Разработать ООП для заполнения в случайном порядке окна (эмулятора) А-Ц консоли мозайкой из разноцветных пробелов или заглавных букв латинского алфавита. При этом консольная картина должна динамически меняться случайным образом в визуально различимом темпе так долго, как необходимо. Чтобы обеспечить требуемый видеоэффект, в программе должен быть реализован случайный выбор рядов и колонок окна консоли (24x80 по умолчанию). Для печати пробелов или случайных латинских букв со случайным фоном и цветом. Случайный процесс консольного вывода должен продолжаться до прерывания по сигналу от клавиатуры (^C) или до консольного ввода произвольного текста (соответствующая информация должна быть отображена в нижнем ряду окна консоли). В любом случае выполнение программы должно быть завершено и восстановлен стандартный видео режим с выводом промптера процессора команд OS в нижнем ряду окна консоли.

Программная реализация должна предусматривать разработку манипуляторов потока стандартного вывода для ESCAPE последовательностей управления курсором (CUP), установки псевдо-графического режима (SGR), гашения экрана (ED), и строки (EL). Весь манипулаторный код с соответствующим классом ESCAPE-потока (estream) для перегрузки оператора вывода и формирования ESCAPE-строк необходимо сосредоточить в консольном пространстве имен (con). Следует также специфицировать функции перехвата сигнала клавиатурного прерывания (SIGINT) и/или контроля очереди входного потока (kbin) без блокировки ввода (O_NONBLOCK) для корректного завершения программы. Случайная динамика консольного изображения должна быть реализована в основной функции программы с использованием перечисленных ESCAPE-манипуляторов в инструкциях обработки потоков консольного вывода и контроля клавиатурных сигналов с консольным вводом. Тип случайных элементов консольной мозайки выбирается средствами условной трансляции макроопределения RANDA (буквы), RANDB (фон пробелов или букв) и RANDF (цвет букв), которые могут быть установлены при компиляции программы (опциями -D), или задается соответствующими ключами –abf командой строки ее вызова.

```
001: #include <iostream>
002: #include <sstream>
003: #include <sys/ioctl.h>
004: #include <termios.h>
005: #include <fcntl.h>
006: #include <stdlib.h>
007: #include <signal.h>
008: #include <unistd.h>
009:
010: static int done = 0;
011: using namespace std;
012: namespace con
013: {
014:       int comax()
015:       {
016:             struct winsize w;
017:             ioctl(0, TIOCGWINSZ, &w);
018:             return(w.ws_col);
019:       }
020:
021:       int romax()
022:       {
023:             struct winsize w;
024:             ioctl(0, TIOCGWINSZ, &w);
025:             return(w.ws_row);
026:       }
027:
028:       ostream& ED(ostream& s)
029:       {
030:             return s << string("\033[2J");
031:       }
032:
033:       ostream& EL(ostream& s)
034:       {
035:             return s << string("\033[K");
036:       }
037:
038:       class estream
039:       {
040:             private:
041:                   string escape;
042:             public:
043:                   estream(string e) : escape(e) {};
044:                   friend ostream& operator<<(ostream&, estream);
045:       };
046:
047:       ostream& operator<<(ostream& s, estream e)
048:       {
049:             s << e.escape << flush;
050:             return s;
051:       }
052:
053:       estream CUP(int y, int x)
054:       {
055:             ostringstream sout;
056:             sout << "\033[" << y << ";" << x << "H";
057:             return estream(sout.str());
058:       }
059:
060:       estream SGR(int r)
061:       {
062:             ostringstream sout;
063:             sout << "\033[" << r << "m";
064:             return estream(sout.str());
065:       }
066: }
```

```
using con::SGR;
using con::CUP;

void interruptor(int signo)
{
     done = signo;
     return;
}

int kbin()
{
     char buf[512];
     int n=0;
     int flags = fcntl(0, F_GETFL);
     usleep(1);
     fcntl(0, F_SETFL, flags | O_NONBLOCK);
     n = read(0, buf, 512);
     fcntl(0, F_SETFL, flags /* & ~O_NONBLOCK */);
     return(n);
}

int parser(int n, char** v)
{
     int p=0;
     int opt;
     while((opt = getopt(n, v, "-baf")) != EOF)
     {
          switch(opt)
          {
               case 'a': p |= 1;
                    break;
               case 'f': p |= 2;
                    break;
               case 'b': p |= 4;
                    break;
               case '?': break;
          }
     }
     return(p);
}

int usage()
{
     cout << "Usage: E33 -afb" << endl;
     cout << "-a RANDOM ALPHA" << endl;
     cout << "-f RANDOM FOREGROUND" << endl;
     cout << "-b RANDOM BACKGROUND" << endl;
     return(0);
}

int main(int argc, char** argv)
{
     int m;
     if((argc = parser(argc, argv)) == 0)
          return(usage());
     cout << SGR(30+7); // white=7, foreground=30
     cout << SGR(40+4); // blue=4, background=40
     cout << SGR(1);    // bold=1

     cout << con::CUP(24, 1);
     cout << con::ED << "^C or Enter to exit" << flush;

     cout << con::CUP(23, 1);
     cout << con::romax() << "x" << con::comax() << flush;

     int x, y;
     char a;
     int f, b;
     srand(getpid());
     signal(SIGINT, interruptor);
     a = 32; b = (40 + 4); f = (30 + 7);
     while(done < 1)
     {
          x = rand() % (80 + 1);
          y = rand() % (24 - 1);
          if(argc & 1)
               a = 'A' + rand() % 26;
          if(argc == (2+1))
               f = 30 + rand() % 8;
          if(argc & 4)
               b = 40 + rand() % 8;
          cout << SGR(b) << CUP(y, x);
          cout << SGR(f) << a << flush;
          if(kbin() > 0)
               break;
     }

     cout << CUP(24, 1) << SGR(0) << con::EL;

/*   {
          int x, y;
          char a;
          struct termios t[2];
          tcgetattr(0, &t[0]);
          tcgetattr(0, &t[1]);
          t[0].c_lflag &= ~(ICANON | ECHO);
          tcsetattr(0, TCSAFLUSH, &t[0]);
          cout << "\033[512;512H" << flush;
          cout << "\033[6n" << flush;
          cin >> a >> a >> y >> a >> x >> a;
          cout << endl << "info:" << y << ';' << x << endl;
          tcsetattr(0, TCSAFLUSH, &t[1]);
     } */
     return(m);
}
```