

Курсова работа по ООП

На Стефан Христов Димитров

Втори курс ФИТА

КСТ 1Б

19621416



Условие на задачата

I. Да се състави клас за песни CSongs с член променливи: Заглавие на песента, изпълнител, клип (да/не), година на производство, жанр.

Съставете функции за:

1. за четене
2. за запис
3. функция за извеждане в изходен поток (отделно файл, отделно конзола)

Съставете конструктори:

1. подразбиращ се конструктор
2. копиращ конструктор
3. експлицитен конструктор

Съставете функции за:

1. За валидизиране на жанра
2. Изчисляване възрастта на песента
3. Извеждане данните за песента на принтер
4. Оператор за присвояване =
5. Логически оператор, който проверява дали песента има клип
6. Оператор за сравнение < (по давност).
7. Оператор за съвпадение (по изпълнител).

II. Да се състави клас CStudio, който съдържа CSongs с член променливи име на звукозаписното студио, брой албуми, брой приходи за една година.

Съставете следните функции:

1. Създаване на обект чрез друг обект
2. Създаване на обект чрез експлицитно зададени параметри
3. Функция, определяща дали албума се издаван повече от 1 път

III. Съставете главна програма, илюстрираща използването на създадените класове. Да се създаде контейнер вектор от 10 обекта от класа CStudio и да се направят следните справки, като:

1. извеждане на данните във файл, за звукозаписни студия с брой албуми над 20
2. да се изведат всички студия, които са издали по няколко албума на един и същи изпълнител
3. да се изведат заглавията на песните, които се повтарят в албумите на изпълнител, чието име е подадено като параметър
4. да се изведат всички данни за студията с уникални албуми (1 път издадени)
5. при подаден като параметър жанр, да се изведат студията, тиражиращи такива песни за песни, издадени преди 3 години

Описание на класове и функции

```
enum Genre{ pop, rock, metal, rap };  
class CSongs  
{  
protected:  
    string songname;  
    string singer;  
    string video;  
    int year;  
    Genre genre;
```

Създаване на класа **CSongs** в който има името на песента, името на изпълнителя, дали има видео, година на продуциране и вид жанр от тип enum.

```
CSongs()  
{  
    songname = " ";  
    singer = " ";  
    video = "No";  
    year = 0;  
    genre = pop;  
}  
CSongs(const CSongs &s)  
{  
    songname = s.songname;  
    singer = s.singer;  
    video = s.video;  
    year = s.year;  
    genre = s.genre;  
}
```

CSongs() е подразбиращ се конструктор който прави името на песента и на певеца да са празни, да няма видео песента, да няма година на създаване и жанра да е поп тъй като в този случай във enum поп е равно на 0.

CSongs(const CSongs &s) е копиращ конструктор който позволява да се създаде нов обект чрез друг.

```
CSongs(string name, string sing, string vid, int y, Genre g)
{
    songname = name;
    singer = sing;
    video = vid;
    year = y;
    genre = g;
}
```

Това е **експлицитен конструктор** за създаване на обекти при подаване на име на песента, име на изпълнител, видео, година и жанр.

```
void setname(string name)
{
    songname = name;
}
void setsinger(string sing)
{
    singer = sing;
}
void setvideo(string vid)
{
    video = vid;
}
void setyear(int y)
{
    year = y;
}
void setgenre(Genre g)
{
    genre = g;
}
```

Това са функции за **запис**. С тях лесно се променя определен елемент на обект без да трябва да се създава нов.

```
string getname()
{
    return songname;
}
string getsinger()
{
    return singer;
}
string setvideo()
{
    return video;
}
int getyear()
{
    return year;
}
Genre getgenre()
{
    return genre;
}
```

Функциите за **четене** са използвани за взимане на нужен елемент на обект от класа.

```

void save()
{
    ofstream file;
    file.open("savefile.txt");
    file << songname << " " << singer << " " << video << " " << year << " " << genre ;
    file.close();
}

friend ostream &operator <<(ostream& ostr, const CSongs& s)
{
    ostr << s.songname << " " << s.singer << " " << s.video << " " << s.year << " " << s.genre << endl;
    return ostr;
}

```

Функции за извеждане на обекти.

Функцията **save()** запазва обект във файла savefile.txt

Оператора << извежда обекта чрез изходен поток в конзолата

И двете функции извеждат името на песента, на изпълнителя, видео, година и жанр


```

void print()
{
    cout << "Song: " << songname << endl;
    cout << "By: " << singer << endl;
    cout << "Video: " << video << endl;
    cout << "Made in: " << year << endl;
    cout << "Genre: ";
    switch (genre)
    {
        case 0:
            cout << "pop" << endl;
            break;
        case 1:
            cout << "rock" << endl;
            break;
        case 2:
            cout << "metal" << endl;
            break;
        case 3:
            cout << "rap" << endl;
            break;
    }
}

```

Функцията **print()** извежда информацията за една песен на принтер. Тук за жанра е нужен switch тъй като genre връща стойност от 0 до 3 (броят въведени жанрове във enum Genre).

```
bool isGenre()  
{  
    if (genre == pop || genre == rock || genre == metal || genre == rap) return true;  
    else return false;  
}  
int calcage()  
{  
  
    return 2021-year;  
}
```

Булевата функция **isGenre()** проверява дали поставения жанр на обект съществува във enum Genre.

Функцията **calcage()** изчислява на колко години е дадена песен като изважда годината на издаване на песента от сегашната.

```
CSongs operator =(CSongs &s)
{
    this->songname = s.songname;
    this->singer = s.singer;
    this->video = s.video;
    this->year = s.year;
    this->genre = s.genre;
    return *this;
}
```

Operator = е функция за присвояване на елементите на един обект от друг. Тук трябва да се използва `this` тъй като този оператор трябва да върне някаква стойност.

```

bool operator <( CSongs &s)
{
    if (this->calcage() < s.calcage())return true;
    else return false;
}
bool operator==(const CSongs& s)
{
    if (singer == s.singer)return true;
    else return false;
}

```

Булеви функции за сравнение.

Operator < проверява дали давността на една песен е по-малка от давността на друга.

Operator == проверява дали две песни имат един и същ изпълнител

```

bool hasvid()
{
    if (video == "yes")return true;
    else return false;
}

```

Булевата функция **hasvid()** проверява дали песента има видео.

```

class CStudio : public CSongs
{
protected:
    vector<CSongs> songs;
    string studioname;
    int albums;
    int income;
public:
    CStudio(const CStudio &s)
    {
        songs = s.songs;
        studioname = s.studioname;
        albums = s.albums;
        income = s.income;
    }
    CStudio(vector<CSongs> a, string b, int c, int d)
    {
        songs = a;
        studioname = b;
        albums = c;
        income = d;
    }
}

```

Класа **CStudio** съдържа контейнер от песни, име на студиото, брой албуми и приход.

По-долу са **копирващия конструктор** и **експлицитния конструктор**

```

bool isReleased()
{
    if (albums > 1)
        return false;
    else return true;
}

```

Булевата функция **isReleased()** проверява дали студиото е издало повече от един албум

```

int getAlbums()
{
    return albums;
}
const vector<CSongs> getSongs()
{
    return songs;
}
string getstudio()
{
    return studioname;
}

```

Функции за четене на брой албуми, лист от песни и име на студио.

```

void savealbum()
{
    ofstream file;
    file.open("savefile.txt", ios::app);
    for (vector<CSongs>::iterator it = songs.begin(); it != songs.end(); it++)
    {
        file << *it;
    }
    file << studioname << " " << albums << " " << income << endl;
    file.close();
}

void printalbum()
{
    for (vector<CSongs>::iterator it = songs.begin(); it != songs.end(); it++)
    {
        (*it).print();
    }
    cout << "Studio: " << studioname << endl;
    cout << "Albums: " << albums << endl;
    cout << "Income: " << income << endl;
}

```

Функция **savealbum()** записва едно студио във savefile.txt.

Функцията **printalbum()** изпринтира информацията за едно студио в конзолата.

И двете функции използват итератор който обхожда контейнера с песни. В единия случай така се записва всяка песен във файла а в другия, всяка песен се изпринтира


```

void outputGenre(vector<CStudio> studio, Genre g)
{
    for (vector<CStudio>::iterator it = studio.begin(); it != studio.end(); it++)
    {
        vector<CSongs>vec = (*it).getSongs();
        for (vector<CSongs>::iterator it2 = vec.begin(); it2 != vec.end(); it2++)
        {
            if ((*it).getgenre() == g && (*it2).calcage() == 3)
            {
                (*it2).print();
                cout << "By studio: " << (*it).getstudio() << endl;
            }
        }
    }
}

```

Функцията **outputGenre()** приема контейнер от студия и жанр. Прави итератор на този контейнер, слага песните на итератора във вектор, обхожда новия вектор със втори итератор. После ако жанра на някои от песните съвпада с подадения и са били направени през 2018 годин (имат 3 години давност), ги изпринтирва заедно със студиото от което са издадени.

```

vector<CStudio> studios;
studios.reserve(10);
vector<CSongs> allsongs, songs1, songs2, songs3, songs4;
string name;
int choice;
int genrechoice;

```

Създаване и запълване на контейнери във главната функция за изпробване на функциите


```

for (vector<CStudio>::iterator it = studios.begin(); it != studios.end(); it++)
{
    if ((*it).getAlbums() > 20)
    {
        (*it).savealbum();
    }
}
cout << "Studios with over 20 albums have been saved to file" << endl;

```

Тук итераторът обхожда елементите в контейнера за студиа и ако техните издадени албуми надвишават 20 записва информацията им във файл

```

for (vector<CStudio>::iterator it = studios.begin(); it != studios.end(); it++)
{
    allsongs = (*it).getSongs();
    for (int i = 1; i < allsongs.size(); i++)
    {
        if ((*it).isReleased() == false && allsongs[i] == allsongs[i - 1])
        {
            cout << (*it).getstudio() << " have multiple albums of singer: " << allsongs[i-1].getsinger() << endl;
        }
        else if ((*it).isReleased() == false)
            cout << (*it).getstudio() << " have multiple albums of singer: " << allsongs[i].getsinger() << endl;
    }
}

```

Тук отново се обхожда чрез итератор контейнера за студиа като всеки път песните на едно студио се записват във вектора allsongs после се проверява дали студиото има повече от един албум и дали някои песни имат един и същ изпълнител. Извежда всички студиа които са издали повече от един албум на един и същи изпълнител

```

cout << "Input name of singer: ";
cin >> name;
for (vector<CStudio>::iterator it = studios.begin(); it != studios.end(); it++)
{
    allsongs = (*it).getSongs();
    for (int i = 1; i < allsongs.size(); i++)
    {
        if (allsongs[i].getsinger() == name && allsongs[i] == allsongs[i - 1])
        {
            if (allsongs[i].getname() == allsongs[i - 1].getname())
                cout << allsongs[i].getname() << endl;
        }
    }
}

```

Тук при въведен изпълнител се изпринтирват
всичките му песни чиито имена се повтарят

```

for (vector<CStudio>::iterator it = studios.begin(); it != studios.end(); it++)
    if ((*it).isReleased() == true)
    {
        (*it).printalbum();
    }

```

Тук се проверява дали едно студио е изкарало само един
албум и ако е така го изпринтирва.

```
cout << "Select genre: " << endl;
cout << "1.Pop" << endl;
cout << "2.Rock" << endl;
cout << "3.Metal" << endl;
cout << "4.Rap" << endl;
cout << "Input your choice: ";
cin >> genrechoice;
switch (genrechoice)
{
case 1:
    outputGenre(studios, pop);
    break;
case 2:
    outputGenre(studios, rock);
    break;
case 3:
    outputGenre(studios, metal);
    break;
case 4:
    outputGenre(studios, rap);
```

Тук при въведен жанр изкарва всички песни от този жанр които са излезли преди три години (използва по-горе описаната outputGenre())

Целия код

```
#include <iostream>
#include <string>
#include <fstream>
#include <vector>
using namespace std;
enum Genre{ pop, rock, metal, rap };
class CSongs
{
protected:
    string songname;
    string singer;
    string video;
    int year;
    Genre genre;
public:
    CSongs()
    {
```

```
songname = " ";
```

```
singer = " ";
```

```
video = "No";
```

```
year = 0;
```

```
genre = pop;
```

```
}
```

```
CSongs(const CSongs &s)
```

```
{
```

```
songname = s.songname;
```

```
singer = s.singer;
```

```
video = s.video;
```

```
year = s.year;
```

```
genre = s.genre;
```

```
}
```

```
CSongs(string name, string sing, string vid, int y, Genre  
g)
```

```
{
```

```
songname = name;
```

```
singer = sing;
```

```
video = vid;
```

```
year = y;
```

```
genre = g;
```

```
}
```

```
void setname(string name)
```

```
{
```

```
songname = name;
```

```
}
```

```
void setsinger(string sing)
```

```
{
```

```
singer = sing;
```

```
}
```

```
void setvideo(string vid)
```

```
{
```

```
video = vid;
```

```
}
```

```
void setyear(int y)
```

```
{
```

```
year = y;
```

```
}  
  
void setgenre(Genre g)  
{  
    genre = g;  
}  
  
string getname()  
{  
    return songname;  
}  
  
string getsinger()  
{  
    return singer;  
}  
  
string setvideo()  
{  
    return video;  
}  
  
int getyear()  
{
```

```
return year;
}

Genre getgenre()
{
return genre;
}

void save()
{
ofstream file;
file.open("savefile.txt");
file << songname << " " << singer << " " << video << " "
<< year << " " << genre ;
file.close();
}

friend ostream &operator <<(ostream& ostr, const
CSongs& s)
{
```



```
ostr << s.songname << " " << s.singer << " " << s.video  
<< " " << s.year << " " << s.genre << endl;
```

```
return ostr;
```

```
}
```

```
void print()
```

```
{
```

```
cout << "Song: " << songname << endl;
```

```
cout << "By: " << singer << endl;
```

```
cout << "Video: " << video << endl;
```

```
cout << "Made in: " << year << endl;
```

```
cout << "Genre: ";
```

```
switch (genre)
```

```
{
```

```
case 0:
```

```
cout << "pop" << endl;
```

```
break;
```

```
case 1:
```

```
cout << "rock" << endl;
```

```
break;
```

case 2:

cout << "metal" << endl;

break;

case 3:

cout << "rap" << endl;

break;

}

}

bool isGenre()

{

**if (genre == pop || genre == rock || genre == metal ||
genre == rap) return true;**

else return false;

}

int calcage()

{

return 2021-year;

}

```
CSongs operator =(CSongs &s)  
{  
    this->songname = s.songname;  
    this->singer = s.singer;  
    this->video = s.video;  
    this->year = s.year;  
    this->genre = s.genre;  
    return *this;  
}  
  
bool operator <( CSongs &s)  
{  
    if (this->calcage() < s.calcage())return true;  
    else return false;  
}  
  
bool operator==(const CSongs& s)  
{  
    if (singer == s.singer)return true;  
    else return false;  
}
```

```
bool hasvid()  
{  
    if (video == "yes")return true;  
    else return false;  
}
```

```
};  
class CStudio : public CSongs  
{  
protected:  
    vector<CSongs> songs;  
    string studioname;  
    int albums;  
    int income;  
public:  
    CStudio(const CStudio &s)  
    {  
        songs = s.songs;  
        studioname = s.studioname;
```

```
albums = s.albums;
income = s.income;
}
CStudio(vector<CSongs> a, string b, int c, int d)
{
songs = a;
studioName = b;
albums = c;
income = d;
}
bool isReleased()
{
if (albums > 1)
return false;
else return true;
}
int getAlbums()
{
return albums;
```

```
}  
  
const vector<CSongs> getSongs()  
{  
    return songs;  
}  
  
string getstudio()  
{  
    return studioname;  
}  
  
void savealbum()  
{  
    ofstream file;  
    file.open("savefile.txt", ios::app);  
    for (vector<CSongs>::iterator it = songs.begin(); it !=  
        songs.end(); it++)  
    {  
        file << *it;  
    }  
}
```

```
file << studioname << " " << albums << " " << income << endl;
```

```
file.close();
```

```
}
```

```
void printalbum()
```

```
{
```

```
for (vector<CSongs>::iterator it = songs.begin(); it != songs.end(); it++)
```

```
{
```

```
(*it).print();
```

```
}
```

```
cout << "Studio: " << studioname << endl;
```

```
cout << "Albums: " << albums << endl;
```

```
cout << "Income: " << income << endl;
```

```
}
```

```
};
```

```
void outputGenre(vector<CStudio> studio, Genre g)
```

```
{
```

```
for (vector<CStudio>::iterator it = studio.begin(); it !=
studio.end(); it++)
{
vector<CSongs>vec = (*it).getSongs();
for (vector<CSongs>::iterator it2 = vec.begin(); it2 !=
vec.end(); it2++)
{
if ((*it).getgenre() == g && (*it2).calcage() == 3)
{
(*it2).print();
cout << "By studio: " << (*it).getstudio() << endl;
}
}
}
}
```

```
void main()
{
vector<CStudio> studios;
```



```
studios.reserve(10);  
vector<CSongs> allsongs, songs1, songs2, songs3,  
songs4;  
string name;  
int choice;  
int genrechoice;  
CSongs song1("Drop it like it's hot", "Haarper", "no",  
2020, rap);  
CSongs song2("B.Zero", "Haarper", "yes", 2021, rap);  
CSongs song3("Welcome", "Beyonce", "yes", 2018,  
pop);  
songs1.push_back(song1);  
songs1.push_back(song2);  
songs1.push_back(song3);  
CStudio studio1(songs1, "VoidTracks", 20, 380000);  
studios.push_back(studio1);  
CSongs song4("Genesis", "Dua Lipa", "yes", 2017, pop);  
CSongs song5("Migrate", "Mariah Carey", "yes", 2008,  
pop);  
CSongs song6("Umbrella", "Rihanna", "no", 2008, pop);
```

```
songs2.push_back(song4);
songs2.push_back(song5);
songs2.push_back(song6);
CStudio studio2(songs2, "Pop Music Studios", 202,
900000);
studios.push_back(studio2);
CSongs song7("Dear Self", "Nine Lashes", "yes", 2021,
rock);
CSongs song8("Dear Self", "Nine Lashes", "no", 2020,
rock);
CSongs song9("T.N.T", "AC/DC", "no", 1975, rock);
songs3.push_back(song7);
songs3.push_back(song8);
songs3.push_back(song9);
CStudio studio3(songs3, "Montage Rock ", 30, 5000000);
studios.push_back(studio3);
CSongs song10("Voice of the Soul", "Death", "no",
1998, metal);
songs4.push_back(song10);
CStudio studio4(songs4, "Relapse Records", 1, 150000);
```

```
studios.push_back(studio4);
cout << "Options: " << endl;
cout << "1.Save studios with over 20 albums to file." <<
endl;
cout << "2.Print studios with more than one album by a
single singer." << endl;
cout << "3.Print songs by a inputed singer that are
identical." << endl;
cout << "4.Print studios with unique albums." << endl;
cout << "5.Print songs from inputed genre which were
created 3 years ago." << endl;
cout << "Input your choice: ";
cin >> choice;
switch (choice)
{

case 1:
// III.1
for (vector<CStudio>::iterator it = studios.begin(); it !=
studios.end(); it++)
```

```
{  
if ((*it).getAlbums() > 20)  
{  
(*it).savealbum();  
}  
}  
  
cout << "Studios with over 20 albums have been saved  
to file" << endl;  
  
break;  
  
case 2:  
  
// III.2
```

```
for (vector<CStudio>::iterator it = studios.begin(); it !=  
studios.end(); it++)  
{  
allsongs = (*it).getSongs();  
for (int i = 1; i < allsongs.size(); i++)  
{
```

```
if ((*it).isReleased() == false && allsongs[i] == allsongs[i  
- 1])
```

```
{
```

```
cout << (*it).getstudio() << " have multiple albums of  
singer: " << allsongs[i-1].getsinger() << endl;
```

```
}
```

```
else if ((*it).isReleased() == false)
```

```
cout << (*it).getstudio() << " have multiple albums of  
singer: " << allsongs[i].getsinger() << endl;
```

```
}
```

```
}
```

```
break;
```

```
case 3:
```

```
//III.3
```

```
cout << "Input name of singer: ";
```

```
cin >> name;
```

```
for (vector<CStudio>::iterator it = studios.begin(); it !=  
studios.end(); it++)
```

```

{
    allsongs = (*it).getSongs();
    for (int i = 1; i < allsongs.size(); i++)
    {
        if (allsongs[i].getsinger() == name && allsongs[i] ==
            allsongs[i - 1])
        {
            if (allsongs[i].getname() == allsongs[i - 1].getname())
                cout << allsongs[i].getname() << endl;
        }
    }
}

break;

case 4:
    //III.4

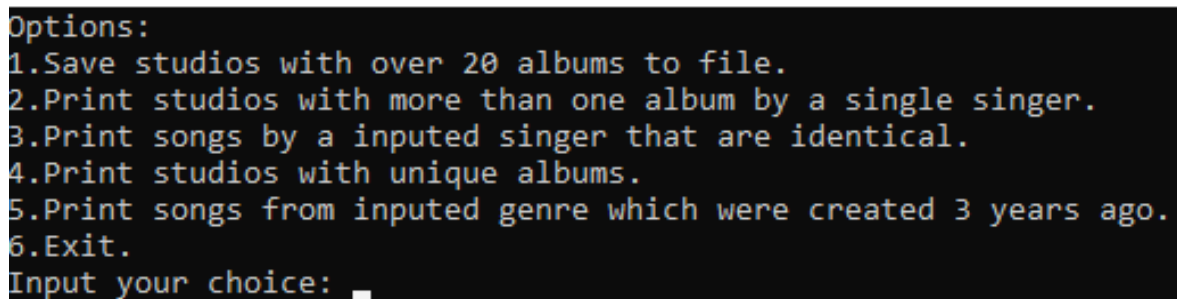
    for (vector<CStudio>::iterator it = studios.begin(); it !=
        studios.end(); it++)
    if ((*it).isReleased() == true)
    {

```

```
(*it).printalbum();  
}  
break;  
case 5:  
cout << "Select genre: " << endl;  
cout << "1.Pop" << endl;  
cout << "2.Rock" << endl;  
cout << "3.Metal" << endl;  
cout << "4.Rap" << endl;  
cout << "Input your choice: ";  
cin >> genrechoice;  
switch (genrechoice)  
{  
case 1:  
outputGenre(studios, pop);  
break;  
case 2:  
outputGenre(studios, rock);  
break;
```

```
case 3:
outputGenre(studios, metal);
break;
case 4:
outputGenre(studios, rap);
break;
}
}
}
```

Как работи програмата



```
Options:
1.Save studios with over 20 albums to file.
2.Print studios with more than one album by a single singer.
3.Print songs by a inputed singer that are identical.
4.Print studios with unique albums.
5.Print songs from inputed genre which were created 3 years ago.
6.Exit.
Input your choice: _
```

Главно меню. Въвежда се от едно до шест като опция шест е за излизане от програмата


```
Input your choice: 1
Studios with over 20 albums have been saved to file
Press any key to continue . . . █
```

При въведено **едно** за
избор всички студиа които
имат повече от двадесет
албума се запазват във
файла savefile.txt. Пример:

```
Genesis Dua Lipa yes 2017 0
Migrate Mariah Carey yes 2008 0
Umbrella Rihanna no 2008 0
Pop Music Studios 202 900000
Dear Self Nine Lashes yes 2021 1
Dear Self Nine Lashes no 2020 1
T.N.T AC/DC no 1975 1
Montage Rock 30 5000000
```

```
Input your choice: 2
VoidTracks have multiple albums of singer: Haarper
VoidTracks have multiple albums of singer: Beyonce
Pop Music Studios have multiple albums of singer: Mariah Carey
Pop Music Studios have multiple albums of singer: Rihanna
Montage Rock have multiple albums of singer: NineLashes
Montage Rock have multiple albums of singer: AC/DC
Press any key to continue . . .
```

При въведено **две** се изпринтирват на конзолата
студиата които имат множество албуми на дадени
изпълнители.

```
Input your choice: 3
Input name of singer: NineLashes
Dear Self
```

При въведено **три** програмата иска да се въведе име на изпълнител, след което изпринтирва песните които се повтарят на този изпълнител.

```
Input your choice: 4
Song: Voice of the Soul
By: Death
Video: no
Made in: 1998
Genre: metal
Studio: Relapse Records
Albums: 1
Income: 150000
```

При въведено **четири** на конзолата излизат всички студиа които са издали само един албум.

```
Input your choice: 5
Select genre:
1.Pop
2.Rock
3.Metal
4.Rap
Input your choice: 1
```

При въведено **пет** програмата изисква да се избере един от жанровете (от 1 до 4). И при въведен жанр изпринтирва песните които са издадени преди три години(2018) Пример:

```
Input your choice: 1
Song: Welcome
By: Beyonce
Video: yes
Made in: 2018
Genre: pop
By studio: VoidTracks
```