T> *d)
165                 {
166                     void InsertList(LinkedList<T>& List) {for(int i = 0 ; i < capacity ; i++) Insert(List.Pop());}
167                     Node<T>* ExtractRoot()
168                     {
169                         void MinHeapify(int i)
170                         {
171                             Node<T>* CreateHuffmanTree()
172                             {
173                                 ;
174                             }
175                         }
176                     }
177                 }
178             }
179         }
180     };
```

Next we will talk about the main class of our program. The Huffman tree is a very interesting tree in which the main data only exists at the leaves of the tree. This ensures that every relevant node has a unique code of traversal from the root of the tree. This can be used to our advantage as we can substitute these values as the new ASCII values of our characters.

To further explain, this is done by a simple process. Encode, Compress, Decode, and Decompress.

Encode basically generates a prefix binary code that is set to a leaf node along with the length of the code. This is done extremely quickly due to the use of bitwise operators working with binary literals as explained in the CharInfo section.

Compress is the function that reads a character and then stores its equivalent binary code into a 1 byte buffer. Then, once the buffer is full, it will write

the contents of the buffer into a new file which will be named the compressed file.

Decode is the function to read the header of a compressed file and recreate a Huffman tree to the T. This ensures that every character will be decoded with the exact same details as it was before the compression.

Decompression is the function to read the contents of a compressed file into a 1 byte buffer and traversing the tree until you reach a leaf node. The character in that leaf node is then written into a new file which will be named the decompressed file and should be equal to the original file.

Here is our implementation of it.

```
233 class HuffmanTree
234 {
235     Node<T>* root;
236 public:
237     // Constructors
238     HuffmanTree(){root = NULL;}
239     HuffmanTree(Node<T>* n){root = n;}
240     // Destructor
241     ~HuffmanTree(){DeleteHuffmanTree();}
242     // Member Functions
243     DeleteHuffmanTree(){DeleteHuffmanTree(root);root=NULL;}
244     void DeleteHuffmanTree(Node<T>* r)
245     {
246     }
247     // Encoding & Decoding Functions
248     void EncodeHuffmanCodes(fstream& file2 , LinkedList<T>& L1, bool NOWRITE = 0, int SKIP = 0)
249     {
250     }
251     void EncodeHuffmanCodes(Node<T>* r, unsigned short int &t, int index, fstream& File2, LinkedList<T>& L, bool NOWRITE = 0)
252     {
253     }
254     void CompressFile(fstream& File1, fstream& File2, LinkedList<T>& L, int Width = 0, int SKIP = 0)
255     {
256     }
257     // Reading and Decompression Functions
258     void DecodeHuffmanCodes(fstream& file2)
259     {
260     }
261     Node<T>* DecodeHuffmanCodes(fstream& file2, Node<T>* r, int size)
262     {
263     }
264     void DecompressFile(fstream& file2, fstream& file3, int Width = 0, int SKIP = 0)
265     {
266     }
267 };
```

To sum up all that was said, a simple function named ZipFile can be created to execute these concepts.

Here is the implementation of it.

```

409 void ZipFile()//Main Zip Function
410 {
411     LinkedList<CharInfo> List;
412     fstream ReadFile, WriteFile;
413     string FileName = "";
414     char c;
415
416     cout << "Enter Name of the File to Zip: ";
417     cin >> FileName;
418
419     //Opening File To ZIP
420     ReadFile.open(FileName.c_str() , ios::in | ios::binary);
421     if(!ReadFile.is_open()) {cout << "No Such File Exists!";return;}
422
423     if(!FileName.compare(FileName.find('.'),4,".txt"))
424     {
449     else if(!FileName.compare(FileName.find('.'),4,".bmp"))
450     {
500     else cout << "Exception Thrown! Incorrect or Unsupported file type entered!";
501     ReadFile.close();
502 }

```

Similarly, to unzip a compressed file, we can create a new function named UnzipFile.

Here is the implementation of it.

```

503 void UnzipFile()//Main Unzip Function
504 {
505     fstream ReadFile, WriteFile;
506     string FileName = "";
507
508     cout << "Enter Name of the File to Unzip: ";
509     cin >> FileName;
510
511     //Opening Compressed File
512     ReadFile.open(FileName.c_str() , ios::in | ios::binary);
513     if(!ReadFile.is_open()) {cout << "No Such File Exists!";return;}
514
515     if(!FileName.compare(FileName.find('.'),4,".txt"))
516     {
530     else if(!FileName.compare(FileName.find('.'),4,".bmp"))
531     {
558     ReadFile.close();
559     cout << "File Successfully Unzipped with Name: " << FileName << "!\n";
560 }

```

Here is the main function.

```
586 int main()
587 {
588     int choice;
589     while(true)
590     {
591         cout << "\t\tFAST NUCES TXT File Zipper\n" << "1.Zip File\n2.Unzip File\n3.Print File\n4.Exit\nChoice: ";
592         cin >> choice;
593         if(choice == 1) ZipFile();
594         else if (choice == 2) UnzipFile();
595         else if (choice == 3) PrintFile();
596         else if (choice == 4) break;
597         cin.get();
598         cin.get();
599         system("CLS");
600     }
601     cout << "\nMade By Mohammad Yehya Hayati (K213309), Sufyan Abdul Rasheed (K213206)";
602 }
```

Here are some examples.

1. Original File:

Lorum ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Cursum mattis molestie a iaculis at erat pellentesque adipiscing. Massa sapien faucibus et molestie ac feugiat sed. Convallis convallis tellus id interdum velit laoreet id. Adipiscing elit pellentesque habitant morbi tristique senectus et. Et malesuada fames ac turpis egestas sed tempus. Sed cras ornare arcu dui. Ac placerat vestibulum lectus mauris ultrices eros in cursus. Odio morbi quis commodo odio aenean sed. Nibh sit amet commodo nulla facilisi nullam vehicula ipsum. Blandit cursus risus at ultrices. Vel pretium lectus quam id leo in vitae turpis massa. Viverra ipsum nunc aliquet bibendum enim facilisis gravida neque convallis. Nullam non nisi est sit. Hac habitasse platea dictumst vestibulum. Risus pretium quam vulputate dignissim suspendisse in est ante. Nulla porttitor massa id neque aliquam. Purus gravida quis blandit turpis cursus in hac. Aliquam eleifend nisl in nulla posuere sollicitudin aliquam ultrices sagittis. Sit amet nulla facilisi morbi tempus laculis urna id volutpat. Volutpat ac tincidunt vitae semper. Quis imperdiet massa tincidunt nunc pulvinar sapien et ligula ullamcorper. Non tellus orci ac auctor augue mauris. Nisi scelerisque eu ultrices vitae. Sit amet nisl purus in mollis nunc sed id. Egestas sed tempus urna et pharetra. Auctor neque vitae tempus quam pellentesque nec nam aliquam. Augue eget arcu dictum varius duis at consectetur lorem donec. Eget nullam non nisi est sit amet facilisis magna. Odio ut enim blandit volutpat. Pretium viverra suspendisse potenti nullam ac tortor. At risus viverra adipiscing et. Bibendum est ultricies integer quis auctor. Eleifend donec pretium vulputate sapien. Cursus metus aliquam eleifend nisl in nulla posuere. Eu tincidunt tortor aliquam nulla facilisi cras fermentum odio. Feugiat nisl pretium fusce id velit ut tortor pretium viverra. Posuere ac ut consequat semper viverra. Ipsum dolor sit amet consectetur adipiscing elit ut aliquam. Et tortor et risus viverra adipiscing et in tellus. Morbi leo urna molestie at. Non curabitur gravida arcu ac tortor dignissim convallis aenean. Eget velit aliquet sagittis id consectetur purus. Porttitor lacus luctus accumsan tortor posuere ac ut. Magna ac placerat vestibulum lectus mauris ultrices eros in cursus. Habitasse platea dictumst quisque sagittis purus. Turpis egestas sed tempus urna et pharetra pharetra massa. Donec adipiscing tristique risus nec feugiat in. Non curabitur gravida arcu ac. Quam nulla porttitor massa id neque aliquam. Urna nec tincidunt praesent semper feugiat. Sem viverra aliquet eget sit amet tellus cras adipiscing enim. Facilisis sed odio morbi quis commodo. Imperdiet nulla malesuada pellentesque elit eget gravida. Ipsum a arcu cursus vitae congue mauris rhoncus aenean vel. Et malesuada fames ac turpis egestas integer eget aliquet nibh. Amet justo donec enim diam vulputate ut. Faucibus pulvinar elementum integer enim neque. Sem et tortor consequat id porta nibh venenatis cras sed. Nulla pellentesque dignissim enim sit amet venenatis urna. Volutpat maecenas volutpat blandit aliquam etiam erat velit scelerisque. Sodales neque sodales ut etiam sit amet. Sagittis orci a scelerisque purus semper eget duis at. Amet tellus cras adipiscing enim eu. Odio euismod lacina et quis risus sed. Nec dui nunc mattis enim ut tellus elementum sagittis. Lacus sed turpis tincidunt id aliquet. Vehicula ipsum a arcu cursus vitae congue mauris rhoncus aenean. Id ornare arcu odio ut sem nulla pharetra diam. Elit ut aliquam purus sit amet luctus venenatis. Ut enim blandit volutpat maecenas volutpat blandit aliquam etiam. Ullamcorper sit amet risus nullam eget. Cursus in hac habitasse platea dictumst quisque sagittis. Nullam vehicula ipsum a arcu. Aliquam sen et tortor consequat id porta nibh venenatis cras. Nisi rhoncus mattis rhoncus urna neque viverra justo nec ultrices. Amet purus gravida quis blandit turpis cursus. Sit amet justo donec enim diam vulputate ut. Facilisis gravida neque convallis a. Vel facilisis volutpat est velit egestas dui id. Habitasse platea dictumst quisque sagittis purus sit amet. Hac habitasse platea dictumst quisque. Est ullamcorper eget nulla facilisi etiam dignissim diam. Enim nulla aliquet porttitor lacus luctus accumsan tortor posuere. Quam id leo in vitae. Risus pretium quam vulputate dignissim suspendisse in. Dignissim suspendisse in est ante in nibh mauris cursus. Sit amet dictum sit amet justo donec enim. Turpis cursus in hac habitasse platea. Dui sapien eget nisl proin sed libero enim. Varius vel pharetra vel turpis. Vitae tortor condimentum lacinia quis vel. Maecenas accumsan lacus vel facilisis volutpat est velit egestas dui. Egestas sed sed risus pretium quam vulputate dignissim suspendisse in. Tempus laculis urna id volutpat lacus laoreet non curabitur gravida. At urna condimentum mattis pellentesque. At lectus urna duis convallis convallis tellus id. Mattis vulputate enim nulla aliquet porttitor lacus luctus accumsan tortor. Facilisis magna etiam tempor orci. Feignilla urna porttitor rhoncus dolor purus.

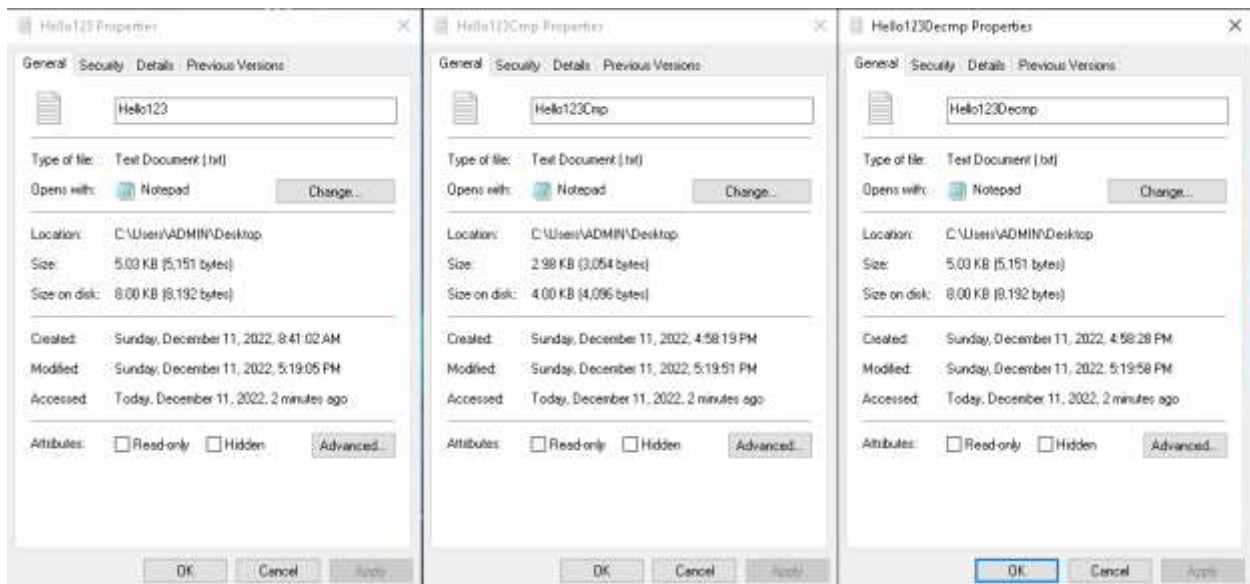
Compressed File:

12345678910111213141516171819202122232425262728293031323334353637383940414243444546474849505152535455565758596061626364656667686970717273747576777879808182838485868788899091929394959697989910010110210310410510610710810911011111211311411511611711811912012112212312412512612712812913013113213313413513613713813914014114214314414514614714814915015115215315415515615715815916016116216316416516616716816917017117217317417517617717817918018118218318418518618718818919019119219319419519619719819920020120220320420520620720820921021121221321421521621721821922022122222322422522622722822923023123223323423523623723823924024124224324424524624724824925025125225325425525625725825926026126226326426526626726826927027127227327427527627727827928028128228328428528628728828929029129229329429529629729829930030130230330430530630730830931031131231331431531631731831932032132232332432532632732832933033133233333433533633733833934034134234334434534634734834935035135235335435535635735835936036136236336436536636736836937037137237337437537637737837938038138238338438538638738838939039139239339439539639739839940040140240340440540640740840941041141241341441541641741841942042142242342442542642742842943430431432433434435436437438439440441442443444445446447448449450451452453454455456457458459460461462463464465466467468469470471472473474475476477478479480481482483484485486487488489490491492493494495496497498499500501502503504505506507508509510511512513514515516517518519520521522523524525526527528529530531532533534535536537538539540541542543544545546547548549550551552553554555556557558559560561562563564565566567568569570571572573574575576577578579580581582583584585586587588589590591592593594595596597598599600601602603604605606607608609610611612613614615616617618619620621622623624625626627628629630631632633634635636637638639640641642643644645646647648649650651652653654655656657658659660661662663664665666667668669670671672673674675676677678679680681682683684685686687688689690691692693694695696697698699700701702703704705706707708709710711712713714715716717718719720721722723724725726727728729730731732733734735736737738739740741742743744745746747748749750751752753754755756757758759760761762763764765766767768769770771772773774775776777778779780781782783784785786787788789790791792793794795796797798799800801802803804805806807808809810811812813814815816817818819820821822823824825826827828829830831832833834835836837838839840841842843844845846847848849850851852853854855856857858859860861862863864865866867868869870871872873874875876877878879880881882883884885886887888889890891892893894895896897898899900901902903904905906907908909910911912913914915916917918919920921922923924925926927928929930931932933934935936937938939940941942943944945946947948949950951952953954955956957958959960961962963964965966967968969970971972973974975976977978979980981982983984985986987988989990991992993994995996997998999100010011002100310041005100610071008100910101011101210131014101510161017101810191020102110221023102410251026102710281029103010311032103310341035103610371038103910401041104210431044104510461047104810491050105110521053105410551056105710581059106010611062106310641065106610671068106910701071107210731074107510761077107810791080108110821083108410851086108710881089109010911092109310941095109610971098109911001101110211031104110511061107110811091110111111211131114111511161117111811191120112111221123112411251126112711281129113011311132113311341135113611371138113911401141114211431144114511461147114811491150115111521153115411551156115711581159116011611162116311641165116611671168116911701171117211731174117511761177117811791180118111821183118411851186118711881189119011911192119311941195119611971198119920020120220320420520620720820921021121221321421521621721821922022122222322422522622722822923023123223323423523623723823924024124224324424524624724824925025125225325425525625725825926026126226326426526626726826927027127227327427527627727827928028128228328428528628728828929029129229329429529629729829930030130230330430530630730830931031131231331431531631731831932032132232332432532632732832933033133233333433533633733833934034134234334434534634734834935035135235335435535635735835936036136236336436536636736836937037137237337437537637737837938038138238338438538638738838939039139239339439539639739839940040140240340440540640740840941041141241341441541641741841942042142242342442542642742842943430431432433434435436437438439440441442443444445446447448449450451452453454455456457458459460461462463464465466467468469470471472473474475476477478479480481482483484485486487488489490491492493494495496497498499500501502503504505506507508509510511512513514515516517518519520521522523524525526527528529530531532533534535536537538539540541542543544545546547548549550551552553554555556557558559560561562563564565566567568569570571572573574575576577578579580581582583584585586587588589590591592593594595596597598599600601602603604605606607608609610611612613614615616617618619620621622623624625626627628629630631632633634635636637638639640641642643644645646647648649650651652653654655656657658659660661662663664665666667668669670671672673674675676677678679680681682683684685686687688689690691692693694695696697698699700701702703704705706707708709710711712713714715716717718719720721722723724725726727728729730731732733734735736737738739740741742743744745746747748749750751752753754755756757758759760761762763764765766767768769770771772773774775776777778779780781782783784785786787788789790791792793794795796797798799800801802803804805806807808809810811812813814815816817818819820821822823824825826827828829830831832833834835836837838839840841842843844845846847848849850851852853854855856857858859860861862863864865866867868869870871872873874875876877878879880881882883884885886887888889890891892893894895896897898899900901902903904905906907908909910911912913914915916917918919920921922923924925926927928929930931932933934935936937938939940941942943944945946947948949950951952953954955956957958959960961962963964965966967968969970971972973974975976977978979980981982983984985986987988989990991992993994995996997998999100010011002100310041005100610071008100910101011101210131014101510161017101810191020102110221023102410251026102710281029103010311032103310341035103610371038103910401041104210431044104510461047104810491050105110521053105410551056105710581059106010611062106310641065106610671068106910701071107210731074107510761077107810791080108110821083108410851086108710881089109010911092109310941095109610971098109911001101110211031104110511061107110811091110111111211131114111511161117111811191120112111221123112411251126112711281129113011311132113311341135113611371138113911401141114211431144114511461147114811491150115111521153115411551156115711581159116011611162116311641165116611671168116911701171117211731174117511761177117811791180118111821183118411851186118711881189119011911192119311941195119611971198119920020120220320420520620720820921021121221321421521621721821922022122222322422522622722822923023123223323423523623723823924024124224324424524624724824925025125225325425525625725825926026126226326426526626726826927027127227327427527627727827928028128228328428528628728828929029129229329429529629729829930030130230330430530630730830931031131231331431531631731831932032132232332432532632732832933033133233333433533633733833934034134234334434534634734834935035135235335435535635735835936036136236336436536636736836937037137237337437537637737837938038138238338438538638738838939039139239339439539639739839940040140240340440540640740840941041141241341441541641741841942042142242342442542642742842943430431432433434435436437438439440441442443444445446447448449450451452453454455456457458459460461462463464465466467468469470471472473474475476477478479480481482483484485486487488489490491492493494495496497498499500501502503504505506507508509510511512513514515516517518519520521522523524525526527528529530531532533534535536537538539540541542543544545546547548549550551552553554555556557558559560561562563564565566567568569570571572573574575576577578579580581582583584585586587588589590591592593594595596597598599600601602603604605606607608609610611612613614615616617618619620621622623624625626627628629630631632633634635636637638639640641642643644645646647648649650651652653654655656657658659660661662663664665666667668669670671672673674675676677678679680681682683684685686687688689690691692693694695696697698699700701702703704705706707708709710711712713714715716717718719720721722723724725726727728729730731732733734735736737738739740741742743744745746747748749750751752753754755756757758759760761762763764765766767768769770771772773774775776777778779780781782783784785786787788789790791792793794795796797798799800801802803804805806807808809810811812813814815816817818819820821822823824825826827828829830831832833834835836837838839840841842843844845846847848849850851852853854855856857858859860861862863864865866867868869870871872873874875876877878879880881882883884885886887888889890891892893894895896897898899900901902903904905906907908909910911912913914915916917918919920921922923924925926927928929930931932933934935936937938939940941942943944945946947948949950951952953954955956957958959960961962963964965966967968969970971972973974975976977978979980981982983984985986987988989990991992993994995996997998999100010011002100310041005100610071008100910101011101210131014101510161017101810191020102110221023102410251026102710281029103010311032103310341035103610371038103910401041104210431044104510461047104810491050105110521053105410551056105710581059106010611062106310641065106610671068106910701071107210731074107510761077107810791080108110821083108410851086108710881089109010911092109310941095109610971098109911001101110211031104110511061107110811091110111111211131114111511161117111811191120112111221123112411251126112711281129113011311132113311341135113611371138113911401141114211431144114511461147114811491150115111521153115411551156115711581159116011611162116311641165116611671168116911701171117211731174117511761177117811791180118111821183118411851186118711881189119011911192119311941195119611971198119920020120220320420520620720820921021121221321421521621721821922022122222322422522622722822923023123223323423523623723823924024124224324424524624724824925025125225325425525625725825926026126226326426526626726826927027127227327427527627727827928028128228328428528628728828929029129229329429529629729829930030130230330430530630730830931031131231

Decompressed File:

lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Cursus mattis molestia a iaculis at erat pellentesque adipiscing. Massa sapien faucibus et molestia ac feugiat sed. Convallis convallis tellus id interdum velit laoreet id. Adipiscing elit pellentesque habitant morbi tristique senectus et. Et malesuada fames ac turpis egestas sed tempus. Sed cras ornare arcu dui. Ac placerat vestibulum lectus mauris ultrices eros in cursus. Odio morbi quis commodo odio aenean sed. Nibh sit amet commodo nulla facilisi nullam vehicula ipsum. Blandit cursus risus at ultrices. Vel pretium lectus quam id leo in vitae turpis massa. Viverra ipsum nunc aliquet bibendum enim facilisis gravida neque convallis. Nullam non nisi est sit. Hac habitasse platea dictumst vestibulum. Risus pretium quam vulputate dignissim suspendisse in est ante. Nulla porttitor massa id neque aliquam. Purus gravida quis blandit turpis cursus in hac. Aliquam eleifend mi in nulla posuere sollicitudin aliquam ultrices sagittis. Sit amet nulla facilisi morbi tempus iaculis urna id volutpat. Volutpat ac tincidunt vitae semper. Quis imperdiet massa tincidunt nunc pulvinar sapien et ligula ullamcorper. Non tellus orci ac auctor augue mauris. Nisi scelerisque eu ultrices vitae. Sit amet nisi purus in mollis nunc sed id. Egestas sed tempus urna et pharetra. Auctor neque vitae tempus quam pellentesque nec nam aliquam. Augue eget arcu dictum varius duis at consectetur lorem donec. Eget nulla non risi est sit amet facilisis magna. Odio ut enim blandit volutpat. Pretium viverra suspendisse potenti nulla ac tortor. At risus viverra adipiscing et. Bibendum est ultricies integer quis auctor. Eleifend donec pretium vulputate sapien. Cursus metus aliquam eleifend mi in nulla posuere. Eu tincidunt tortor aliquam nulla facilisi cras fermentum odio. Feugiat nisi pretium fusce id velit ut tortor pretium viverra. Posuere ac ut consequat semper viverra. Ipsum dolor sit amet consectetur adipiscing elit ut aliquam. Et tortor at risus viverra adipiscing et in tellus. Morbi leo urna molestie at. Non curabitur gravida arcu ac tortor dignissim convallis aenean. Eget velit aliquet sagittis id consectetur purus. Porttitor lacus luctus accumsan tortor posuere ac ut. Magna ac placerat vestibulum lectus mauris ultrices eros in cursus. Habitasse platea dictumst quisque sagittis purus. Turpis egestas sed tempus urna et pharetra pharetra massa. Donec adipiscing tristique risus nec feugiat in. Non curabitur gravida arcu ac. Quam nulla porttitor massa id neque aliquam. Urna nec tincidunt praesent semper feugiat. Sem viverra aliquet eget sit amet tellus cras adipiscing enim. Facilisis sed odio morbi quis commodo. Imperdiet nulla malesuada pellentesque elit eget gravida. Ipsum a arcu cursus vitae congue mauris rhoncus aenean vel. Et malesuada fames ac turpis egestas integer eget aliquet nibh. Amet justo donec enim diam vulputate ut. Faucibus pulvinar elementum integer enim neque. Sem et tortor consequat id porta nibh venenatis cras sed. Nulla pellentesque dignissim enim sit amet venenatis urna. Volutpat maecenas volutpat blandit aliquam etiam erat velit scelerisque. Sodales neque sodales ut etiam sit amet. Sagittis orci a scelerisque purus semper eget duis at. Amet tellus cras adipiscing enim eu. Odio euismod lacinia at quis risus sed. Nec dui nunc mattis enim ut tellus elementum sagittis. Lacus sed turpis tincidunt id aliquet. Vehicula ipsum a arcu cursus vitae congue mauris rhoncus aenean. Id ornare arcu odio ut sem nulla pharetra diam. Elit ut aliquam purus sit amet luctus venenatis. Ut enim blandit volutpat maecenas volutpat blandit aliquam etiam. Ullamcorper sit amet risus nullam eget. Cursus in hac habitasse platea dictumst quisque sagittis. Nullam vehicula ipsum a arcu. Aliquam sem et tortor consequat id porta nibh venenatis cras. Nisi rhoncus mattis rhoncus urna neque viverra justo nec ultrices. Amet purus gravida quis blandit turpis cursus. Sit amet justo donec enim diam vulputate ut. Facilisis gravida neque convallis a. Vel facilisis volutpat est velit egestas dui id. Habitasse platea dictumst quisque sagittis purus sit amet. Hac habitasse platea dictumst quisque. Est ullamcorper eget nulla facilisi etiam dignissim diam. Enim nulla aliquet porttitor lacus luctus accumsan tortor posuere. Quam id leo in vitae. Risus pretium quam vulputate dignissim suspendisse in. Dignissim suspendisse in est ante in nibh mauris cursus. Sit amet dictum sit amet justo donec enim. Turpis cursus in hac habitasse platea. Dui sapien eget mi proin sed libero enim. Varius vel pharetra vel turpis. Vitae tortor condimentum lacinia quis vel. Maecenas accumsan lacus vel facilisis volutpat est velit egestas dui. Egestas sed sed risus pretium quam vulputate dignissim suspendisse in. Tempus iaculis urna id volutpat lacus laoreet non curabitur gravida. At urna condimentum mattis pellentesque. At lectus urna duis convallis convallis tellus id. Mattis vulputate enim nulla aliquet porttitor lacus luctus accumsan tortor. Facilisis magna etiam tempor orci. Feignilla urna porttitor rhoncus dolor purus.

Comparison:



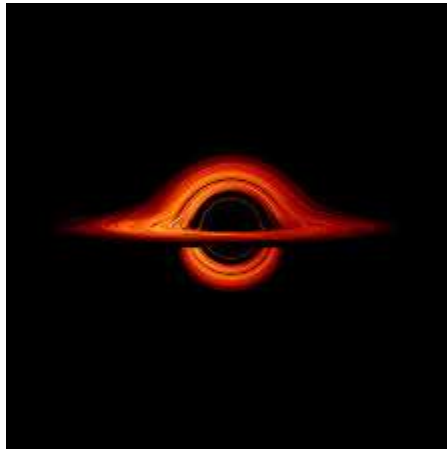
2. Original:



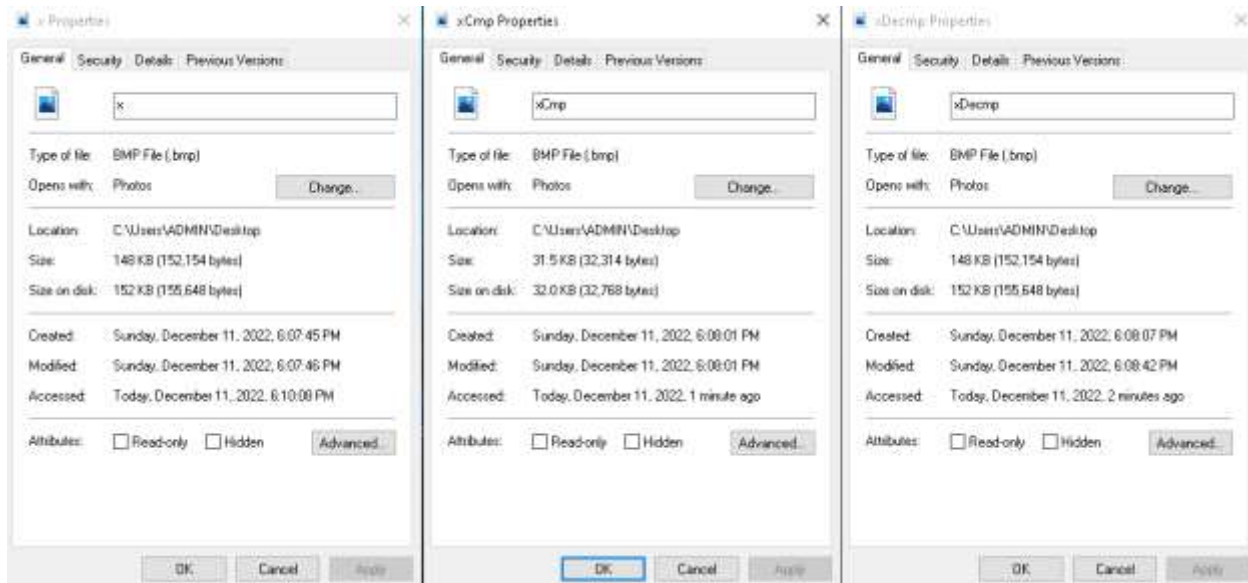
Compressed (In Text form):

2604	2605	2606	2607	2608	2609	2610	2611	2612	2613	2614	2615	2616	2617	2618	2619	2620	2621	2622	2623	2624	2625	2626	2627	2628	2629	2630	2631	2632	2633	2634	2635	2636	2637	2638	2639	2640	2641	2642	2643	2644	2645	2646	2647	2648	2649	2650	2651	2652	2653	2654	2655	2656	2657	2658	2659	2660	2661	2662	2663	2664	2665	2666	2667	2668	2669	2670	2671	2672	2673	2674	2675	2676	2677	2678	2679	2680	2681	2682	2683	2684	2685	2686	2687	2688	2689	2690	2691	2692	2693	2694	2695	2696	2697	2698	2699	2700	2701	2702	2703	2704	2705	2706	2707	2708	2709	2710	2711	2712	2713	2714	2715	2716	2717	2718	2719	2720	2721	2722	2723	2724	2725	2726	2727	2728	2729	2730	2731	2732	2733	2734	2735	2736	2737	2738	2739	2740	2741	2742	2743	2744	2745	2746	2747	2748	2749	2750	2751	2752	2753	2754	2755	2756	2757	2758	2759	2760	2761	2762	2763	2764	2765	2766	2767	2768	2769	2770	2771	2772	2773	2774	2775	2776	2777	2778	2779	2780	2781	2782	2783	2784	2785	2786	2787	2788	2789	2790	2791	2792	2793	2794	2795	2796	2797	2798	2799	2800	2801	2802	2803	2804	2805	2806	2807	2808	2809	2810	2811	2812	2813	2814	2815	2816	2817	2818	2819	2820	2821	2822	2823	2824	2825	2826	2827	2828	2829	2830	2831	2832	2833	2834	2835	2836	2837	2838	2839	2840	2841	2842	2843	2844	2845	2846	2847	2848	2849	2850	2851	2852	2853	2854	2855	2856	2857	2858	2859	2860	2861	2862	2863	2864	2865	2866	2867	2868	2869	2870	2871	2872	2873	2874	2875	2876	2877	2878	2879	2880	2881	2882	2883	2884	2885	2886	2887	2888	2889	2890	2891	2892	2893	2894	2895	2896	2897	2898	2899	2900	2901	2902	2903	2904	2905	2906	2907	2908	2909	2910	2911	2912	2913	2914	2915	2916	2917	2918	2919	2920	2921	2922	2923	2924	2925	2926	2927	2928	2929	2930	2931	2932	2933	2934	2935	2936	2937	2938	2939	2940	2941	2942	2943	2944	2945	2946	2947	2948	2949	2950	2951	2952	2953	2954	2955	2956	2957	2958	2959	2960	2961	2962	2963	2964	2965	2966	2967	2968	2969	2970	2971	2972	2973	2974	2975	2976	2977	2978	2979	2980	2981	2982	2983	2984	2985	2986	2987	2988	2989	2990	2991	2992	2993	2994	2995	2996	2997	2998	2999	3000	3001	3002	3003	3004	3005	3006	3007	3008	3009	3010	3011	3012	3013	3014	3015	3016	3017	3018	3019	3020	3021	3022	3023	3024	3025	3026	3027	3028	3029	3030	3031	3032	3033	3034	3035	3036	3037	3038	3039	3040	3041	3042	3043	3044	3045	3046	3047	3048	3049	3050	3051	3052	3053	3054	3055	3056	3057
------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------

Decompressed:



Comparison:



Problems and Challenges

There were many hurdles that we had to face to complete this project. The first being a filing error where the file pointer would go beyond the allowed boundaries. This was simply solved by opening the file using binary mode.

Another challenge was to reduce the auxiliary space used. This was done by replacing a Boolean array of size 16 to an unsigned short integer with size 2 using binary literals. This is one the rare cases where both time and space complexity decreases.

One major issue was that since the size of the BMP header was 14, the compiler self-allocates 2 bytes to make it a multiple of 4, but that would mess all the reading and would not compress the file properly. There we had to use a preprocessor directive call `pack (int)`. This would make it package all the bytes in multiple of the number specified.

Another great issue was that since the pixels are written in multiples of 3, and the width is not a multiple of 4, it would self-allocate enough bytes to make it a multiple of four. Therefore when reading or writing we had to make sure to deal with this issue using the concept of padding. We would calculate the amount of extra bytes beforehand and skip them when reading or write that many bytes of garbage when writing.

Conclusion and Breakdown

In the end, the outcome of the project is very satisfactory, and we have learnt a lot from participating in this project.

Zippping File ----→Done by 17/Nov/2022

Unzipping File ----→Done by 22/Nov/2022

Fixed File out of Bounds----→Done by 22/Nov/2022

Fixed class self-allocation----→Done by 3/Dec/2022

Fixed file self-allocation----→Done by 4/Dec/2022

Made improvements----→Done by 9/Dec/2022

Thank You