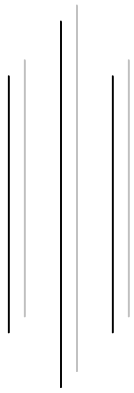
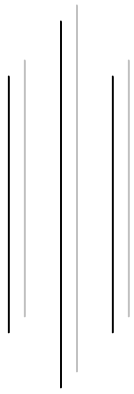
TRIBHUWAN UNIVERSITY

INSTITUTE OF ENGINEERING
Pulchowk Campus, Lalitpur



mini project

"Breaker"



SUBMITTED BY:

- Pravesh Gaire (073/BEX-427)

SUBMITTED TO:

Department of Electronics and Computer

ACKNOWLEDGEMENT

It would have been very hard for us to complete our project on time and in efficient way. Without the support, guidance and suggestion from our faculty members and friends, we could not have completed this project on time and in efficient way.

First of all, we have no words that can adequately carry our sincere gratitude to, ***Mr. Sanjay Bhandari,***, *Subject teacher* for providing us the all the opportunity to conduct the project. We are very grateful to him for his valuable suggestions and encouragement for this project work. He has provided us all sort of basic needs and techniques essential for carrying out this project work.

We would also like to acknowledge and thank all the faculty members for their kind support and guidance as well as their inspiration to move forward for the completion of project on time.

We can't leave to thank our dear friends especially Sandesh Bhusal-BCT and Mitesh Pandey-BEX as they have provided us with their valuable comments and suggestions for the completion along with motivation when we were let down.

We would also like to thank all the writer of the books that we have used. Without their books, it would have been hard and tedious for us to get knowledge and information regarding the subject matter and it might have taken long time to complete this project.

Finally, we would like to provide our sincere gratitude to all who have helped us in completing this project in highly efficient way.

Thanks.

TABLE OF CONTENTS

BACKGROUND

C is a programming language developed at AT & T's Bell laboratories of USA in 1972. It was designed and written by Denis Ritchie.

All programming languages can be divided into two categories:

(a) Problem oriented languages or High level languages:
Examples of languages falling in this category are FORTRAN, BASIC, Pascal, COBOL etc. These languages have been designed to give a better programming efficiency i.e. faster program development

(b) *Machine oriented language or Low level languages:*
Examples of languages falling in this category are Assembly language and Machine language.

These languages have been designed to give a better machine efficiency. i.e. faster program execution.

C stands in between these two categories. That's why it is called as Middle level Language, since it was designed to have both; a relatively good programming efficiency (as compared to Machine Oriented Languages) and relatively good machine efficiency (as compared to Problem Oriented Languages).

Communicating with a computer involves speaking the language the computer understands, which immediately rules out English as the language of communication with computer. However, there is a close analogy between learning English language and learning C language. The classical method of learning English is to first learn the alphabets or characters used in the language, then learn to combine these alphabets to form words, which in turn are combined to form sentences are combined to form paragraphs. Learning C is similar and much easier.

Therefore, instead of straight-away learning how to write programs, we must first know what alphabets, numbers and special symbols are used in C, then how the constants, variables and keywords are constructed, and finally how these are combined to form an instruction. A group of instructions would be combined later on to form a program.

Features of good program:-

Programmers must keep in mind the following things before writing a program in order to have a good program.

- ★ **Integrity**
  The calculations used in the program should be very accurate. It must provide the desired output (functionally correct) for the given input. It must do the work according to the specification.

- ★ **Clarity:-**
  The program should be well readable to aid maintenance later. This can be provided inline-documentation or external documentation. On inline documentation, the function of each piece of code is defined within the program itself. In the external documentation, a separate report includes the working principle of each module inside the program.

- ★ **Simplicity:-**
  The program should be able to express the logic in a considerably simple way. This feature enhances Integrity and Clarity. Same problem can be done in two or more ways, but one needs to choose the simplest way to solve the problem.

- ★ **Efficiency:-**
  The program should have a good compromise between Time and space used, i.e. it should run as fast as possible (time) with the minimum memory requirement (space).

- ★ **Modularity:-**
  Program should be separated to different logical and self-contained modules. If the entire program is divided into simpler contained modules then one can easily understand what's happening inside it.

- ★ **Generality:-**
  The program should be as general as possible within certain limits.

- ★ **Robustness:-**
  Program must be fault-tolerant as much as possible. A program cannot be 100% full-proof, so it must be built in such a way that, if some unavoidable circumstance appears, then it tackles with this without being crashed.

- ★ **Security:-**
  A program must be secured enough so as to avoid tampering from unwanted people. All the loopholes in the programs must be avoided as much as possible.

INTRODUCTION

"Breaker" is the program written on C language using TurboC++ compiler. It is a game where player has to destroy all the bricks using a ball controlled by paddle. It can be very entertaining and good to play in leisure time.

It has been made very user friendly and we have hoped that the user may feel very comfortable to use this game without having any such difficulty. It has three levels,so user can play accordingly to his capability.

We have also made a fake loading at starting to make it more attractive.The main screen has a big BRICK written using small boxes and octave sound is played till user hits any key.Then we enter to menu page. In menu there are four options.User can choose it by typing P for Play, I for instructions, C to know about developer and E to exit. If user chooses to play then he is asked to enter the level he wants to play. There are three levels for now, Novice, Advanced, Expert.The game has 3 lives. The aim is to hit all the bricks using ball and score 500 points. Each brick destroyes increases score by 5 and if ball is dropped 20 points is subtracted from score.User can leave anytime he wants pressing Esc to menu.

Before starting the development of this program, we had certain objectives in mind to fulfill which are listed as below:

- To use different in-built functions of C and also make the user defined functions for making the program look simple.
- To make the use of array and pointer as far as possible to make the program look simple and easier to understand.
- To make use of structure and union.
- To use graphics to make the program look attractive and user friendly as far as possible.

GENERAL THEORY

The different in-built functions that are used in the program are:

- **gotoxy**

  Positions cursor in text window

  Syntax:

  gotoxy(int x, int y);

- **switch case and default**

  switch
  Causes control to branch to one of of a list of possible statements in the block defined by <statement>.
  case
  The list of possible branch points within <statement> is determined by preceding substances with
  case <constant expression> :
  default
  If the match is not found and the "default :" statement prefix is found within<statement>, execution continues at this point. Otherwise, <statement> is skipped entirely.

  Syntax:

  switch(<expression>) <statement>
  case <constant expression> :
  default :

- **printf**

  Sends formatted output to stdin

  Syntax

  printf ("<formatted string>",<list of variable>);

  Where, <format string> could be:

  %f->for printing real values
  %d->for printing integer values
  %c->for printing character values
  %s->for printing character values.

- **for loop**

  Executes the specified statement as long as the condition is TRUE.

  Syntax

  for ([initialization] ; [condition] ; [expression]) statement

  initialization Initializes variables for the loop. initialization can be an expression or a declaration. Variables are initialized before the first iteration of the loop.

  condition Must evaulate to either TRUE or FALSE. When FALSE, statement stops executing.

  expression The expression to evaluate after each iteration of the loop. expression usually increments or decrements the initialization variable in some way.

  statement The statement to be executed. statement executes repeatedly as long as the value of condition remains TRUE.

- **while**

  Repeats the statements till condition is true.

  Syntax:

  while(<expression>) <statement>

- **if**

  Runs a statement if condition is true else ignores it.

  Syntax:

  if (<expression>) <statement>

  else if (<expression>) <statement)

  else (<statement>)

- **continue and break**

  continue causes control to pass to the end of the innermost enclosing while, do, for statement, at which point the loop continuation condition is re-evaluated.

break causes control to pass to the statement following the innermost enclosing while, do, for, or switch statement.

Syntax:

continue;
break;

- **getch and getche**

getch gets a character from console but doesnot echo to the screen.
getche gets a character from console and echoes to the screen.

Syntax:
getch();
getche();

- **getmaxx and getmaxy**

Returns maximum x or y screen coordinate

Syntax

Maxx=getmaxx(); // Maxx is available to store the maximum x coordinate
Maxy=getmaxy();

- **rectangle**

Draws a rectangle.

Syntax:

rectangle(int left, int top, int right, int bottom);

- **floodfill**

Fills an enclosed area on bitmap devices.

Syntax

floodfill(int x, int y, int border);

- **line**

Creates a line from (x1,y1) to (x2,y2).

Syntax:

line (int x1, int y1, int x2, int y2);

- **circle**

  Draws a circle.

  Syntax

  circle(int x, int y, int radius);

- **imagesize**

  Retuns the number of bytes required to store a bit image.

  Syntax

  imagesiz(int left, int top, int right, int bottom);

- **malloc**

  Allocates memory.

  Syntax

  malloc( );

- **settextstyle**

  Sets the current text characteristics.

  Syntax:

  settextstyle(int font, int direction, int charsize);

- **setcolor**

  Sets the current drawing color.

  Syntax:

  setcolor(color);

- **getimage**

Saves a bit image of specified region into memory

Syntax

getimage(int left, int top, int right, int bottom);

- **putimage**

Outputs a bit image onto the screen

Syntax

putimage(int left, int top, int right, int bottom);

- **outtextxy**

Displays a string at the specified location (graphics mode)

Syntax

Outtextxy(int x, int y, " <string> ");

- **setviewport**

Sets the curremt viwport for graphics output.

Syntax

setviewport(int left, int top, int right, int bottom, int clip);

- **clearviewport**

Clears the current viewport

Syntax:

clearviewport();

- **restorecrtmode**

Restores screen mode to pre0initgraph setting

Syntax:

restorecrtmode();

- **closegraph**

Shuts down the graphics system.

Syntax:

closegraph();

- **delay**

Suspends executiom for interval (milliseconds)

Syntax

delay (milliseconds);

- **exit**

Terminates the program

Syntax:

exit(0);

- **kbhit**

Checks for currently available keystrokes.

Syntax:

int kbhit(void);

- **sound and nosound**

sound turns the PC speaker on at the specified frequency and nosound turns it off.

Syntax:

sound(unsigned frequency);

- **REGS (union)**

The union REGS is used to pass information to and from these functions:
int 86 int86x intdos intdosx

- **fflush(stdin)**

  Flushes a stream

  Syntax:

  fflush(stdin);

- **toupper**

  Translates characters to uppercase.

  Syntax:

  toupper(string);

ALGORITHM

- **Header's**
  1. Start
  2. Declare all header file, stdio.h, stdlib.h, conio.h, dos.h and graphics.h, used in program.
  3. Define all macros.
  4. Declare all global variables.
  5. Declare all function prototypes.
  6. Stop

- **main function**
  1. Start
  2. Declare all variables needed in the program.
  3. Initialize the graphics function.
  4. Get the maximum x and y coordinates and also midvalues.
  5. Call mainscreen UDF to playerlevel.
  6. Set the speed of ball as per the level choosen in UDF mainscreen().
  7. Draw the bricks in function bricks paddle and ball.
  8. Allocate memory of paddle to area using malloc.
  9. Display balls in hand initially three.
  10. Display initial score.
  11. Make different conditions to control the movement of ball as it hits different sides or bricks or paddle and also the movement of paddle.
  12. Send the data's to UDF erasebricks of bricks hit by ball to erase them.
  13. Stop

- **mainscreen**
  1. Start
  2. Declare and initialize a two dimensional array showing the positions where a brick is to be formed to form figure "BRICKS".
  3. Declare all variables needed.
  4. Draw boundary and form the word bricks using above array and rectangle function.
  5. Draw the pattern, paddle and ball at the bottom of opening screen.
  6. Make a menu screen which contains Play, Instructions, Credits, Exit option's
  7. User can choose any option by typing.
  8. Write set of instructions to display them as user enters I.
  9. Write about developers name and roll no as user enters C.
  10. Use exit function to quit if user enters E.
  11. Create a level selection menu if user enters P.
  12. User can choose three levels.
  13. Stop

- **bricks**
  1. Start
  2. Declare variables needed.
  3. Use nested for loops and send the values for making rectangle at appropriate plces repeatdly to UDF drawbrick.
  4. Stop

- **drawbrick**
  1. Start
  2. Use the values from UDF bricks and make rectangles using function rectangle.
  3. Stop

- **erasebrick**
  1. Start
  2. Set the color black.
  3. Get the values of brick to erase and color it black to look erased.
  4. Stop

- **music**
  1. Start
  2. Initialize array octave with natural frequencies of 7 notes.
  3. Use switch to use different sound's
  4. Case 1 produces octave sound at a delay of 500 miliseconds.
  5. Case 2 produces octave souds at a delay of 100 miliseconds.
  6. Similarly make sounds as per requirement.
  7. Stop

SOURCE CODE

```
# include <stdio.h>
# include <stdlib.h>
# include<conio.h>
# include <dos.h>
# include <graphics.h>
# define NULL 0
# define YES 1
# define NO 0
int maxx, maxy, midx, midy ;
int bri[5][20] ;
void load();
void bricks();
void erasebrick();
void drawbrick();
void music();
void main()
{
  union REGS ii, oo ;
  int ballx, bally, paddlex, paddley, dx = 1, dy = -1, oldx, oldy, playerlevel  ;
  int gm = CGAHI, gd = CGA;
  int i, flag = 0, speed = 25, welldone = NO, score = 0, chance = 4, area ;
  int layer[5] = { 10, 20, 30, 40, 50 }, limit = 50, currentlayer = 4 ;
  char *p1, *p2 ;
  initgraph ( &gd, &gm, "C:\\TURBOC3\\BGI" ) ;
  maxx = getmaxx() ;
  maxy = getmaxy() ;
  midx = maxx / 2 ;
  midy = maxy / 2 ;
  load();
  playerlevel = mainscreen() ;
  switch ( playerlevel )
 {
   case 'N' :
   case 'n'  :
               speed=30
               break;
   case 'A' :
   case 'a'  :
```

```
                speed = 15 ;
                break ;

  case 'E' :
  case 'e'  :
                speed = 5 ;
                break;
 }
rectangle ( 0, 0, maxx, maxy - 12 ) ;
bricks() ;
rectangle ( midx - 25, maxy - 7 - 12, midx + 25, maxy - 12 ) ;
floodfill ( midx, maxy - 1 - 12, 1 ) ;
circle ( midx, maxy - 13 - 12, 12 ) ;
floodfill ( midx, maxy - 10 - 12, 1 ) ;
area = imagesize ( midx - 12, maxy - 18, midx + 12, maxy - 8 ) ;
p1 = malloc ( area ) ;
area = imagesize ( midx - 25, maxy - 7, midx + 25, maxy - 1 ) ;
p2 = malloc ( area ) ;
getimage ( midx - 12, maxy - 7 - 12 - 12 + 1, midx + 12, maxy - 8 - 12, p1 ) ;
getimage ( midx - 25, maxy - 7 - 12, midx + 25, maxy - 1 - 12, p2 ) ;
paddlex = midx - 25 ;
paddley = maxy - 7 - 12 ;
ballx = midx - 12 ;
bally = maxy - 7 - 12 + 1 - 12 ;
gotoxy ( 45, 25 ) ;
printf ( "Balls Remaining:" ) ;
for ( i = 0 ; i < 3 ; i++ )
{
  circle ( 515 + i * 35, maxy - 5, 12 ) ;
  floodfill ( 515 + i * 35, maxy - 5, 1 ) ;
}
gotoxy ( 1, 25 ) ;
printf ( "Your Score:   %4d", score ) ;
settextjustify ( CENTER_TEXT, CENTER_TEXT ) ;
settextstyle ( SANS_SERIF_FONT, HORIZ_DIR, 4 ) ;
while ( 1 )
        {
                flag = 0 ;
               oldx = ballx ;oldy = bally ;
                ballx = ballx + dx ;
```

```
 bally = bally + dy ;
if ( bally > 40 )
 {
        limit = 50 ;
        currentlayer = 4 ;
 }
 else
 {
        if ( bally > 30 )
        {
                limit = 40 ;
                currentlayer = 3 ;
        }
        else
        {
                if ( bally > 20 )
                {
                        limit = 30 ;
                        currentlayer = 2 ;
                }
                else
                {
                        if ( bally > 10 )
                        {
                                limit = 20 ;
                                currentlayer = 1 ;
                        }
                        else
                        {
                                limit = 10 ;
                                currentlayer = 0 ;
                        }
                }
        }
 }

if ( ballx < 1 )
{
music ( 4) ;
ballx = 1 ;
```

```
dx = -dx ;
}
if ( ballx > ( maxx - 24 - 1 ) )
{
music ( 4 ) ;
ballx = maxx - 24 - 1 ;
dx = -dx ;
}
if ( bally < 1 )
{
music ( 4) ;
bally = 1 ;
dy = -dy ;
}
if ( bally < limit )
{
if ( bri[currentlayer][ ( ballx + 10 ) / 32 ] == 1 )
{
for ( i = 1 ; i <= 6 ; i++ )
{
if ( bri[currentlayer][ ( ballx + i + 10 ) / 32 ] == 0 )
{
ballx = ballx + i ;
flag = 1 ;
break ;
}
if ( bri[currentlayer][ ( ballx - i + 10 ) / 32 ] == 0 )
{
ballx = ballx - i ;
flag = 1 ;
break ;
}
}
if ( !flag )
{
if ( bally < layer[currentlayer - 1] )
{
currentlayer-- ;
limit = layer[currentlayer] ;
}
```

```
putimage ( oldx, oldy, p1, OR_PUT ) ;
putimage ( oldx, oldy, p1, XOR_PUT ) ;
putimage ( ballx, bally, p1, XOR_PUT ) ;
delay ( speed ) ;
continue ;
}
}
music ( 3 ) ;
erasebrick ( ( ballx + 10 ) / 32, currentlayer ) ;
if ( ( ballx + 10 ) / 32 == 19 )
line ( maxx, 0, maxx, 50 ) ;
if ( ( ballx + 10 ) / 32 == 0 )
line ( 0, 0, 0, 50 ) ;
if ( currentlayer == 0 )
line ( 0, 0, maxx, 0 ) ;  /* redraw top boundary */
bri[currentlayer][ ( ballx + 10 ) / 32 ] = 1 ;
bally = bally + 1 ;
dy = -dy ;  score += 5 ;
gotoxy ( 16, 25 ) ;
printf ( "%4d", score )
if ( welldone == NO )
welldone = YES ;
else
{
outtextxy ( midx, midy, "Well done!" ) ;
delay(10);music ( 3 ) ;
}
}
if ( bally > 50 && welldone == YES )
{
setviewport ( midx - 32 * 2.5, midy - 32 / 2, midx + 32 * 2.5, midy + 32 / 2, 1 ) ;
clearviewport() ;
setviewport ( 0, 0, maxx, maxy, 1 ) ;
welldone = NO ;
}

if ( bally > 180 - 12 )
{
welldone = NO ;
```

```
if ( ballx < paddlex - 20 || ballx > paddlex + 50 )
{
while ( bally < 177 )
{
putimage ( oldx, oldy, p1, XOR_PUT ) ;
putimage ( ballx, bally, p1, XOR_PUT ) ;
delay ( speed ) ;
oldx = ballx ;
oldy = bally ;
ballx = ballx + dx ;
bally = bally + dy ;
}
chance-- ;
score -= 20 ;
gotoxy ( 16, 25 ) ;
printf ( "%4d", score ) ;
music ( 2 ) ;
if ( chance )
putimage ( 515 + ( chance - 1 ) * 35 - 12 , maxy - 10, p1, XOR_PUT ) ;
if ( chance == 1 )
{
gotoxy ( 45, 25 ) ;
printf ( "Your last ball... Be careful!" ) ;
}
if ( !chance )
{
gotoxy ( 45, 25 ) ;
printf ( "Press any key...              " ) ;
outtextxy ( midx, midy, "I warned you! Try again" ) ;
music ( 2) ;
closegraph() ;
restorecrtmode() ;
exit ( 0 ) ;
}
}
music ( 4) ;
bally = 180 - 12 ;
dy = -dy ;  }
putimage ( oldx, oldy, p1, OR_PUT ) ;
putimage ( oldx, oldy, p1, XOR_PUT ) ;
```

```
putimage ( ballx, bally, p1, XOR_PUT ) ;

if ( score == 500 - ( ( 4 - chance ) * 20 ) )
{
outtextxy ( midx, midy, "You win !!!" ) ;
if ( score < 500 )
outtextxy ( midx, midy + 30, "Try scoring 500" ) ;
else
outtextxy ( midx, midy + 30, "You are simply GREAT!" ) ;
music ( 2) ;
closegraph() ;
restorecrtmode() ;
exit ( 0 ) ;
}
delay ( speed ) ;
if ( kbhit() )
{
ii.h.ah = 0 ;
int86 ( 22, &ii, &oo ) ;
putimage ( paddlex, paddley, p2, OR_PUT ) ;
putimage ( paddlex, paddley, p2, XOR_PUT ) ;
/* if Esc key has been pressed */
if ( oo.h.ah == 1 )
exit ( 0 ) ;
if ( oo.h.ah == 75 )
paddlex = paddlex - 20 ;
if ( oo.h.ah == 77 )
paddlex = paddlex + 20 ;
if ( paddlex < 0 )
paddlex = 0 ;
if ( paddlex > 589 )
paddlex = 589 ;
putimage ( paddlex, paddley, p2, XOR_PUT ) ;
}
}
}
mainscreen()
{
```

```
int ff[12][40] = {
1,1,1,1,0,0,0,1,1,1,1,0,0,0,1,1,1,1,1,0,0,0,1,1,1,0,0,1,0,0,0,0,1,0,0,0,1,1,1,1,0,
1,0,0,0,1,0,0,1,0,0,0,1,0,0,0,0,1,0,0,0,0,1,0,0,0,1,0,1,0,0,0,1,0,0,0,1,0,0,0,1,
1,0,0,0,0,1,0,1,0,0,0,0,1,0,0,0,1,0,0,0,1,0,0,0,0,0,0,1,0,0,1,0,0,0,1,0,0,0,0,0,
1,0,0,0,0,1,0,1,0,0,0,0,1,0,0,0,1,0,0,0,1,0,0,0,0,0,0,1,0,1,0,0,0,0,1,0,0,0,0,0,
1,0,0,0,1,0,0,1,0,0,0,0,1,0,0,0,0,1,0,0,0,1,0,0,0,0,0,0,1,1,0,0,0,0,0,1,0,0,0,0,0,
1,1,1,1,0,0,0,1,1,1,1,0,0,0,0,0,1,0,0,0,1,0,0,0,0,0,0,1,1,0,0,0,0,0,0,1,1,1,0,0,
1,0,0,0,1,0,0,1,0,0,0,1,0,0,0,0,0,1,0,0,0,1,0,0,0,0,0,0,1,1,0,0,0,0,0,0,0,0,0,1,0,
1,0,0,0,0,1,0,1,0,0,0,0,1,0,0,0,1,0,0,0,1,0,0,0,0,0,0,0,1,0,1,0,0,0,0,0,0,0,0,0,1,
1,0,0,0,0,1,0,1,0,0,0,0,1,0,0,0,1,0,0,0,1,0,0,0,0,0,0,0,1,0,0,1,0,0,0,0,0,0,0,0,1,
1,0,0,0,1,0,0,1,0,0,0,0,1,0,0,0,1,0,0,0,0,1,0,0,0,1,0,1,0,0,0,1,0,0,1,0,0,0,0,1,
1,1,1,1,0,0,0,1,0,0,0,0,1,0,1,1,1,1,1,0,0,0,1,1,1,0,0,1,0,0,0,0,1,0,0,1,1,1,1,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
} ;
int i, j, lx = 0, ly = 0, ch ;
setviewport ( 0, 0, maxx, maxy, 1 ) ;
rectangle ( 0, 0, maxx, maxy ) ;
for ( i = 0 ; i < 12 ; i++ )
{
for ( j = 0 ; j < 40 ; j++ )
{
if ( ff[i][j] )
rectangle ( lx, ly, lx + 15, ly + 9 ) ;
lx = lx + 16 ;
}
lx = 0 ;
ly = ly + 10 ;
}
line ( 0, maxy - 12, maxx, maxy - 12 ) ;
setfillstyle ( XHATCH_FILL, WHITE ) ;
floodfill ( 2, maxy - 2, WHITE ) ;
setfillstyle ( SOLID_FILL, WHITE ) ;
rectangle ( midx - 25, maxy - 7 - 12, midx + 25, maxy - 12 ) ;
floodfill ( midx, maxy - 1 - 12, 1 ) ;
circle ( midx, maxy - 13 - 12, 12 ) ;
floodfill ( midx, maxy - 10 - 12, 1 ) ;
outtextxy ( 15, 178, "Enter any key to continue" ) ;
music ( 1 ) ;
getch();
while ( 1 )
```

```
{
clearviewport();
outtextxy ( 50, 50, "Select any of the following:" ) ;
rectangle(40,45,275,60);
rectangle(40,85,85,100);
outtextxy ( 50, 90, "Play " ) ;
rectangle(40,105,150,120);
outtextxy ( 50, 110, "Instructions " ) ;
rectangle(40,125,130,140);
outtextxy ( 50, 130, "Credits " ) ;
rectangle(40,145,90,160);
outtextxy ( 50, 150, "Exit " ) ;
ch = 0 ;
while ( ! ( ch == 'E' || ch == 'I' || ch == 'P' || ch=='D') )
{
fflush ( stdin ) ;
if ( ( ch = getch() ) == 0 )
getch() ;
else
ch = toupper ( ch ) ;
}
if ( ch == 'P' )
break ;
switch ( ch )
{
case 'I' :
clearviewport() ;
setviewport ( 0, 0, maxx, maxy, 1 ) ;
settextstyle ( DEFAULT_FONT, HORIZ_DIR, 1 ) ;
outtextxy ( 50, 15, "        Instructions        " ) ;
settextstyle ( DEFAULT_FONT, HORIZ_DIR, 0 ) ;
outtextxy ( 50, 40, "Use left and right arrow keys to move paddle." ) ;
outtextxy ( 50, 60, "If you don't collect the ball on the paddle, you lose the ball." ) ;
outtextxy ( 50, 80, "On loosing a ball you loose 20 points." ) ;
outtextxy ( 50, 100, "On taking a brick you gain 5 points." ) ;
outtextxy ( 50, 180, "Press any key to go back" ) ;
fflush ( stdin ) ;
if ( getch() == 0 )
getch() ;
break ;
```

```
case 'D' :
clearviewport();
setviewport ( 0, 0, maxx, maxy, 1 ) ;
settextstyle ( DEFAULT_FONT, HORIZ_DIR, 1 ) ;
outtextxy ( 50, 15, "Developer" ) ;
settextstyle ( DEFAULT_FONT, HORIZ_DIR, 0 ) ;
outtextxy ( 50, 50, "Pravesh Gaire ") ;
outtextxy ( 50, 70, "073 / BEX / 427" ) ;
outtextxy ( 50, 180, "Press any key to go back" ) ;
fflush ( stdin ) ;
if ( getch() == 0 )
getch() ;
break ;
case 'E' :
closegraph() ;
restorecrtmode() ;
exit ( 0 ) ;
}
}
clearviewport() ;
setviewport ( 0, 0, maxx, maxy, 1 ) ;
outtextxy ( 50, 50, "Select any of the following levels:" ) ;
outtextxy ( 50, 70, "Novice" ) ;
outtextxy ( 50, 90, "Advanced" ) ;
outtextxy ( 50, 110, "Expert" ) ;
fflush ( stdin ) ;
if ( ( ch = getch() ) == 0 )
getch() ;
clearviewport() ;
return ( ch ) ;
}
void bricks()
{
int i, j, lx = 0, ly = 0 ;
for ( i = 0 ; i < 5 ; i++
{
for ( j = 0 ; j < 20 ; j++ )
 {
drawbrick ( lx, ly ) ;
```

```
lx = lx + 32 ;
}
lx = 0 ;
ly = ly + 10 ;
}
}
void drawbrick ( int lx, int ly )
{
rectangle ( lx, ly, lx + 31, ly + 9 ) ;
rectangle ( lx + 2, ly - 2, lx + 31 - 2, ly + 9 - 2 ) ;
floodfill ( lx + 1, ly + 1, 2 ) ;
}
void erasebrick ( int b, int l )
{
setcolor ( BLACK ) ;
rectangle ( b * 32, l * 10, ( b * 32 ) + 31 , ( l * 10 ) + 9 ) ;
rectangle ( b * 32 + 1, l * 10, ( b * 32 ) + 31 - 1, ( l * 10 ) + 9 - 1 ) ;
rectangle ( b * 32 + 2, l * 10, ( b * 32 ) + 31 - 2, ( l * 10 ) + 9 - 2 ) ;
setcolor ( WHITE ) ;
}
void music ( int type )
{
float octave[7] = { 130.81, 146.83, 164.81, 174.61, 196, 220, 246.94 } ;
int n, i ;
switch ( type )
{
case 1 :
for ( i = 0 ; i<100; i++ )
{
if(i==7) i=0;
sound ( octave[i] ) ;
delay ( 500 ) ;
if(kbhit()) break;
}
nosound() ;
break;
case 2 :
for ( i = 0 ; i < 15 ; i++ )
{
n = random ( 7 ) ;
```

```
sound ( octave[n] * 4 ) ;
delay ( 100 ) ;
}
nosound() ;
break ;
case 3:
for ( i = 4 ; i >= 0 ; i-- )
{
sound ( octave[i] * 4 ) ;
delay ( 15 ) ;
}
nosound() ;
break ;
case 4:
sound ( octave[6] * 2 ) ;
delay ( 50 ) ;
nosound() ;
}
}
void load()
{
int x=20;
moveto(200,160);
outtextxy("Loading, please wait...");
rectangle(10,maxy-100,maxx-10,maxy-75);
setcolor(WHITE);
while(x<maxx-20){
bar(20,maxy-95,x,maxy-80);
x++;
delay(10);
}
clearviewport();
}
```

OUTPUT

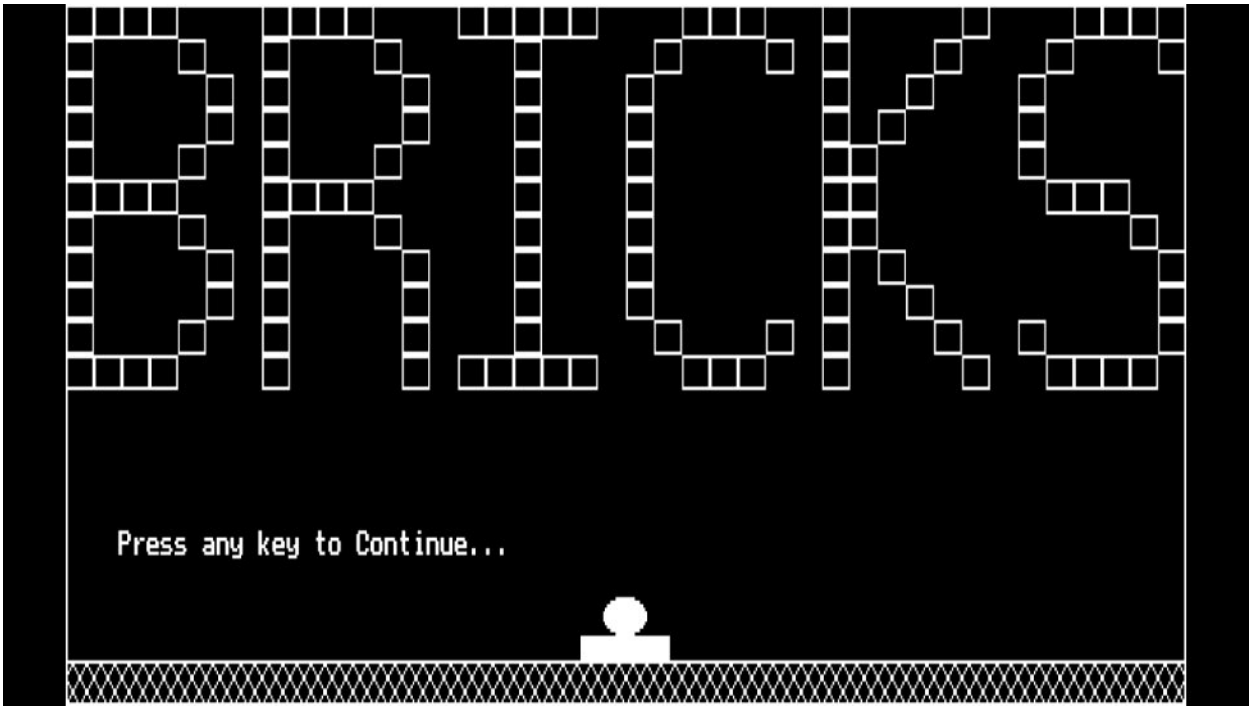


Fig: First Look

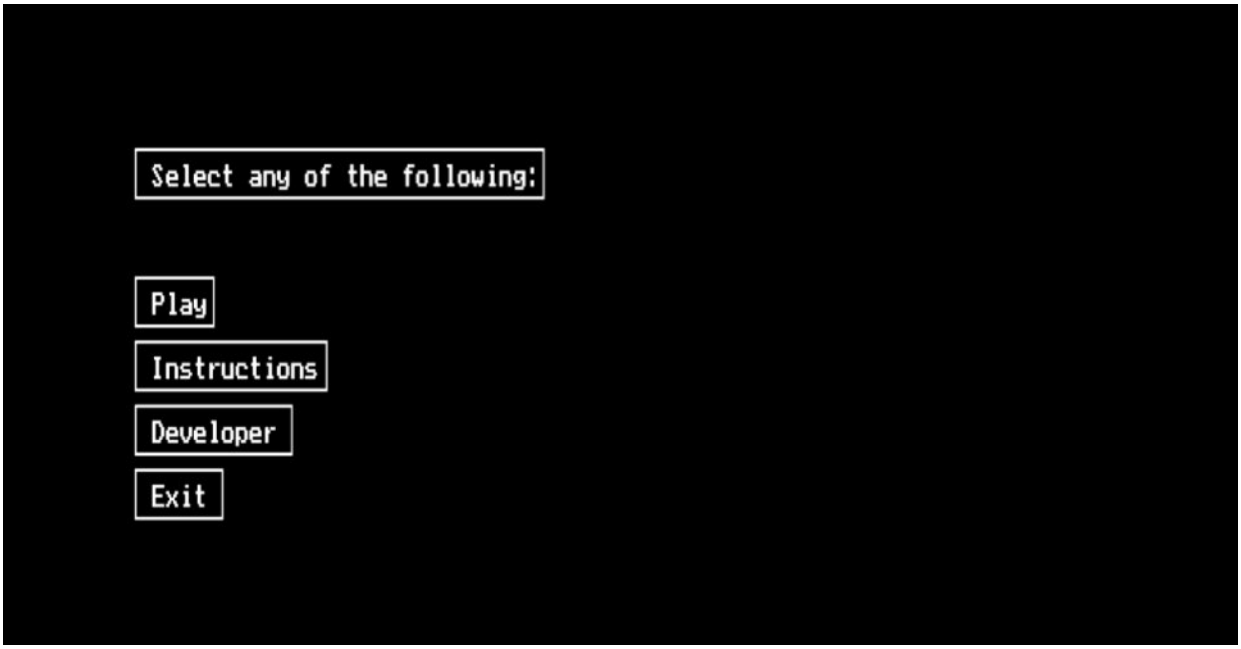


Fig: Menu

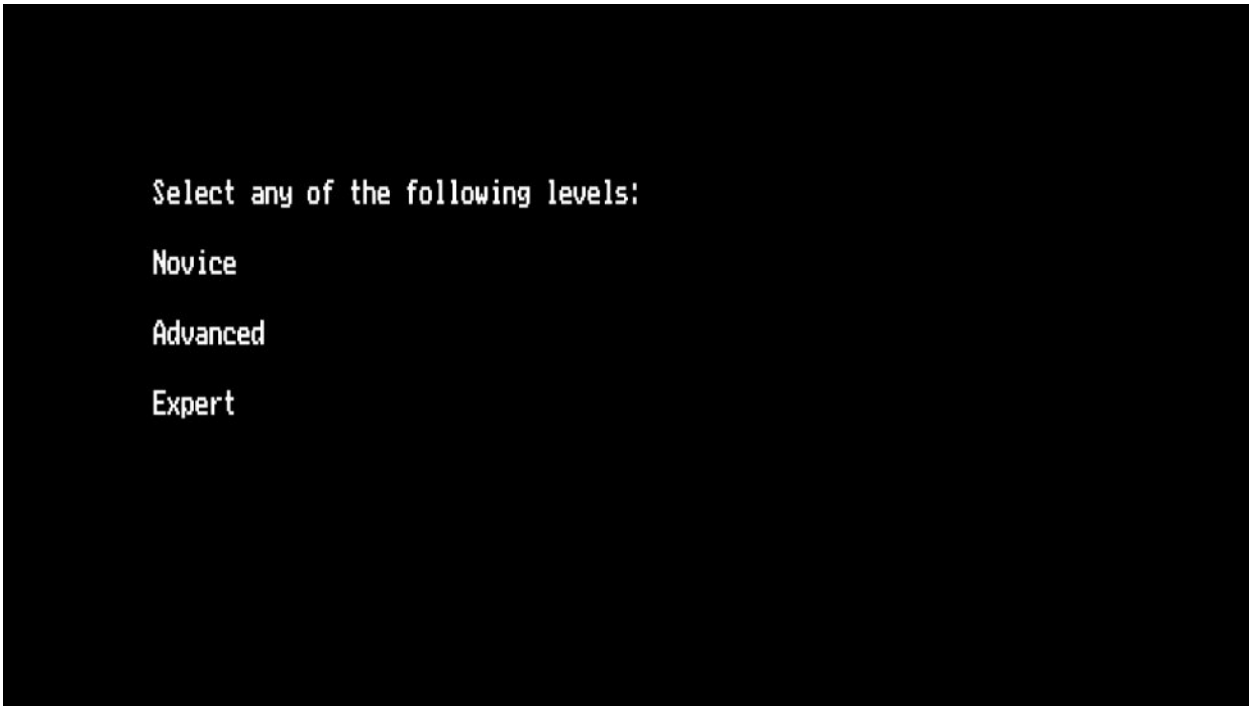


Fig: Level

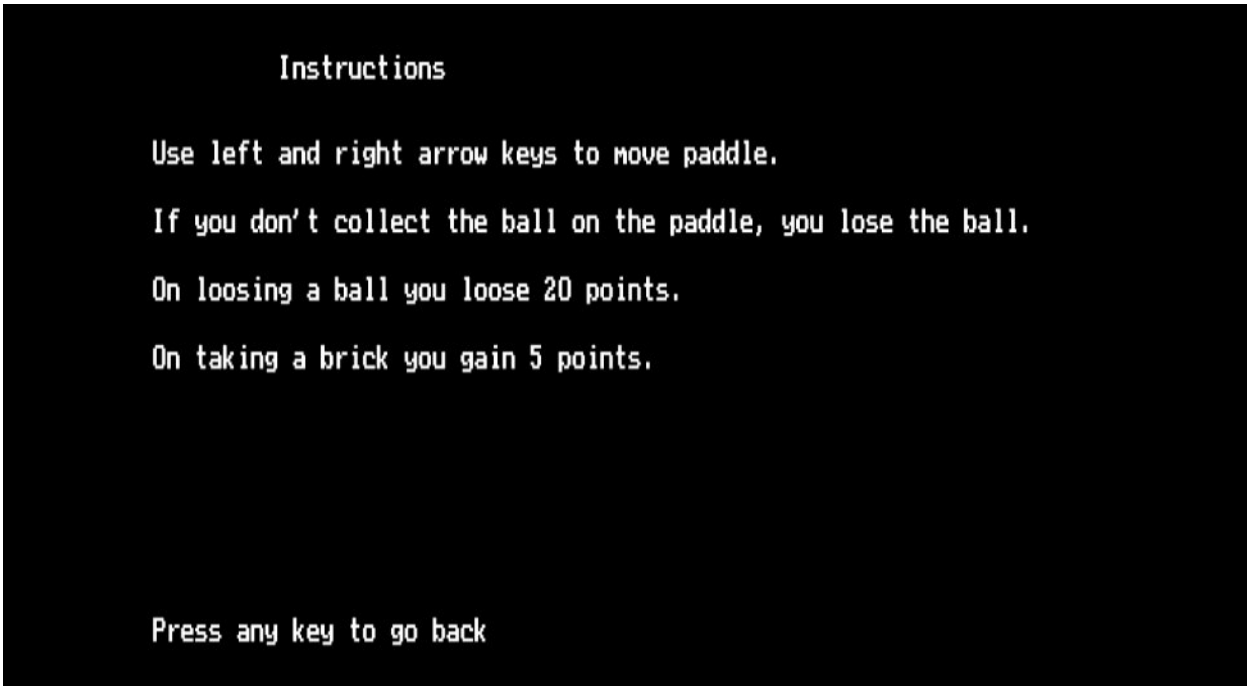


Fig: Instructions

Fig: About developer

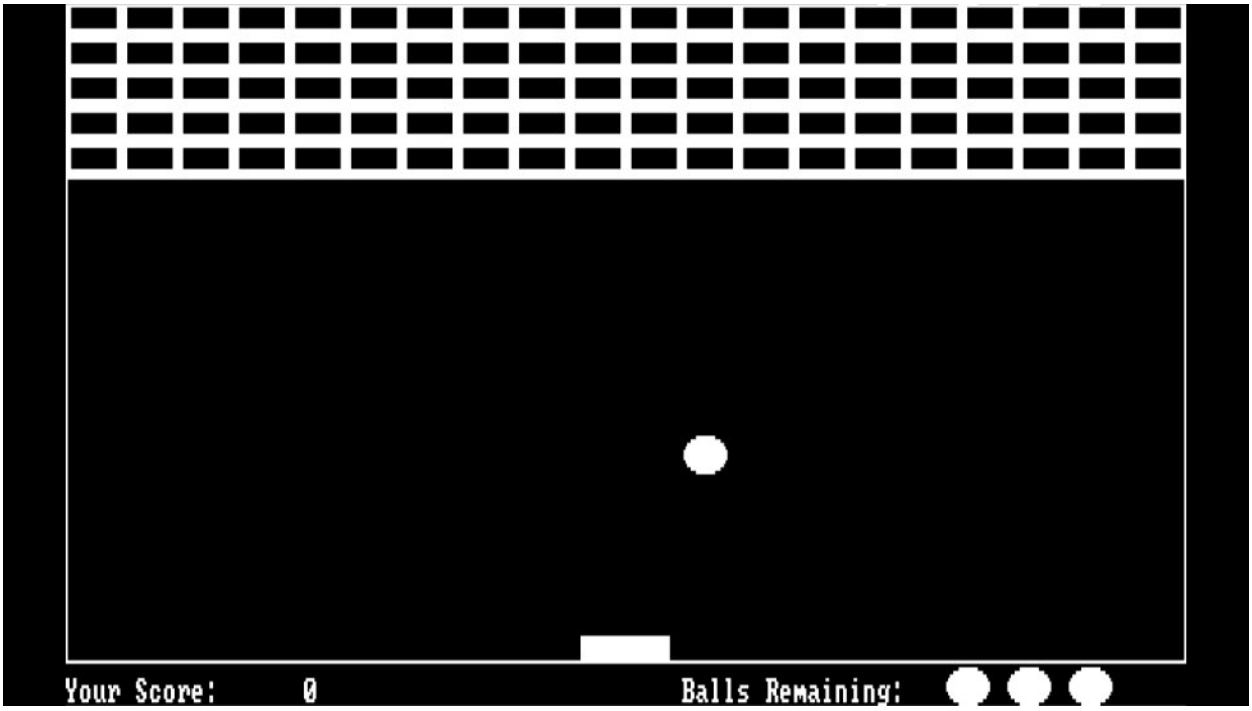


Fig: Game

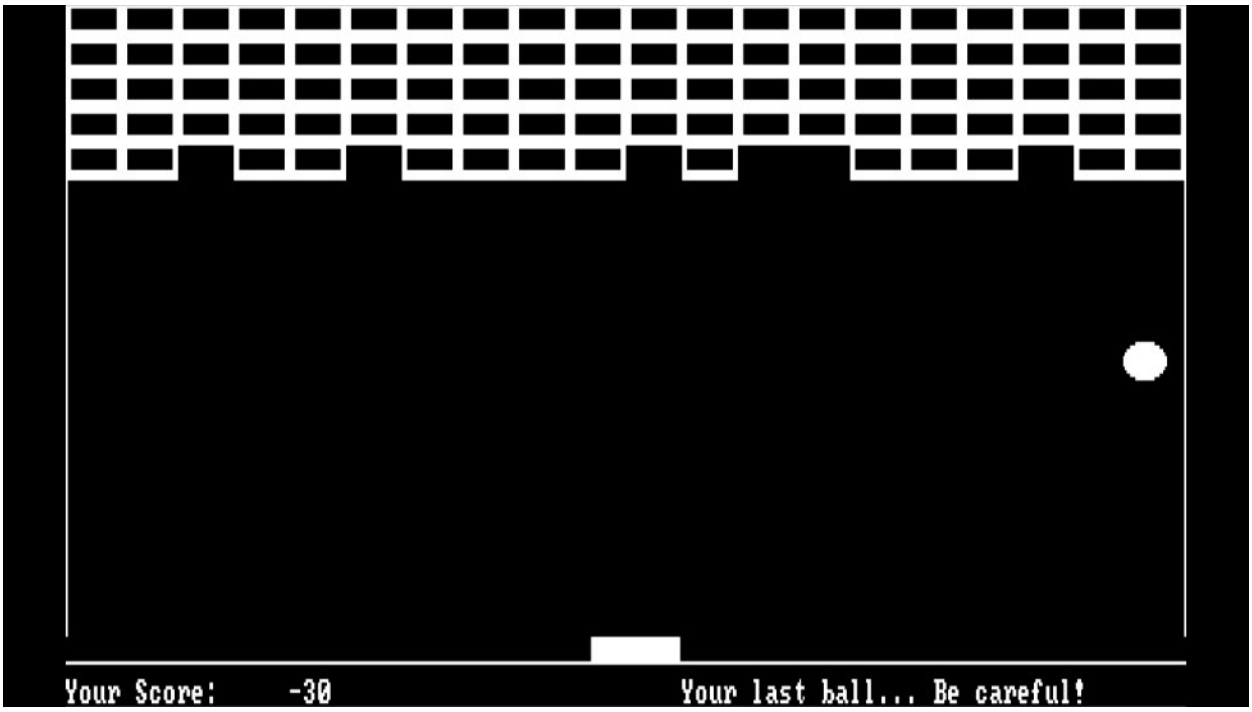


Fig: Game

DISCUSSION

As the course study we studied C programming language in the 1st sem BEX. No course is complete only with the theoretical knowledge. So with the same fact, we are given to submit the mini project on C programming that covers all the topic that we are studied during our study. With the same objective, we have made a game for mini-project.

The idea to game first came from my hobby as I play a lot of games and it was always a dream to make my own game once, I didnot expect that I would be successful making it while strating. But as I started making it it was quite easy learning syntax and many more user defined functions to make it. I was very satisfied with the work I did within short period.

The game source code is fully written in C language and includes stdio.h, dos.h, conio.h,stdlib.h and graphics.h as header's. graphic.h and dos.h being outside of syllabus was quite different at first but I used internet as reference to know how to use them. Unlike other header's graphics.h needs to be initialized and closed. The game consists five user defined functions. A fake loading is made to make game more attractive. Sound were also added in the game. Giving different sound different times was quite lengthy and unmanaged so I made a function named music and also different cases to make it easier. The making of bricks and erasing it was quite difficult at first. But what I did is made three functions, one would calculate the right position and send data to other function which used rectangle function to make bricks and the bricks which were stroke by ball were colored black by remaining function. Next challenge for me was to change the direction of ball when it hit any surface. So I changed the axis of ball movement when it stroke any surface so as to change direction perpendicularly. Game is made to be controlled by keyboard only.

The idea to game first came from my hobby as I play a lot of games and it was always a dream to make my own game once, I didnot expect that I would be successful making it while starting. But as I started making it it was quite easy learning syntax and many more user defined functions to make it. I was very satisfied with the work I did within short period. Being alone in my project I felt quite easy to do it as I neednot depend on my group members to make it and could make or modify it anytime what I thought was right Finally my mini-project for C was completed quite nicely.

SUMMARY

This mini-project is prepared in C-programming using simple logic available in high level language of this programming. This program, game is concerned with the entertainment purpose to the players. With the help of inbuilt functions of different header's we were able to make games extra user-friendly and attractive. The program also had user defined functions which were used to divide the program's function. We used arrays, pointers, union & function mainly focused in our course to complete the project.

There are seven views in the game. We made a fake loading too and sound were added when needed. User need to do all operations with keystrokes. The user have to destroy all the bricks with the ball. To make bricks we made two function bricks() and drawbrick() sends the right coordinates of bricks and draws the bricks at that place respectively. We also made erasebrick() function to erase the bricks when ball hit them. The movement of ball was simply made to change the axis when hit the boundaries or paddle. The ball movement can be still improved more. Mouse function could be added. Due to lack of time, knowledge and experience we couldnot make it flawless, still it has some drawbacks which can be cured to make the game more better. But we are sure that we did a descent work comparing to the course.

FURTHER ENHANCEMENT

- Extra graphics can be added to the game.
- Mouse function can also be added.
- Many more levels can be added.
- Information of players along with their score can be stored.
- The ball movement can be made more accurate according to the movement of paddle

REFERENCES

- Programming with C by Bryon S.Gottfried
- ANSI C, by Balaguruswamy

- Turbo C++ v 3.0 Help
- Borland C++ v 5.03 Help
- Internet