**OVERVIEW OF C++**

The C++ Programming language was developed at AT&T Bell Laboratories in the early 1980’s by Bjarne Stroustrup. He found ‘C’ lacking for simulations and decided to extend the language by adding features from his favorite language, Simula 67. Simula 67 was one of the earliest object-oriented languages. Bjarne Stroustrup called it ‘C With Classes" originally. The name C++ (pronounced C plus plus) was coined by Rick Mascitti where "++" is the C increment operator. Ever since its birth, c++ evolved to cope with problems encountered by users, and through discussions at AT&T. However, the maturation of the CH language was attested to by two events:

(i) The formation of an ANSI (American National Standard Institute) C++ committee

(ii) The publication of The Annotated C++ Reference Manual by Ellis and Stroustrup.

The latest C++ standards document was issued by ANSI/ISO in year 2003.

The major reason behind the success and popularity of C++ is that it supports the Object oriented technology, the latest in the software development and the near to the real world. One can easily judge the importance of C++ as given in the following lines:

*"Object Oriented Technology is regarded as the ultimate paradigm for the modeling of information be that data or logic. The C + + has by now shown to fulfill this goal. "*

Object is an identifiable entity with some characteristics and behavior.

**OOP Terminology and Features:**

The OOP approach is based on certain concepts that help it attain its goal of overcoming the drawbacks or shortcomings of conventional programming approaches.

These general concepts of OOP are given below:

1. Data Abstraction

2. Data Encapsulation

3. Modularity

4. Inheritance (Hierarchy)

5. Polymorphism

**Data Abstraction** :

Abstraction refers to the act of representing essential features without including the background details or explanations.

**Encapsulation :**

The wrapping up of data and functions (that operate on the data into a single unit (called class) is known as Encapsulation.

**Modularity:**

Modularity is the property of a system that has been decomposed into a set of cohesive and loosely coupled modules.

**Inheritance:**

Inheritance is the capability of one class of things to inherit capabilities or properties from another class.

**Polymorphism:**

Polymorphism is the ability for a message or data to be processed in more than one form. Polymorphism is a property by which the same message can be sent to objects of several different classes.

**C++ Data Types**

Data types are means to identify the type of data and associated operations for handling it.

C++ data types are of two types:

(i) Fundamental Types

(ii) Derived Types

* **Fundamental Data Types:**

1. int Data Type (for integers)

2. char Data Type (for characters)

3. float Data Type (for floating-point numbers)

4. double Data Type (for double precision floating-point numbers)

5. void Data Type (for empty set of values and non-returning functions)

* **Derived Data Types:**

1. Arrays

2. Function

3. Pointer

4. Reference

5. Constant

* **User-Defined Derived Data Types**

1. Class

2. Structure

3. Union

4. Enumeration

* **C ++ Tokens**

\* Identifiers

\* Keywords

\* Constants

\* Operators

* **Identifiers**

Symbolic names can be used in c++ for various data items used by a programmer in his program. For example, if a programmer desires to store a value 27 in a memory location then he can choose any symbolic name (say ROLL) and use it as given below:

ROLL = 27

The symbol ‘=’ is an **assignment** **operator**. The significance of the above statement is that ‘ROLL" is a symbolic name for a memory location where the value 27 is being stored. The character ‘ : ’ is the **statement terminator**. It marks the ends of a C++ statement.

* **Keywords**

**A keyword is a reversed word of C++.** This cannot be used as an identifier by the user in his program.

Examples of some valid constant declarations are:

• Class

• Private

• Return

• While

* **Constants**

A symbolic name or identifier which does not change its value during exception of a program is known as a constant. Constant qualifier can be used to declare constant as shown below

**Const** float Pi = 3.1415;

Examples of some valid constant declarations are:

• Const int rate = 50;

• Const Boat pi = 3.1415;

• Const char ch = ‘A’;

* **Variables**

A variable is an identifier. It is the most fundamental aspect of any computer language. It is a location in the computer memory which can store data and is given a symbolic name for easy reference.

* **I/0 statements:**

The following C++ streams can be used for the input / output purpose:

|  |  |
| --- | --- |
| Stream | Description |
| Cin | Console input |
| Cout | console output |
| Cerr | Standard error |
| Clog | Buffered version of error |

* **Operators**:

An operator is a symbol or letter used to indicate a specific operation on variable in a program. For example ‘+’ is an addition operator that adds two data items called operands.

* **Flow of control**

The architecture of a general purpose computer is von — Neumarm architecture. Such a computer use serial and sequential nature. Therefore the normal flow of execution of statements in a high level language program is also sequential i.e. each statement is executed in the order of its appearance in the program.

However, depending upon requirements of a problem, it is often required to alter the normal sequence of execution in the program. This means that we may desire to selectively execute a program segment by suitably controlling the sequence of execution of statements in the segment.

The following are the available control statements in C++

\* If

\* If . .. else

\* Switch case

* **Repetitive statements**

Some problems require that a set of statements should be executed more than one time, each time changing values of one or more variables, so that every execution is different from the previous one. This kind of repetitive execution of a set of statements in a program is known as an iterative Loop.

\* While loop

\* Do While

\* For loop

Are the iterative statements available in C++.

* **Arrays**

An Array is a structure with the help of which a programmer can refer to and perform operations on a group of similar data items such as simple lists or Tables of information.

An array whose elements are specified by a single subscript is known as single dimensional array. The array whose elements are specified by two or more subscripts is known as multi dimensional array.

* **Strings**

A string is a group of characters of any length. A string is enclosed within quotation mark is known as a literal. For example, "Hello" is a literal. The strings can be stored and manipulated as array of characters. The last character in string is always ‘\o’ a null character with ASCII value equal to 0. Thus, the effective size of an array is one more than the size of string it can hold.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| C | O | M | P | U | T | E | R | ‘\0’ |

* **Arrays of Strings**

A two dimensional array of characters is known as an array of strings. The row subscript decides as to how many strings can be stored in the array and the column subscript tells the size of each string.

For example:

Char dir[10][30];

* **Functions**:

A subprogram or function is a name given to a set of instructions that can be called by another program or a subprogram.

A function is a complete program in itself in the sense that its structure is similar to C++ main function except that the name main is replaced by the name of the function. The general form of a function if given below:

**<type> <name> (arguments)**

* **Function Prototypes**

Similar to variables, all function must be declared before they are used in a program. C++ allows the declaration of functions in the calling program with the help of a function prototype in the calling program is given below:

**<type> <name> (arguments)**

* **Parameter passing in functions**

The two way communication between the various function can be achieved through parameter and return statement. The set of parameters defined in a function are called formal or dummy parameter. Whereas the set of corresponding parameters sent by the calling function are called actual parameter.

The functions that have parameters can be called in one of the following two ways:

1. call by value

2. call by reference

* **Functions**

Before reading this tutorial you should have knowledge of pointers.

Functions are building blocks of the programs. They make the programs more modular and easy to read and manage. All C++ programs must contain the function main( ). The execution of the program starts from the function main( ). A C++ program can contain any number of functions according to the needs.

The general form of the function is:

return\_type function\_name(parameter list)

{

body of the function

}

The function consists of two parts function header and function body.

The function header is:-

return\_type function\_name(parameter list)

The return\_type specifies the type of the data the function returns. The return\_\_type can be void which means function does not return any data type. The function\_name is the name of the function. The name of the function should begin with the alphabet or underscore.

The parameter list consists of variables separated with comma along with their data types.

The parameter list could be empty which means the function do not contain any parameters. The parameter list should contain both data type and name of the variable.

For example,

int factorial(int n, float j)

is the function header of the function factorial. The return type is of integer which means function should return data of type integer. The parameter list contains two variables n and j of type integer and float respectively. The body of the function performs the computations.

* **Function Declaration**

A function declaration is made by declaring the return type of the function, name of the function and the data types of the parameters of the function. A function declaration is same as the declaration of the variable. The function declaration is always terminated by the semicolon. A call to the function cannot be made unless it is declared.

The general form of the declaration is:-

return\_type function\_name(parameter list);

For example function declaration can be

int factorial(int n, float j );

The variables name need not be same as the variables of parameter list of the function.

Another method can be

int factorial(int , float);

The variables in the function declaration can be optional but data types are necessary.

* **Function Arguments**

The information is transferred to the function by the means of arguments when a call to a function is made. Arguments contain the actual value which is to be passed to the

function when it is called. The sequence of the arguments in the call of the function should be same as the sequence of the parameters in the parameter list of the declaration of the function. The data types of the arguments should correspond with the data types of the parameters. When a function call is made arguments replace the parameters of the function.

* **The Return Statement and Return values**

A return statement is used to exit from the function where it is. It returns the execution of the program to the point where the function call was made. It returns a value to the calling code. The general form of the return statement is:-

return expression;

The expression evaluates to a value which has type same as the return type specified in the function declaration. For example the statement,

return(n);

is the return statement of the factorial function. The type of variable n should be integer I as specified in the declaration of the factorial function. If a function has return type as void then return statement does not contain any expression. It is written as:-

return;

The function with return type as void can ignore the return statement. The closing braces at the end indicate the exit of the function. Here is a program which illustrates the working of functions.

#include<iostream>

using, namespace std;

int factorial(int n);

int main ()

{

int n ,fact;

cout <<"Enter the number whose factorial has to be calculated” << endl;

cin >> n;

fact=factorial(int n );

cout << "The factorial of" << n << " is ; " << fact << endl;

return(0);

}

int factorial(int n)

{

int i=0,fact=1;

if(n<=l )

{

return(l );

}

else

{

for(i=1 ;i<=n;i++)

{

fact=fact\*i;

}

return(fact);

}

}

The function factorial calculates the factorial of the number entered by the user. If the number is less than or equal to l then function returns 1 else it returns the factorial of the number.

The statement :

int factorial( int n);

is a declaration of the function. The return type is of integer. The parameter list consists of one data type which is integer. The statement

cout <<"Enter the number whose factorial has to be calculated" << endl;

cin >> n;

makes the user enter the number, whose factorial is to be calculated. The variable n stores the number entered by the user. The user has entered number 5. The statement

fact=factorial( n1 );

makes a call to the function. The variable n is now argument to the function factorial.

The argument is mapped to the parameters in the parameter list of the function. The function header is

int factorial (int n)

The body of the function contains two return statements. If the value entered by the user is less than and equal to l then value 1 is returned else computed factorial is returned.

The type of the expression returned is integer.

* **Parameter passing mechanism**

There are two parameter passing mechanisms for passing arguments to functions such as pass by value and pass by reference.

* **Pass by value**

In pass be value mechanism copies of the arguments are created and which are stored in

the temporary locations of the memory. The parameters are mapped to the copies of the arguments created. The changes made to the parameter do not affect the arguments. Pass by value mechanism provides security to the calling program.

Here is a program which illustrates the working of pass by value mechanism.

#include<iostream>

using namespace std;

int add(int n);

int main()

{

int number,result;

number=5; ·

cout << " The initial value of number : " << number << endl;

result=add(number);

cout << " The final value of number : " << number << endl;

cout << " The result is : " << result << endl;

return(0);

}

int add(int number)

{

number=number+100;

return(number);

}

The value of the variable number before calling the function is 5.

The function call is made and function adds 100 to the parameter number. When the function is returned the result contains the added value. The final value of the number remains same as 5. This P shows that operation on parameter does not produce effect on arguments.

* **Pass by reference**

Pass by reference is the second way of passing parameters to the function. The address of the argument is copied into the parameter. The changes made to the parameter affect the arguments. The address of the argument is passed to the function and function modifies the values of the arguments in the calling function.

Here is a program which illustrates the working of pass by reference mechanism.

#include<iostream.h>

namespace std;

int add(int &number);

int main ()

{

int number;

int result;

number=5;

cout << "The value of the variable number before calling the function : " <<

number << endl;

result=add(&number); ‘

cout << "The value of the variable number after the function is returned 1 " <<

number << endl;

cout << "The value of result ; " << result << endl;

return(0);

}

int add(int &p)

{

\*p=\*p+l OO;

return(\*p);

}

The address of the variable is passed to the function. The variable p points to the memory address of the variable number. The value is incremented by 100. It changes the actual contents of the variable number. The value of variable number before calling the function is 100 and after the function is retuned the value of variable number is changed to 105.

* **THE FSTREAM HEADER FILE :**

The C++ input/output operations are very much similar to the console input and output operations. The file operations also make use of streams as an interface between the programs and the files.

A STREAM IS A SEQUENCE OF BYTES.

A stream is a general name given to a flow of data. Different streams are used to represent different kinds of data flow. Each stream is associated with a particular class, which contains member functions and definitions for dealing with that particular kind of data flow.

The stream that supplies data to the program is known as input stream. It reads the data from the file and hands it over to the program. The stream that receives data from the file is known as output stream.

Write data Read data

Disk file

T o

memo r y

File

to File

Memo r y

Program

✠ **FUNCTIONS OF FILE STREAM CLASSES:**

Filebuf it sets the file buffers to read and write.it contains Close( ) and open( ) member functions in it,

Fstreambase this is the base class for fstream, ifstream and

Ofstream classes.

Ifstream being an input file stream class, it provides input operations for file. It inherits the function get( ), getline( ), read( ) and functions supporting random access from istream class defined inside iostream.h file.

Ofstream being an output file stream class, it provides output

Operations.

Fstream it is an input/output stream class. It provides

support for simultaneous input/output operations.

**Introduction**

Ever since human began to count and do simple arithmetic they have tried to make the process easier and faster by the use of machines. From counting on their fingers, humans progressed to using pebbles to represent numbers and this led to the invention of abacus (a Form of digital computer), a device that is still widely used in many countries today.

A major advance in mathematics was the system of calculation by logarithms, devised by John Napier at the end of sixteenth century and first published in 1614. Following this discovery, the English clergyman William Oughtred invented sliding scales, an early type of slide rule, which by its use of lengths to repressed numbers, is in effect a form of analog computer. The abacus, which operates by counting rather than measuring is in comparison a digital device.

The first mechanical calculating machines appeared during the seventeenth century. Blaise Pascal produced a machine that could add and subtract, performing multiplications and division by repeated addition and subtraction. Some years later Leibniz invented a calculator, which could perform all these functions individually.

Charles Babbage designed the true ancestor of the modem computer in the 1830s. This machine, the analytical Engine, was never completed but it was intended to perform any desired calculation automatically by means of a mechanical calculating unit controlled by punched cards, originally developed for the Jacquard loom, from the basis of the car — handling machines developed by Dr.Herman Hollerith in the 1880s.

Much water has flown since then and the invention of modem electronic computers has made possible what seemed to impossible. The modem computers, though smaller in size, can handle any amount of data and perform complex functions.

The project entitled as “**Text Editor** (notepad)”which helps the user to keep intact with day to day life. This project includes the following features such as

**Type** any data or information ,

**Save** the information typed using save option ,

**Edit** the information previously written ,

**View** the information or data which is already saved using the open option.

This project is very useful for students, organization like school to save any

information .

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* TEXT EDITOR\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

This is a C++ program created for management of information using text editor.

Various C++ concepts like

• Data File handling

• Classes and Objects

• For Loop, do while Loop switch cases.

have been used for its simplicity.

It provides with a menu in which a person can

* Add information
* View the entire details
* Edit the information
* Save the information
* View the help menu
* Exit from the editor.

The user can view all the information from the program.

The user can create a new file by specifying the path of file.

Similarly this program helps the users to edit their information in the existing file. This ensures that if any information is stored incorrectly, it can be rectified.

The user has to enter the path of the file where he has saved to open the existing file and can edit or view it.

**SYSTEM REQUIREMENTS**

INTEL 1.5 GHZ PENTIUM I PROCESSOR

VGA CARD

14" CRT MONITOR

16 MB RAM

20 GB HDD

104 KEYS KEYBOARD

**\*+\*+\*PROGRAM PROCEDURES\*+\*+\***

Various functions have been used in this program to make the program easier and simpler.

In **main()** funtion the graphic driver is detected and the graphic mode, graphic driver is intialised.

Then the member function **first()** in the public of class **text** is called using its object **a.**

The function **textmode(BW80)** changes screen mode in text mode. When textmode is called, the current window is reset to the entire screen, and the current text attributes are reset to normal.

BW80 changes screen into black and white having 80 columns.

Then the functions **starting()** and **typing()** is called successively from the **main()**.

Inline functions like

* **void first();**
* **void intro();**
* **void help();**
* **void bye();**

have been used inside a class called **text** as its public members.

The **first()** is a function which is used to display the menu page of the text editor. It requests the user to enter one of the options from menu.

In the menu, the user has the choice to

* Start the text editor
* Read the help menu
* Exit from the text editor

If the user presses 1, then the function **intro()** is called which loads the text editor program.

If the user presses 2, then the function **help()** is called which displays the various keys and shortcuts that can be used in the text editor.

If the user presses 3, then the function **bye()** is called and the user can exit from the text editor program.

If the user presses any other key other than those mentioned above then the program displays error message called **“INVALID CHOICE”**

And then the function **first()** is called.

Functions like

* **void starting();**
* **void typing();**
* **void openfile(char \*);**
* **void newfile(char \*);**

have been declared outside the class.

•The **starting()** function is used to give layout of the notepad .

•The **typing()** function is used to type in the notepad using their ASCII value.

•The **openfile(char \*)** function is used to open the existing file from the Directory.

•The **newfile(char \*)** function is used to create a new file by closing the existing file.

In the **typing()** function all the cases are repeated till the save option is given by the user. If while typing the information, if the line is ended then the cursor automatically goes to the next line.

While typing, each character is converted into its ASCII value and the required option by the key is done. For example : if enter key is pressed then the row is incremented and goes to the first column. It is done by comparing the ASCII value of enter key. Similarly other functions like

* Backspace key
* Tab function
* Enter key
* Upper arrow key
* Right arrow key
* Left arrow key
* Down arrow key
* Case to create new file ctrl+n
* Case to open existing file ctrl+o
* Case to display help portion ctrl+k
* Case to quit the text editor ctrl+q
* Case to save the information ctrl+s

Can also be used in the text editor program.

If ctrl+n is pressed its ASCII value is 14 and the program prompts the user to type the path of the file where he has to create the new file. Then the function **newfile(char \*s)** is called along with file’s path as its arguments. The file is opened in **ofstream** so that new information could be added, it creates the empty file at the specified path and then the file is closed.

If ctrl+o is pressed its ASCII value is 15 and the program prompts the user to type the path of the file which has to be open. Then the function **openfile(char \*xs)** is called along with file’s path as its arguments. The file is opened in **ifstream** mode using the object.

It is read using the **file.read((char\*)&ch,sizeof(ch))** function and then the file is closed.

If ctrl+s is pressed its ASCII value is 19 and the program prompts the user to type the path of the file where he has to be save. The file is opened in **ofstream** mode using the object.The information is written using the **f1.write((char\*)&p[j],sizeof(p[j]))** function and then the file is closed.

Thus the user is benefited greatly by using this program for storing the information. This program displays anything he wanted, before his eyes, in a time less than a second.

**THANK YOU FOR USING THIS PROGRAM**

**\*+\*+\*PROGRAM CODE\*+\*+\***

#include<dos.h>

#include<process.h>

#include<string.h> // for various functions of strings like gets or puts

#include<fstream.h> // for files

#include<stdio.h>

#include<iostream.h>

#include<conio.h>

#include<graphics.h>

void starting();

void typing();

void openfile(char \*);

void newfile(char \*);

char ch;

char s[20];

char xs[20];

char ys[20];

class text

{

private:

int l;

public:

void first()

{

clrscr();

cleardevice();

setcolor(RED+BLINK);

settextstyle(4,0,6);

outtextxy(120,0,"DEVI ACADEMY ");

setcolor(YELLOW);

settextstyle(5 ,0,6);

outtextxy(35,110,"Computer science Project");

setcolor(9+128);

settextstyle(10,0,3);

outtextxy(0,220,"Done by") ;

setcolor(GREEN);

settextstyle(5 ,0,4);

outtextxy(0,280,"E.Rahul");

outtextxy(0,320,"M.Ramkrishna");

outtextxy(0,360,"K.Swetharanyan");

getch();

cleardevice();

settextstyle(5,0,4);

setcolor(YELLOW);

outtextxy(220,0,"WELCOME TO");

setcolor(RED+BLINK);

settextstyle(4,0,6);

outtextxy(150,30,"TEXT EDITOR");

setcolor(9+128);

settextstyle(10 ,0,3);

outtextxy(220,110,"Main Menu");

outtextxy(0,180,"1. START ");

outtextxy(0,220,"2. HELP");

outtextxy(0,260,"3. EXIT");

setcolor(YELLOW);

settextstyle(5,0,6);

outtextxy(0,320,"Enter your option ");

cin>>l;

if(l==1)

intro();

else if (l==2)

help();

else if(l==3)

bye();

else

{

cleardevice();

clrscr();

cout<<"INVALID CHOICE ";

delay(500);

first();

}

}

void intro()

{

cleardevice();

settextstyle(2,0,10);

setcolor(15);

outtextxy(220,0,"LOADING...");

for(int k=0;k<710;k++)

{

settextstyle(2,0,10);

setcolor(15);

outtextxy(k,20,".");

delay(30);

}

sound(5600);

delay(50);

nosound();

cout<<"\n\n\n\n\nPress enter key to continue";

getch();

}

void help()

{

clrscr();

gotoxy(20,3);

cout<<"~`~`~`~`~ HELP PORTION ~`~`~`~`~";

gotoxy(20,4);

cout<<"Following Are the Keys Used in Text editor ";

int za=10;

for(int i=0;i<46 ;i++ )

{

gotoxy(za,5);

cout<<"=";za++;

}

za=6;

for(i=0;i<15;i++)

{

gotoxy(10,za);

cout<<"|"<<endl;

za++;

}

za=10;

for( i=0;i<46 ;i++ )

{

gotoxy(za,21);

cout<<"=";

za++;

}

za=6;

for(i=0;i<15;i++)

{

gotoxy(56,za);

cout<<"|"<<endl;

za++;

}

gotoxy(13,6);

cout<<" 1-To Open File Press ( ctrl + o ) ";

gotoxy(13,8);

cout<<" 2-To create New File Press(ctrl+n) " ;

gotoxy(13,10);

cout<<" 3-Move Arrow keys (up,down,right,left)" ;

gotoxy(13,12);

cout<<" 4-Back Space ";

gotoxy(13,14);

cout<<" 5-Save File(ctrl+s)";

gotoxy(13,16);

cout<<" 6-Enter For new line ";

gotoxy(13,18);

cout<<" 7-Enter ctrl+q(Quit)";

gotoxy(13,20);

cout<<"Press any key to Go back " ;

getch();

clrscr();

a.first() ;

}

void bye()

{

cleardevice();

settextstyle(5,0,50);

setcolor(10);

outtextxy(150,120,"BYE!!!");

getch();

exit(0);

}

}a;

void main()

{

clrscr();

int gdriver=DETECT,gmode;

initgraph(&gdriver,&gmode,"c:\\tc\\bgi");

a.first();

textmode(BW80);

starting();

typing();

getch();

}

void starting()

{

textcolor(WHITE);

textbackground(BLUE);

clrscr();

for(int i=0 ;i<45;i++)

{

if(i==0||i==44)

cout<<"||";

else if(i==4)

cout<<" ctrl+O(Open file)";

else if(i==24)

cout<<" ctrl+n(New file)";

else

cout<<"=";

}

gotoxy(80,24); cout<<endl;

for( i=0 ;i<37;i++)

{

if(i==0||i==47)

cout<<"||";

else if(i==4)

cout<<" ctrl+s(save file)";

else if(i==16)

cout<<" ctrl+k(Help)";

else if(i==26)

cout<<"ctrl+q(Quit)";

else

cout<<"=";

}

}

void typing()

{

int i=0;

char \*p=new char[2000];

step1 : int row=2 ,col=1;

while(ch!=19) //to save the present file ctrl+s

{

gotoxy(col,row);

step2: if(col==79)

{

col=1;row++;

}

else if (row==24)

goto step1;

int a;

ch=getch();

if(ch == 0)

ch = getch();

a=ch; // To convert character to its ascii code

switch(a)

{

case 13 :

row=row++;col=1; // case for press enter for next line

gotoxy(col,row);

ch='\n';

break;

case 8 :

if(col==1) // case for backspace

{

row--;col=78;

}

else

{

col--;

cout<<" ";

i--;

}

continue;

case 9 :

col=col+8; // case for tab function

if(col>79)

{

row++;col=col-79;

}

continue;

case 72 :

row--; //case for upper arrow key

continue;

case 77 :

col++;

continue; //case for right arrow key

case 75 :

col--;

continue; //case for left arrow key

case 80 : //case for down arrow key

row++;

continue;

case 14 :

clrscr(); //case to create new file ctrl+n

cout<<"Enter File`s Path ";

gets(s);

newfile(s);

starting();

break;

case 15 :

clrscr(); //case to open existing file ctrl+o

cout<<"Enter file`s path to be open :" ;

gets(xs);

starting();

openfile(xs);

continue;

case 11 :

clrscr(); //case to display help portion ctrl+k

gotoxy(20,3);

cout<<"~`~`~`~`~ HELP PORTION ~`~`~`~`~";

gotoxy(20,4);

cout<<"Following Are the Keys Used in Note PAD : ";

int za=10;

for(int i=0;i<46 ;i++ )

{

gotoxy(za,5);

cout<<"=";za++;

}

za=6;

for(i=0;i<15;i++)

{

gotoxy(10,za);

cout<<"|"<<endl;

za++;

}

za=10;

for( i=0;i<46 ;i++ )

{

gotoxy(za,21);

cout<<"=";

za++;

}

za=6;

for(i=0;i<15;i++)

{

gotoxy(56,za);

cout<<"|"<<endl;

za++;

}

gotoxy(13,6);

cout<<" 1-To Open File Press ( ctrl + o ) ";

gotoxy(13,8);

cout<<" 2-To create New File Press(ctrl+n) " ;

gotoxy(13,10);

cout<<" 3-MOve Arrow keys (up,down,right,left)" ;

gotoxy(13,12);

cout<<" 4-Back Space ";

gotoxy(13,14);

cout<<" 5-Save File(ctrl+s)";

gotoxy(13,16);

cout<<" 6-Enter For new line ";

gotoxy(13,18);

cout<<" 7-Enter ctrl+q(Quit)";

gotoxy(13,20);

cout<<"Press any key to Go back " ;

getch();

clrscr();

starting();

continue;

case 17 :

clrscr(); //for closing the text editor ctrl+q

gotoxy(24,6);

cout<<"Thank You For using TEXT EDITOR ";

cout<<endl;

gotoxy(24,8);

cout<<"Have a Nice Time ";

getch();

exit(0);

}

cout<<ch;

p[i]=ch;

col++;

i++;

p[i]='\0';

}

clrscr();

cout<<"enter path where you want to save the file ";

gets(ys); int j=0;

ofstream f1(ys,ios::out);

while(p[j]!='\0')

{

f1.write((char\*)&p[j],sizeof(p[j]));

j++;

}

f1.close();

delete p;

exit(0);

}

void newfile(char \*s)

{

ofstream file(s,ios::out);

file.close();

}

void openfile(char \*xs)

{

char ch; int col=3,row=2;

gotoxy(col,row);

ifstream file(xs,ios::in);

while( file.read((char\*)&ch,sizeof(ch)))

cout<<ch;

file.close();

}

**Sample input & output:**

|  |
| --- |
| **DEVI ACADEMY**  **Computer science project**  **Done by:**  **E.Rahul**  **M.Ramkrishna**  **K.Swetharanyan** |

|  |
| --- |
| Welcome to  Text Editor  Main menu  1. Start  2. Help  3. Exit  Enter your option…  1 |

**Starting editor….**

|  |  |
| --- | --- |
| Loading. . .  Enter any key to continue   |  | | --- | | ||=====ctrl+o(openfile)=====ctrl+n(newfile)=====||  Welcome to text editor  ||===ctrl+s(save file)==ctrl+h(help)==ctrl+q(quit)==|| |   . |

**Saving file:**

|  |
| --- |
| Enter path to save the file: C:\tc\bin\newfloder\comp.txt  // the file created on the specified location |

**Opening file:**

|  |  |
| --- | --- |
| Enter path to open the file:C:\tc\bin\newfoleder\comp.txt   |  | | --- | | //this is opened file, the file contains  Welcome to text editor | |

**Getting help:**

|  |
| --- |
| Welcome to  Text Editor  Main menu  1. Start  2. Help  3. Exit  Enter your option…  2 |

|  |
| --- |
| ~`~`~`~`~ HELP PORTION ~`~`~`~`~  Following Are the Keys Used in Text editor  ==================================  || 1-To Open File Press ( ctrl+o ) ||  || 2-To create New File Press(ctrl+n) ||  || 3-MOve Arrow keys (up,down,right,left) ||  || 4-Back Space ||  || 5-Save File(ctrl+s) ||  || 6-Enter For new line ||  || 7-Enter ctrl+q(Quit) ||  ==================================  Press any key to Go back…  Esc |

**Exiting the editor:**

|  |
| --- |
| Welcome to  Text Editor  Main menu  1. Start  2. Help  3. Exit  Enter your option…  3 |

|  |
| --- |
| BYE!!! |

**Bibliography**

* Introduction to C++ - Sumita Arora
* Object Oriented Programming with C++ - Balagurusamy
* Teach Yourself C++
* Let us C – BPB Publication