

SPRAWOZDANIE Z PROJEKTU NR 1

Wydział Informatyki Pracownia specjalistyczna Podstaw Programowania	Data:
Sprawozdanie z projektu nr1 Grupa 23 Magda Zaborowska Patryk Wójtowicz	Prowadzący: mgr inż. Tomasz Kuczyński Ocena:

Contents

TEMAT:	3
Programiŝterzy	3
WIZJA PROGRAMU	3
KOD PROGRAMU	4
Biblioteka f_wygladu.h	5
Biblioteka autorzy.h	6
Biblioteka dodaj_pytania.h	7
Biblioteka funkcje_kol_ratunkowych.h	12
Biblioteka ranking.h	16
Biblioteka zakoncz.h	20
Biblioteka menu.h	21
Biblioteka gra.h	23
Główna funkcja main()	34
int main()	34
JAK OSTATECZNIE WYGLADA GRA?	35

TEMAT:

Programisterzy

Gra w stylu teleturnieju „Milionerzy”. Gracz udziela odpowiedzi na kolejno zadawane pytania. Do każdego pytania jest kilka opcji odpowiedzi do wyboru. Kwota wygranej wzrasta w przypadku wyboru poprawnej odpowiedzi na pytanie. Do dyspozycji mamy kilka kół ratunkowych (np. 50 na 50, publiczność, telefon do przyjaciela). Pytania są odczytywane z pliku tekstowego. Program powinien umożliwiać dodawanie nowych pytań oraz zapis ich do pliku.

WIZJA PROGRAMU

Tworząc program chcieliśmy jak najlepiej odwzorować klimat Milionerów. Programisterzy to gra, która składa się z 12 rund. Przed rozpoczęciem turnieju program losuje 13 pytań z puli. Ostatnie jest zarezerwowane dla koła ratunkowego „Zamiana pytania”.

Użytkownikowi przysługują jeszcze dwa koła: 50 na 50 oraz telefon do przyjaciela. Każda poprawna odpowiedź stawia gracza o krok bliżej wygranej. Nasza gra zapewnia dwa gwarantowane progi: 1tys po drugiej rundzie i 40 tys to siódmej rundzie. Program daje również możliwość dopisywania pytań. Ciekawym urozmaicheniem jest ranking. Program wie, który wynik kwalifikuje się do zapisania w rankingu 10 najlepszych graczy. Dołożyliśmy wszelkich starań, aby program był intuicyjny i przyjazny dla oka.

KOD PROGRAMU

Poniżej przedstawimy napisane przez nas biblioteki, w których można zapoznać się z kluczowymi funkcjami i zmiennymi, oraz główną funkcję „int main()”. Biblioteki podzielone są tematycznie.

UWAGA!

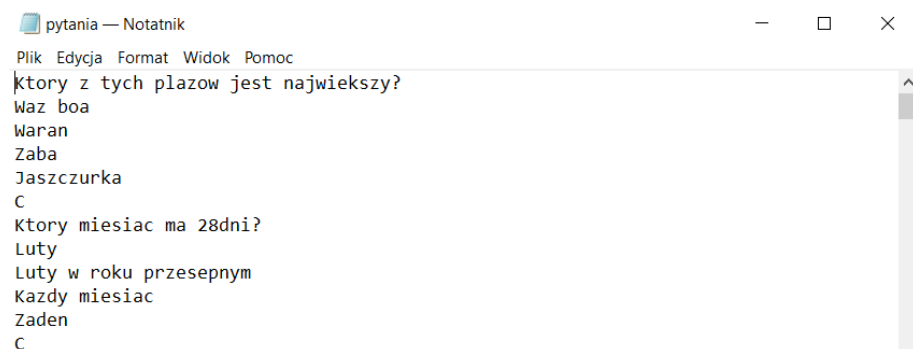
Plik z pulą pytań jest skonstruowany w sposób:

Wiersz I – pytanie;

Wiersze II-V – możliwe odpowiedzi;

Wiersz VI – znak poprawnej odpowiedzi;

Dlatego w kodzie, szukając bądź licząc pytania z pliku, dzielimy lub mnożymy przez 6.



```
pytania — Notatnik
Plik Edycja Format Widok Pomoc
który z tych plazow jest największy?
Waz boa
Waran
Zaba
Jaszczurka
C
Który miesiac ma 28dni?
Luty
Luty w roku przeseptym
Kazdy miesiac
Zaden
C
```

Biblioteka f_wygladu.h

Zmienna użyte w bibliotece		
SK	Globalna	Szerokość ramki
WK	Globalna	Wysokość ramki
LOGO	Globalna	Długość nazwy gry
logo	Lokalna (char)	Pod nią przypisana jest nazwa gry
i	Lokalna (int)	Zmienna do poruszania się w petli
j	Lokalna (int)	Zmienna do poruszania się w petli

```
#define SK 100
```

```
#define WK 5
```

```
#define LOGO 15
```

```
void wypisz_logo()
```

```
{
```

```
    char logo[LOGO]="PROGRAMISTERZY";
```

```
    for(int i=0;i<WK;i++)
```

```
    {
```

```
        for(int j=0;j<SK;j++)
```

```
        {
```

```
            if(i==0 || i==WK-1)
```

```
            {
```

```
                if(j!=SK-1) printf("*");
```

```
                else printf("*\n");
```

```
            }
```

```
            else if(i==(WK/2)&&((j>=SK/2-(LOGO/2-1))&&(j<=SK/2+(LOGO/2))))
```

```
            {
```

```
                printf("%c",logo[j-(SK/2-(LOGO/2-1))]);
```

```
            }
```

```
        }
    }
```

Commented [MZ1]: Funkcja wypisuje logo: tabela o wymiarach 100 x 5 złożoną z „*”, wewnątrz której widnieje nazwa gry.

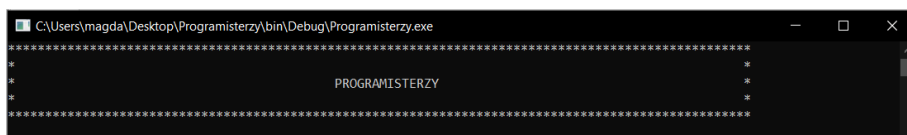
Commented [MZ2]: Pętla tworzy ramkę złożoną z gwiazdek

Commented [MZ3]: Wylicza środek ramki i wypisuje nazwę gry, tak aby była wyśrodkowana

```

    {
        if(j==0) printf("*");
        else if(j==SK-1) printf("*\n");
        else printf(" ");
    }
}
}
}

```



```
void czyszc()
```

```

{
    system("cls");
}

```

Commented [MZ4]: Funkcja czyści ekran

```
void pasek()
```

```

{
    for(int i=0;i<SK;i++)
    {
        printf("-");
        if(i==SK-1) printf("\n");
    }
}

```

Commented [MZ5]: Funkcja wypisuje pasek o długości 100,
W grze będzie on oddzielał poszczególne fragmenty

Biblioteka autorzy.h

```
void autorzy()
```

```

{

```

Commented [MZ6]: Funkcja czyści ekran, następnie wyświetla dane autorów programu

```

czyszc();
printf("AUTORZY:\n");
printf("1. Patryk Wojtowicz, student Politechniki Bialostockiej.\n");
printf("2. Magda Zaborowska, studentka Politechniki Bialostockiej.\n");
printf("\n");
printf("Wcisnij dowolny klawisz aby wrocic do menu kontekstowego...");
getch();
czyszc();
main();
}

```

Commented [MZ7]: Po wprowadzeniu dowolnego klawicza ekran czyści się i przechodzi do funkcji głównej

Biblioteka `dodaj_pytania.h`

Commented [MZ8]: Biblioteka zawiera wszystkie funkcje związane z dodawaniem pytania do puli pytań.

Zmienna użyte w bibliotece		
max_dlug	Globalna	W kodzie głównie wykorzystana do określenia długości ciągów znaków
komunikat1/2/3	Lokalna (char*)	Przypisano do nich instrukcje związane z dodawaniem pytania
pomocnicza	Lokalna (char*)	Chwilowo przypisujemy do niej wprowadzone przez użytkownika kolejno pytanie i odpowiedzi.
pytanie	Lokalna (char*)	Pytanie zawarte w „pomocnicza” trafia to zmiennej „pytanie”
odpA/B/C/D	Lokalna (char*)	Kolejne odpowiedzi zawarte w „pomocnicza” trafiają kolejno to zmiennej „odpA/B/C/D”
popodp	Lokalna (char*)	Do niej trafia znak poprawnej odpowiedzi (A-D) zapisanej wcześniej w „pomocnicza”
wybor	Lokalna (int)	Zapisuje numer opcji którą chce wybrać użytkownik

```
#define max_dlug 100
```

```

void dodaj_pytania()
{
    czysc();

    char komunikat1[max_dlug]="1. Maksymalna dlugosc pytania jak i odpowiedzi moze zawierac 100
    znakov. \n",

    komunikat2[max_dlug]="2. Poprawna odpowiedz powinna zawierac tylko i wylacznie literke
    odpowiedzi ktora jest prawidlowa.\n",

    komunikat3[max_dlug]="3. Kazde pytanie jak i odpowiedzi POTWIERDZ ENTEREM\n";

    for(int i=0;i<5;i++)
    {
        for(int j=0;j<100;j++)
        {
            if(i==0 || i==4)
            {
                if(j==99) printf("-\n");

                else printf("-"); //Nastepnie komunikaty (instrukcja), na koniec znów linia.
            }
            else
            {
                if(i==1) printf("%s",komunikat1);
                if(i==2) printf("%s",komunikat2);
                if(i==3) printf("%s",komunikat3);

                break;
            }
        }
    }

    char pomocnicza[max_dlug];
    printf("Podaj tresc pytania: "); fgets(pomocnicza,max_dlug,stdin);
    char pytanie[strlen(pomocnicza)];

    przypisz(pytanie,pomocnicza,strlen(pomocnicza)+1);

```

Commented [MZ9]: Dzięki funkcji użytkownik może dodać własne pytanie do puli.

Commented [MZ10]: Funkcja czyści ekran

Commented [MZ11]: W zerowym i czwartym wierszu wypisuje się linia.

Commented [MZ12]: Między liniami wypisują się komunikaty, czyli instrukcje dodawania pytania.

Commented [MZ13]: Do zmiennej pomocnicza pobieramy pytanie o długości max_dlug(100) znaków ze standardowego wejścia

Commented [MZ14]: Pod zmienną „pytanie” przypisujemy treść zmiennej „pomocnicza”. (Funkcja przypisz (...) opisana niżej)


```

printf("Podaj odpowiedz A: "); fgets(pomocnicza,max_dlug,stdin);
char odpA[strlen(pomocnicza)];
przypisz(odpA,pomocnicza,strlen(pomocnicza)+1);
printf("Podaj odpowiedz B: "); fgets(pomocnicza,max_dlug,stdin);
char odpB[strlen(pomocnicza)];
przypisz(odpB,pomocnicza,strlen(pomocnicza)+1);
printf("Podaj odpowiedz C: "); fgets(pomocnicza,max_dlug,stdin);
char odpC[strlen(pomocnicza)];
przypisz(odpC,pomocnicza,strlen(pomocnicza)+1);
printf("Podaj odpowiedz D: "); fgets(pomocnicza,max_dlug,stdin);
char odpD[strlen(pomocnicza)];
przypisz(odpD,pomocnicza,strlen(pomocnicza)+1);
printf("Podaj poprawna odpowiedz : "); fgets(pomocnicza,max_dlug,stdin);

while(!(((pomocnicza[0]>=65)&&(pomocnicza[0]<=68)))&&!((pomocnicza[0]>=97)&&(pomocnicza[0]
<=100))))
{
    printf("Nieprawidlowa odpowiedz! Odpowiedz powinna byc literkami a lub b lub c lub d!
Wprowadz poprawna odpowiedz: "); fgets(pomocnicza,max_dlug,stdin);
}

if((pomocnicza[0]>=97)&&(pomocnicza[0]<=100))
{
    pomocnicza[0]=pomocnicza[0]-32;
}

char popodp[strlen(pomocnicza)];
przypisz(popodp,pomocnicza,strlen(pomocnicza)+1);
FILE *plik=fopen("pytania","a");
if(plik==NULL)
{
    printf("Nie udalo sie otworzyc listy pytan do dopisywania! Powrot do menu!");
    main();
}

```

Commented [MZ15]: Analogicznie postępujemy pobierając do zmiennej „pomocnicza” odpowiedzi A,B,C,D i przypisując je odpowiednio do zmiennych „odpA/B/C/D”

Commented [MZ16]: Pobieramy znak poprawnej odpowiedzi.

Commented [MZ17]: W przypadku nieodpowiedniego znaku wyświetla się stosowny komunikat i ponowna próba pobrania znaku

Commented [MZ18]: Jeżeli użytkownik wpisał małą litera a-d, program zamienia ją na wielką A-D

Commented [MZ19]: Przypisanie znaku poprawnej odpowiedzi zawartej w „pomocnicza” do zmiennej „popodp”

Commented [MZ20]: Otwieramy plik z pulą pytań i odpowiedzi w trybie dopisywania

Commented [MZ21]: W przypadku błędu wyświetla stosowny komunikat i wraca do funkcji głównej

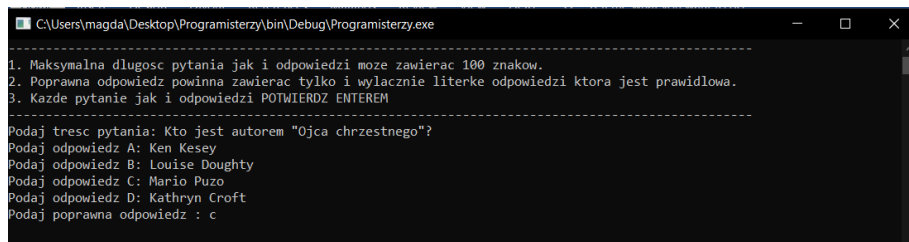
```

fprintf(plik,"%s",pytanie);
fprintf(plik,"%s",odpA);
fprintf(plik,"%s",odpB);
fprintf(plik,"%s",odpC);
fprintf(plik,"%s",odpD);
fprintf(plik,"%s",popodp);
fclose(plik);
menu_dodaj_pytania();
}

```

Commented [MZ22]: Do pliku dopisujemy pytanie, możliwe odpowiedzi i znak poprawnej odpowiedzi

Commented [MZ23]: Funkcja kieruje nas do następnej - menu dodawania pytania



```

C:\Users\magda\Desktop\Programistery\bin\Debug\Programistery.exe
-----
1. Maksymalna dlugosc pytania jak i odpowiedzi moze zawierac 100 znakow.
2. Poprawna odpowiedz powinna zawierac tylko i wytlacznie literke odpowiedzi ktora jest prawdziwa.
3. Kazde pytanie jak i odpowiedzi POTWIERDZ ENTEREM
-----
Podaj tresc pytania: Kto jest autorem "Ojca chrzestnego"?
Podaj odpowiedz A: Ken Kesey
Podaj odpowiedz B: Louise Doughty
Podaj odpowiedz C: Mario Puzo
Podaj odpowiedz D: Kathryn Croft
Podaj poprawna odpowiedz : c

```

```

void menu_dodaj_pytania()
{
    int wybor;
    czysc();
    printf("Pytanie wraz z odpowiedziami zostalo zapisane.\n");
    printf("1. Dodaj kolejne pytanie\n");
    printf("2. Powrot do menu\n");
    printf("3. Wylacz program\n");
    wybor=getch();
    switch(wybor)
    {
        case 49:
        {
            dodaj_pytania();
            break;

```

Commented [MZ24]: Funkcja wyświetla się po dodaniu pytania. Użytkownik ma do wyboru kilka opcji.

Commented [MZ25]: W przypadku wybrania „1” otwiera się funkcja dodawania pytania

```

    }
    case 50:
    {
        czysc();
        main();
        break;
    }
    case 51:
    {
        zakoncz(2);
        break;
    }
    default:
    {
        for(int i=3;i>0;i--)
        {
            czysc();
            printf("Podano nie prawidlowa wartosc! Program odswiezy sie za: %d \n",i);
            sleep(1);
        }
        czysc();
        menu_dodaj_pytania();
        break;
    }
}
}

```

Commented [MZ26]: W przypadku wybrania opcji „2” ekran czyści się i wraca do funkcji głównej.

Commented [MZ27]: W przypadku wybrania „3” otwiera się funkcja „zakoncz()” – opisana w bibliotece „zakoncz.h”

Commented [MZ28]: W przypadku niewłaściwego znaku wyświetla się stosowny komunikat. Ekran się czyści a funkcja menu_dodaj_pytania odzwieża się

```
void przypisz(char *napis1,char *napis2,int dl)
```

Commented [MZ29]: Funkcja, której argumentami są dwie zmienne typu char* oraz ich długość (int). Kopiuje zawartość drugiego napisu do zmiennej pierwszego napisu.

```

{
    for(int i=0;i<dl;i++)
    {
        napis1[i]=napis2[i];
    }
}

```

Biblioteka `funkcje_kol_ratunkowych.h`

Zmienne użyte w bibliotece		
nr1, nr2	Lokalne (int)	Zmienne na numery odpowiedzi w kodzie ASCII
nr_losowy	Lokalna (int)	Do zmiennej losujemy 0 lub 1, od tego zależy czy przyjaciel poda trafną odpowiedź
tresc_odp	Lokalna (char*)	Do zmiennej przypiszemy odpowiedź przyjaciela

```

int piedziesiat_na_piedziesiat(int nr,char pytanie[100],char odpA[100],char odpB[100],char odpC[100], char odpD[100], char popodp[2],char znak)

```

```

{
    czysc();
    printf("Wybrano koło ratunkowe 50 na 50!\n");
    pasek();
    printf("Pytanie nr %d\n",nr);
    pasek();
    printf("%s\n",pytanie);
    int nr1=0,nr2=0;
    nr1=rand()%4+65;
    while(nr1==popodp[0]) nr1=rand()%4+65;
    nr2=rand()%4+65;
    while(nr2==popodp[0] || nr2==nr1) nr2=rand()%4+65;
    if(popodp[0]==65 || (nr1!=65&&nr2!=65)) printf("A. %s\n",odpA);
    if(popodp[0]==66 || (nr1!=66&&nr2!=66)) printf("B. %s\n",odpB);
}

```

Commented [MZ30]: Biblioteka mieści funkcję do dwóch kół ratunkowych: 50 na 50 oraz telefon do przyjaciela.

Commented [MZ31]: Funkcja koła 50 na 50. Jej argumentami są:
Numer pytania(int), treść pytania(char*), cztery możliwe odpowiedzi (char*), znak poprawnej odpowiedzi (char*), zmienna char znak (do niej przypiszemy odpowiedź wybraną przez użytkownika)

Commented [MZ32]: Program czyści ekran, wypisuje nazwę koła ratunkowego, następnie treść pytania.

Commented [MZ33]: Losujemy odpowiedź, która jest błędna. (65-69 to w kodzie ASCII A-D) Odrzucimy ją w kole 50 na 50.

Commented [MZ34]: Losujemy drugą błędną odpowiedź. Nie może ona się pokryć z pierwszą błędną odpowiedzią.

```
if(popodp[0]==67 || (nr1!=67&&nr2!=67)) printf("C. %s\n",odpC);
```

```
if(popodp[0]==68 || (nr1!=68&&nr2!=68)) printf("D. %s\n",odpD);
```

```
pasek();
```

```
printf("Twój wybór: "); znak=getchar(); getchar();
```

```
while(!((znak>=65&&znak<=68)&&!(znak>=97&&znak<=100)) || ((znak==nr1 || znak==nr2) || (znak==nr1+32 || znak==nr2+32))))
```

```
{
```

```
printf("Wybrano odpowiedz z poza zakresu! Proszę sprobowac ponownie! Twój wybór: ");
```

```
znak=getchar(); getchar();
```

```
}
```

```
if((znak>=97)&&(znak<=100))
```

```
{
```

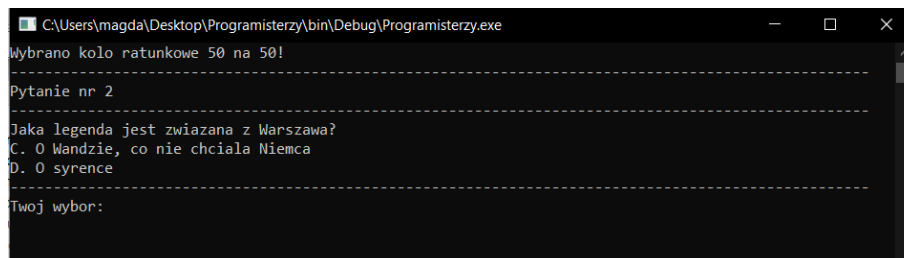
```
znak=znak-32;
```

```
}
```

```
printf("\n");
```

```
return znak;
```

```
}
```



```
void telefon_do_przyjaciela(char pytanie[100],char odpA[100],char odpB[100],char odpC[100], char odpD[100], char popodp[2])
```

```
{
```

```
printf("Za 3 sekundy polaczmy sie z twoim przyjacielem... \n");
```

```
sleep(3);
```

```
czyisc();
```

```
int nr_losowy;
```

Commented [MZ35]: Na ekran wypisuje te odpowiedzi które:
Jest poprawną odpowiedzią i tą która nie została wykluczona. Wykluczone odpowiedzi to te, które wylosowano do nr1, nr2.

Commented [MZ36]: Pobiera odpowiedz od gracza.

Commented [MZ37]: W przypadku błędnego znaku wypisuje stosowny komunikat i ponawia próbę pobrania odpowiedzi.

Commented [MZ38]: Jeżeli gracz wpisał małą literę, program zmienia ją na wielką (w kodzie ASCII)

Commented [MZ39]: Funkcja zwraca „znak” czyli odpowiedz wybraną przez gracza.

Commented [MZ40]: Funkcja telefonu do przyjaciela. Argumentami są: treść pytania, możliwe odpowiedzi i znak poprawnej odpowiedzi.
W naszej grze przyjaciel nie jest nieomylny. Mamy 50% szans że dobrze nam podpowie. Zależy to od zmiennej „nr_losowy”

```

char tresc_odp[100];
nr_losowy=rand()%2;
if(nr_losowy==0)
{
    int nr=rand()%4+65;
    while(nr==popodp[0]) nr=rand()%4+65;
    if(nr==65) strcpy(tresc_odp,odpA);
    if(nr==66) strcpy(tresc_odp,odpB);
    if(nr==67) strcpy(tresc_odp,odpC);
    if(nr==68) strcpy(tresc_odp,odpD);
}
else
{
    if(popodp[0]==65) strcpy(tresc_odp,odpA);
    if(popodp[0]==66) strcpy(tresc_odp,odpB);
    if(popodp[0]==67) strcpy(tresc_odp,odpC);
    if(popodp[0]==68) strcpy(tresc_odp,odpD);
}

printf("Wybrano kolo ratunkowe telefon do przyjaciela. Pamietaj ze masz 35 sekund.\n");
for(int i=0;i<113;i++)
{
    printf("- ");
    if(i==112) printf("\n");
}
sleep(3);
printf("TY:      | Czesc Kamil!\n"); sleep(1);
printf("PRZYJACIEL: | Czesc!\n"); sleep(1);
printf("TY:      | Sluchaj mam takie pytanie: %s\n",pytanie); sleep(3);
printf("TY:      | Mozliwe odpowiedzi to:\n"); sleep(1);
printf("TY:      | A. %s\n",odpA); sleep(1);
printf("TY:      | B. %s\n",odpB); sleep(1);

```

Commented [MZ41]: Losujemy do zmiennej 0 lub 1

Commented [MZ42]: Jeżeli wylosowano 0, odpowiedz przyjaciela będzie błędna.

Commented [MZ43]: Losujemy znak błędnej odpowiedzi.

Commented [MZ44]: Następnie do zmiennej „tresc_odp” przypisujemy treść wylosowanej błędnej odpowiedzi.

Commented [MZ45]: Jeżeli nr_losowy=1, przyjaciel podpowie trafnie.

Commented [MZ46]: Do zmiennej „tresc_odp” przypisujemy poprawną odpowiedz.

```

printf("TY:      | C. %s\n",odpC); sleep(1);
printf("TY:      | D. %s\n",odpD); sleep(1);
printf("TY:      | Ktora odpowiedz wedlug ciebie jest prawidlowa?\n"); sleep(3);
printf("PRZYJACIEL: | Rzeczywiscie padlo na trudne pytanie. Waham sie miedzy dwiema
odpowiedziami.\n"); sleep(3);
printf("PRZYJACIEL: | Jednak wydaje mi sie, ze poprawna odpowiedz brzmi: %s\n",tresc_odp);
sleep(3);
printf("PRZYJACIEL: | Trzymamy za Ciebie kciuki! Powodzenia!\n"); sleep(3);
for(int i=0;i<113;i++)
{
    printf("-");
    if(i==112) printf("\n\n");
}
sleep(8);
for(int i=5;i>0;i--)
{
    printf("Rozmowa zakonczy sie za %d\n",i);
    sleep(1);
}
}

```

Commented [MZ47]: Na ekranie wyświetla się rozmowa z przyjacielem, na końcu której pada odpowiedź przyjaciela. Trwa ona 45 sekund.

```

C:\Users\magda\Desktop\Programistery\bin\Debug\Programistery.exe
Wybrano kolo ratunkowe telefon do przyjaciela. Pamietaj ze masz 35 sekund.
-----
TY:      | Czesz Kamil!
PRZYJACIEL: | Czesz!
TY:      | Sluchaj mam takie pytanie: Jaki jest numer strazy pozarnej?
TY:      | Mozliwe odpowiedzi to:
TY:      | A. 999
TY:      | B. 997
TY:      | C. 998
TY:      | D. 996
TY:      | Ktora odpowiedz wedlug ciebie jest prawidlowa?
PRZYJACIEL: | Rzeczywiscie padlo na trudne pytanie. Waham sie miedzy dwiema odpowiedziami.
PRZYJACIEL: | Jednak wydaje mi sie, ze poprawna odpowiedz brzmi: 998
PRZYJACIEL: | Trzymamy za Ciebie kciuki! Powodzenia!
-----
Rozmowa zakonczy sie za 5

```

Biblioteka `ranking.h`

Zmienne użyte w bibliotece		
<code>tablica_dziesieciu_wynikow[10]</code>	Globalna (struct rank)	W niej zapisujemy dane z rankingu
<code>t</code>	Lokalna (time_t)	Do zmiennej zapisywany jest czas lokalny
<code>pom</code>	struct rank	Zmienna pomocnicza do sortowania tablicy struktur zawierającej dane rankingowe

Commented [MZ48]: Mieści funkcje odczytywania rankingu i modyfikowania jego.

```
struct rank
{
    char nazwa[20];
    int punkty;
    int rok;
    int miesiac;
    int dzien;
} tablica_dziesieciu_wynikow[10];
```

Commented [MZ49]: Struktura, dzięki której zapisujemy dane z rankingu, bądź wpisujemy dane do rankingu.

```
void ranking()
{
    czysc();
    pobierz_ranking(tablica_dziesieciu_wynikow);
    pasek();
    printf("TOP 10 Najlepszych zawodnikow Programisterow!!! \n");
    pasek(); printf("\n");
    for(int i=0;i<10;i++)
    {
        printf("%d. %s, wynik: %d, data: %d-%02d-%02d \n", i+1, tablica_dziesieciu_wynikow[i].nazwa,
        tablica_dziesieciu_wynikow[i].punkty, tablica_dziesieciu_wynikow[i].rok, tablica_dziesieciu_wynikow[i].miesiac, tablica_dziesieciu_wynikow[i].dzien);
    }
```

Commented [MZ50]: Kolejno pseudonim gracza, ilość zdobytych punktów i data.

Commented [MZ51]: Wypisuje ranking 10 najlepszych graczy

Commented [MZ52]: Pobiera ranking z pliku i zapisuje go do tablicy struktur. Funkcja `pobierz_ranking` opisana jest niżej.

Commented [MZ53]: Wypisuje nagłówek. Następnie wyświetla dziesięć pozycji z tablicy struktur, w której są dane pobrane z pliku zawierającego ranking.


```

printf("\n");
printf("Aby powrocic do glownego menu wcisnij dowolny klawisz...");
getch();
czyszc();
main();
}

```

```

C:\Users\magda\Desktop\Programistery\bin\Debug\Programistery.exe
-----
TOP 10 Najlepszych zawodnikow Programisterow!!!
-----
1. Patryk, wynik: 1000000, data: 2021-01-03
2. Madzia, wynik: 1000000, data: 2021-01-04
3. Magda, wynik: 20000, data: 2021-01-03
4. Byczek, wynik: 10000, data: 2021-01-08
5. Dominik, wynik: 5000, data: 2021-01-03
6. Marcin, wynik: 5000, data: 2021-01-08
7. milaczek, wynik: 2000, data: 2021-01-08
8. Marcin332, wynik: 1000, data: 2021-01-04
9. Piotr, wynik: 500, data: 2021-01-04
10. nikt, wynik: 0, data: 2000-09-27

Aby powrocic do glownego menu wcisnij dowolny klawisz...

```

Commented [MZ54]: Ranking wyłączamy dowolnym przyciskiem. Ekran się czyści i przechodzi do funkcji głównej.

```

void zapisz_wynik(int wynik)
{
    time_t t = time(NULL);
    struct tm tm = *localtime(&t);
    tablica_dziesieciu_wynikow[9].punkty=wynik;
    printf("\n");
    printf("Podaj swój pseudonim: "); scanf("%s",&tablica_dziesieciu_wynikow[9].nazwa);
    tablica_dziesieciu_wynikow[9].rok=tm.tm_year + 1900;
    tablica_dziesieciu_wynikow[9].miesiac=tm.tm_mon+1;
    tablica_dziesieciu_wynikow[9].dzien=tm.tm_mday;
    struct rank pom;
    for(int i=0;i<10;i++)
    {
        for(int j=0;j<9;j++)
        {
            if(tablica_dziesieciu_wynikow[j].punkty<tablica_dziesieciu_wynikow[j+1].punkty)

```

Commented [MZ55]: Funkcja zapisuje dane gracza do rankingu i sortuje go.

Commented [MZ56]: Pobiera do struktury czas systemowy.

Commented [MZ57]: Wynik gracza zapisuje na ostatniej pozycji w tablicy struktur.

Commented [MZ58]: Pobiera pseudonim gracza do tablicy struktur.

Commented [MZ59]: Zapisuje do tablicy dzień, miesiąc i rok czasu systemowego.

Commented [MZ60]: Sortowanie tablicy struktur względem zmiennej „tablica_dziesieciu_wynikow[j].punkty”. (Sort bąbelkowy)

```

{
    pom.punkty=tablica_dziesieciu_wynikow[j].punkty;
    strcpy(pom.nazwa,tablica_dziesieciu_wynikow[j].nazwa);
    pom.rok=tablica_dziesieciu_wynikow[j].rok;
    pom.miesiac=tablica_dziesieciu_wynikow[j].miesiac;
    pom.dzien=tablica_dziesieciu_wynikow[j].dzien;

    tablica_dziesieciu_wynikow[j].punkty=tablica_dziesieciu_wynikow[j+1].punkty;
    strcpy(tablica_dziesieciu_wynikow[j].nazwa,tablica_dziesieciu_wynikow[j+1].nazwa);
    tablica_dziesieciu_wynikow[j].rok=tablica_dziesieciu_wynikow[j+1].rok;
    tablica_dziesieciu_wynikow[j].miesiac=tablica_dziesieciu_wynikow[j+1].miesiac;
    tablica_dziesieciu_wynikow[j].dzien=tablica_dziesieciu_wynikow[j+1].dzien;

    tablica_dziesieciu_wynikow[j+1].punkty=pom.punkty;
    strcpy(tablica_dziesieciu_wynikow[j+1].nazwa,pom.nazwa);
    tablica_dziesieciu_wynikow[j+1].rok=pom.rok;
    tablica_dziesieciu_wynikow[j+1].miesiac=pom.miesiac;
    tablica_dziesieciu_wynikow[j+1].dzien=pom.dzien;
}
}
}

FILE *plik=fopen("ranking","w");
if(plik==NULL)
{
    printf("Nie udalo sie otworzyc rankingu do zapisu! Powrot do menu!");
    main();
}
for(int i=0;i<10;i++)
{
    fprintf(plik,"%s\n",tablica_dziesieciu_wynikow[i].nazwa);
    fprintf(plik,"%d\n",tablica_dziesieciu_wynikow[i].punkty);
}

```

Commented [MZ61]: Otwiera plik rankingowy w trybie wpisywania.

Commented [MZ62]: W przypadku błędu wyświetla stosowny komunikat i otwiera funkcję główną.

Commented [MZ63]: Tablica struktur jest posortowana. Jej zawartość wpisujemy do pliku.

```

    fprintf(plik,"%d ",tablica_dziesieciu_wynikow[i].rok);
    fprintf(plik,"%d ",tablica_dziesieciu_wynikow[i].miesiac);
    fprintf(plik,"%d\n",tablica_dziesieciu_wynikow[i].dzien);
}
fclose(plik);
}

```

```
int sprawdz_czy(int wynik)
```

```

{
    pobierz_ranking(tablica_dziesieciu_wynikow);
    if(wynik>tablica_dziesieciu_wynikow[9].punkty)
    {
        return true;
    }
    return false;
}

```

Commented [MZ64]: Funkcja sprawdza czy wynik użytkownika kwalifikuje się do wpisu do rankingu. Argumentem funkcji jest jedynie wynik gry użytkownika.

Commented [MZ65]: Pobiera ranking z pliku do tablicy struktur.

Commented [MZ66]: Sprawdza czy wynik gracza jest większy niż ostatnia pozycja rankingu.

```
void pobierz_ranking(struct rank* tablica_dziesieciu_wynikow)
```

```

{
    FILE *plik=fopen("ranking","r");
    char znak;
    if(plik==NULL)
    {
        printf("Nie udało się otworzyć rankingu do odczytu! Powrot do menu!");
        main();
    }
    for(int i=0;i<10;i++)

```

Commented [MZ67]: Pobiera dane z pliku rankingowego i umieszcza je w tablicy struktur.

Commented [MZ68]: Pobiera dziesięć pozycji z rankingu.

```

{
    fscanf(plik,"%s",tablica_dziesieciu_wynikow[i].nazwa); przesun_o_jedna_linie(znak,plik);
    fscanf(plik,"%d",&tablica_dziesieciu_wynikow[i].punkty); przesun_o_jedna_linie(znak,plik);
    fscanf(plik,"%d",&tablica_dziesieciu_wynikow[i].rok);
    fscanf(plik,"%d",&tablica_dziesieciu_wynikow[i].miesiac);
    fscanf(plik,"%d",&tablica_dziesieciu_wynikow[i].dzien); przesun_o_jedna_linie(znak,plik);
}
fclose(plik);
}

```

Biblioteka zakoncz.h

Zmienne użyte w bibliotece		
wybor	Lokalna (int)	Do zmiennej zapisuje się wybór gracza

```
void zakoncz(int podprogram)
```

```

{
    int wybor;
    czysc();
    printf("Czy na pewno chcesz zakonczyc program?\n");
    printf("1. TAK   ");
    printf("2. NIE  \n");
    wybor=getch();
    switch(wybor)
    {
    case 49:
    {
        exit(1);
        break;
    }
    }
}

```

Commented [MZ69]: Zawiera menu zakończenia programu. W zależności argumentu funkcji możemy powrócić do innych części programu.

Commented [MZ70]: Wypisuje menu zakończenia programu.

Commented [MZ71]: Jeżeli użytkownik wybrał opcję „TAK” - program kończy się.

case 50: //w zaleznosci od argumentu funkcji (int podprogram):

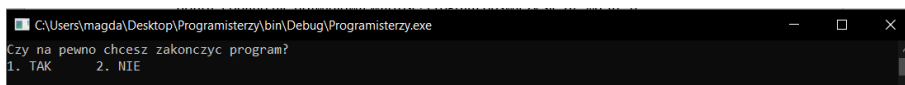
```
{
    czysc();
    if(podprogram==1) main();
    if(podprogram==2) menu_dodaj_pytania();
    if(podprogram==3) powrot();
    break;
}
```

default:

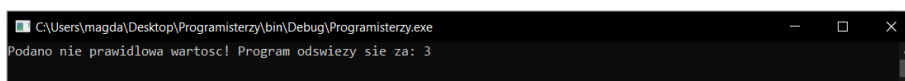
```
{
    for(int i=3;i>0;i--)
    {
        czysc();
        printf("Podano nie prawidlowa wartosc! Program odswiezy sie za: %d \n",i);
        sleep(1);
    }
    czysc();
    zakoncz(podprogram);
    break;
}
}
```

Commented [MZ72]: Jeżeli użytkownik wybrał opcję „NIE” – następuje powrót do innej części programu. Zależy to od argumentu funkcji („podprogram”)

Commented [MZ73]: W przypadku niewłaściwego wyboru, wyświetla stosowny komunikat, następnie funkcja „zakoncz()” odświeża się.



C:\Users\magda\Desktop\Programistery\bin\Debug\Programistery.exe
Czy na pewno chcesz zakonczyc program?
1. TAK 2. NIE



C:\Users\magda\Desktop\Programistery\bin\Debug\Programistery.exe
Podano nie prawidlowa wartosc! Program odswiezy sie za: 3

Biblioteka menu.h

Commented [MZ74]: Biblioteka zawiera funkcje z głównym menu gry.

Zmienne użyte w bibliotece		
wybor	Lokalna (int)	Do zmiennej zapisuje się wybór gracza

```
void wypisz_listewyborow()
```

```
{
    int wybor;
    printf("1. ZAGRAJ \n");
    printf("2. RANKING \n");
    printf("3. DODAJ WLASNE PYTANIE \n");
    printf("4. AUTORZY \n");
    printf("5. WYJSCIE \n\n");
    wybor=getch();
    switch(wybor)
    {
        case 49:
        {
            graj();
            break;
        }
        case 50:
        {
            ranking();
            break;
        }
        case 51:
        {
            dodaj_pytania();
            break;
        }
    }
}
```

Commented [MZ75]: Funkcja rozpoczynająca gre. W zależności od wyboru użytkownika przejdziemy do innej funkcji gry.

Commented [MZ76]: Wyświetla menu.

Commented [MZ77]: Pobiera wybór gracza.

Commented [MZ78]: Rozpoczyna grę. (funkcja opisana niżej w bibliotece gra.h)

Commented [MZ79]: Wyświetla ranking.

Commented [MZ80]: Otwiera okno dodawania pytania.

```

case 52:
{
    autorzy();
    break;
}
case 53:
{
    zakoncz(1);
    break;
}
default:
{
    for(int i=3;i>0;i--)
    {
        czysc();
        printf("Podano nie prawidlowa wartosc! Program odswiezy sie za: %d \n",i);
        sleep(1);
    }
    czysc();
    main();
    break;
}
}
}

```

Commented [MZ81]: Wypisuje autorów projektu.

Commented [MZ82]: Wypisuje menu zakończenia programu.

Commented [MZ83]: Wyświetla stosowny komentarz, powraca do funkcji głównej.

Biblioteka gra.h

Zmienne użyte w bibliotece		
max_znakow	Globalna	Określa długości pytań i odpowiedzi
ile_pytan	Globalna	Określa ilość pytań potrzebną do gry
wynik	Lokalna (int)	Liczy punkty użytkownika
ile	Lokalna (int)	Liczy wiersze z pliku z pytaniami

tabela_nr_pytan[ile_pytan]	Lokalna (int)	Tabela, do której program losuje numery pytan z pliku, które zostaną użyte w jednej konkretnej grze.
koło1/2/3	Lokalna (int)	Zmienne oznaczają czy koło ratunkowe zostało zużyte
znak	Lokalna (char)	Służy do sprawdzania kolejnych znaków z pliku z pytaniami oraz do pobrania odpowiedzi użytkownika
pytanie[max_znakow]	Lokalna (char*)	Do zmiennej zapisywane będą kolejne pytania podczas gry
odpA/B/C/D	Lokalna (char*)	Do zmiennych przypisane są możliwe odpowiedzi do konkretnego pytania
popodp	Lokalna (char*)	Zawiera znak poprawnej odpowiedzi

```
#include "funkcje_kol_ratunkowych.h"
```

```
#define max_znakow 100
```

```
#define ile_pytan 13
```

```
void graj()
```

```
{
    srand(time(NULL));

    int
    wynik=0,ile=0,nr=0,tabela_nr_pytan[ile_pytan]={0,0,0,0,0,0,0,0,0,0,0,0,0},koło1=1,koło2=1,koło3=1;

    char
    znak,pytanie[max_znakow],odpA[max_znakow],odpB[max_znakow],odpC[max_znakow],odpD[max_znakow],popodp[2];
```

```
FILE *plik=fopen("pytania", "r");
```

```
if(plik==NULL)
```

```
{
    printf("Nie udało się otworzyć listy pytań do odczytu! Powrót do menu!");
    main();
}
```

```
while(!feof(plik))
```

```
{
    znak=getc(plik);
```

Commented [MZ84]: Otwiera plik z pytaniami do odczytu.

Commented [MZ85]: W przypadku błędu wypisuje stosowny komentarz i wraca do funkcji głównej.

Commented [MZ86]: Zlicza wiersze w pliku z pytaniami.


```

        if(znak=="\n") ++ile;
    }
    fclose(plik);
    for(int i=0;i<ile_pytan;i++)
    {
        nr=rand()%(ile/6)+1;
        while(czy_bylo_to_pytan(nr,tabela_nr_pytan))
        {
            nr=rand()%(ile/6)+1;
        }
        tabela_nr_pytan[i]=nr;
    }
    for(int i=0;i<ile_pytan;i++)
    {
        czysc();
        printf("Pytanie nr %d\n",i+1);
        pasek();
        plik=fopen("pytania", "r");
        if(plik==NULL)
        {
            printf("Nie udało się otworzyć listy pytań do odczytu! Powrót do menu!");
            main();
        }
        for(int j=0;j<(tabela_nr_pytan[i]*6)-6;j++)
        {
            znak=getc(plik);
            while(znak!="\n")
            {
                znak=getc(plik);
            }
        }
    }

```

Commented [MZ87]: Z puli pytań wybiera 13 losowych numerów.
Do gry potrzebnych jest 12 pytań. Trzynaste zarezerwowane jest dla koła ratunkowego „zmiana pytania” (opisane niżej)

Commented [MZ88]: Losuje numer i sprawdza czy występuje on już w tabeli z numerami pytań (funkcja sprawdzająca opisana niżej). Jeżeli nie zapisuje go do tabeli.

Commented [MZ89]: Wyznacza kolejne rundy gry.

Commented [MZ90]: Otwiera plik z pytaniami do odczytu.

Commented [MZ91]: W tej petli opuszczamy w pliku kolejne wiersze do momentu gdy trafimy na numer pytania zawarty pod i-tym indeksem w tabeli "tabela_nr_pytan"

```

fscanf(plik, "%[^\n]", pytanie); przesun_o_jedna_linie(znak,plik);
fscanf(plik, "%[^\n]", odpA); przesun_o_jedna_linie(znak,plik);
fscanf(plik, "%[^\n]", odpB); przesun_o_jedna_linie(znak,plik);
fscanf(plik, "%[^\n]", odpC); przesun_o_jedna_linie(znak,plik);
fscanf(plik, "%[^\n]", odpD); przesun_o_jedna_linie(znak,plik);
fscanf(plik, "%[^\n]", popodp); przesun_o_jedna_linie(znak,plik);
fclose(plik);
printf("%s\n", pytanie);
printf("A. %s\n", odpA);
printf("B. %s\n", odpB);
printf("C. %s\n", odpC);
printf("D. %s\n", odpD);
pasek();
if(kolo1==1 || kolo2==1 || kolo3==1)
{
    printf("Dostepne kola ratunkowe to: \n");
    if(kolo1==1) printf("1. 50 na 50 \n");
    if(kolo2==1) printf("2. Zamiana pytania \n");
    if(kolo3==1) printf("3. Telefon do przyjaciela \n");
}
else
{
    printf("Brak dostepnych kol ratunkowych.\n");
}
pasek();
printf("Twój wybór: "); znak=getchar(); getchar();
while(!(znak>=49&&znak<=51)&&!(znak>=65&&znak<=68)&&!(znak>=97&&znak<=100))
{
    printf("Wybrano odpowiedz z poza zakresu! Prosze sprobować ponownie! Twój wybór: ");
    znak=getchar(); getchar();
}

```

Commented [MZ92]: Pobiera z pliku treść pytania, możliwych odpowiedzi i znak poprawnej odpowiedzi.

Commented [MZ93]: Wypisuje treść pytania i możliwe odpowiedzi.

Commented [MZ94]: Wyświetla dostępne koła ratunkowe.

Commented [MZ95]: W przypadku wykorzystania przez użytkownika wszystkich kół, wyświetla się komunikat.

Commented [MZ96]: Pobiera odpowiedź użytkownika.

Commented [MZ97]: Dopóki znak jest spoza a-d/A-D/1-3 – prosi o ponowny wybór.

```

    if((znak>=97)&&(znak<=100))
    {
        znak=znak-32;
    }

    printf("\n");
    if(znak>=49&&znak<=51)
    {
        if(znak==49)
        {
            if(kolo1==1)
            {
                znak=piedziesiat_na_piedziesiat(i+1,pytanie,odpA,odpB,odpC,odpD,popodp,znak);
                kolo1=0;
            }
            else
            {
                brak_dostepnego_kola();
                i--;
                continue;
            }
        }
        if(znak==50)
        {
            if(kolo2==1)
            {
                for(int i=3;i>0;i--)
                {
                    czyszc();
                    printf("Wybrano kolo ratunkowe zamiana pytania! Pytanie zostanie wylosowane za
%d...",i);
                    sleep(1);
                }
            }
        }
    }
}

```

Commented [MZ98]: Jeżeli użytkownik wprowadził małą literę, zmienia ją na wielką.

Commented [MZ99]: Jeżeli użytkownik wybrał „1” i koło jest dostępne, Otwiera się funkcja „piedziesiat_na_piedziesiat” a następnie koło zeruje się – stąd wiadomo, że zostało już użyte. Jeżeli koło jest niedostępne, otwiera się funkcja „brak_dostepnego_kola()”(funkcja opisana niżej)

Commented [MZ100]: Jeżeli wybrał „2” i koło jest dostępne:

Commented [MZ101]: Wyświetla komunikat.

```

    }
    tabela_nr_pytan[i]=tabela_nr_pytan[ile_pytan-1];
    kolo2=0;
    i--;
    continue;
}
else
{
    brak_dostepnego_kola();
    i--;
    continue;
}

}
if(znak==51)
{
    if(kolo3==1)
    {
        telefon_do_przyjaciela(pytanie,odpA,odpB,odpC,odpD,popodp);
        kolo3=0;
        i--;
        continue;
    }
    else
    {
        brak_dostepnego_kola();
        i--; //ponownie wyswietla sie to samo pytanie
        continue;
    }
}
}

```

Commented [MZ102]: Numer pytania w tabeli zmienia się na numer z ostatniej pozycji. Koło zeruje się.

Commented [MZ103]: Gwarantuje że ponownie wyświetli się i-te pytanie – tym razem zmienione.

Commented [MZ104]: Jeżeli koło nie jest dostępne, otwiera się funkcja „brak_dostepnego_kola()” a pytanie wyświetla się ponownie dzięki „i - - „

Commented [MZ105]: Jeżeli opcja „3” jest dostępna, otwiera się funkcja „telefon_do_przyjaciela()”. Koło zeruje się. Gdy rozmowa z przyjacielem się zakończy, wyświetla się ponownie te same pytania.

Commented [MZ106]: Analogicznie do poprzednich.

```

if(popodp[0]==znak)
{
    if(i==0) wynik=500;
    else if(i==3) wynik=5000;
    else if(i==7) wynik=75000;
    else if(i==8) wynik=125000;
    else wynik*=2;
    printf("Prawidlowa odpowiedz! Twój aktualny wynik to: %d \n",wynik);
    if(i==1)
    {
        pasek();
        printf("$$$ Gratulacje! Osiągnąłeś gwarantowany prog!
$$$ \n$$$ Od tego momentu w przypadku przegranej nie wychodzisz z pustymi rekami!
$$$ \n");
        pasek();
    }
    if(i==6)
    {
        pasek();
        printf("$$$ Gratulacje! Osiągnąłeś kolejny gwarantowany prog!
$$$ \n");
        pasek();
    }
    printf("Kontynuować? (Wcisnij dowolny klawisz)");
    getch();
    if(wynik==1000000)
    {
        koniec(wynik);
        break;
    }
}
else

```

Commented [MZ107]: W przypadku poprawnej odpowiedzi:

Commented [MZ108]: W zależności od poziomu gry, użytkownik zdobywa określoną liczbę punktów.

Commented [MZ109]: W przypadku osiągnięcia danego progu, użytkownik jest o tym informowany.

Commented [MZ110]: Aby przejść do kolejnego pytania użytkownik wybiera dowolny znak.

Commented [MZ111]: W przypadku poprawnej odpowiedzi i uzyskania miliona punktów otwiera się funkcja „koniec()” (funkcja opisana niżej)

```

{
    printf("Niestety nie byla to prawidlowa odpowiedz, poprawna to %s.\n",popodp);
    printf("Wcisnij dowolny klawisz aby kontynuowac..."); getch();
    koniec(wynik);
    break;
}
}
}

//funkcja oznajmia wygrana glownej nagrody, gwarantowanej kwoty lub przegranej nastepnie za
pomoca funkcji sprawdz_czy

// sprawdza czy wynik moze trafic do rankingu
void koniec(int wynik)
{
    int wybor;
    czysc();
    if(wynik==1000000)
    {
        printf("Gratulacje wygrales Programisterow! Zgarniasz 1,000,000 PESOS MEKSYKANSKICH!
\n\n");
    }
    else if(wynik<1000000&&wynik>=40000)
    {
        printf("Niestety przegrales! Zgarniasz jedynie 40,000 PESOS MEKSYKANSKICH! \n\n");
    }
    else if(wynik<40000&&wynik>=1000)
    {
        printf("Niestety przegrales! Zgarniasz jedynie 1,000 PESOS MEKSYKANSKICH! \n\n");
    }
    else
    {
        printf("Niestety przegrales! Nic nie zyskujesz! \n\n");
    }
}

```

Commented [MZ112]: W przypadku błędu wypisuje stosowny komentarz i po pobraniu dowolnego znaku otwiera funkcje „koniec()”

Commented [MZ113]: Funkcja kończąca grę. (samą grę – nie program). Argumentem jest ilość punktów zdobytych przez gracza.

Commented [MZ114]: W przypadku wygranej wypisuje stosowny komentarz.

Commented [MZ115]: Jeżeli wynik jest większy od drugiego progu 40tys. Gracz zdobywa jedynie gwarantowany próg.

Commented [MZ116]: Jeżeli gracz osiągnął tylko pierwszy próg i przegrał, zdobywa tylko 1tys.

Commented [MZ117]: Jeżeli gracz nie osiągnął żadnego progu, przegrywa bez nagrody.

```

if(sprawdz_czy(wynik))
{
    printf("Czy chcesz zapisac swoj wynik? \n");
    printf("1. Tak \n");
    printf("2. Nie \n");
    wybor=getch();
    switch(wybor)
    {
        case 49:
        {
            zapisz_wynik(wynik);
            break;
        }
        case 50:
        {
            powrot();
            break;
        }
        default:
        {
            for(int i=3;i>0;i--)
            {
                czysc();
                printf("Podano nie prawidlowa wartosc! Program odswiezy sie za: %d \n",i);
                sleep(1);
            }
            czysc();
            koniec(wynik);
            break;
        }
    }
}

```

Commented [MZ118]: Jeżeli wynik kwalifikuje się do rankingu, gracz może go zapisać.

Commented [MZ119]: W przypadku „1” otwiera menu zapisywania wyniku do rankingu.

Commented [MZ120]: W przypadku „2” wyświetla menu powrotu do funkcji głównej lub zakończenia programu.

Commented [MZ121]: W przypadku niewłaściwego znaku, odświeża funkcję „koniec()”

```
}  
powrot();  
}
```

```
int czy_bylo_to_pytanie(int nr,int tabela[ile_pytan])
```

```
{  
    int t=0;  
    for(int i=0;i<ile_pytan;i++)  
    {  
        if(tabela[i]==nr)  
        {  
            t=1;  
            break;  
        }  
    }  
    if(t==1) return true;  
    else return false;  
}
```

Commented [MZ122]: Sprawdza czy numer (nr) znajduje się już w tabeli.

```
void przesun_o_jedna_linie(char znak,FILE *plik)
```

```
{  
    znak=getc(plik);  
    while(znak!='\n')  
    {  
        znak=getc(plik);  
    }  
}
```

Commented [MZ124]: Funkcja przesuwa kursor o jeden wiersz dalej.

```
void powrot()
```

```
{  
    int wybor;
```

Commented [MZ125]: Wyświetla menu powrotu.


```

czyszc();
printf("1. Powrot do menu \n");
printf("2. Zakoncz program \n");
wybor=getch();
switch(wybor)
{
    case 49:
    {
        czyszc();
        main();
        break;
    }
    case 50:
    {
        zakoncz(3);
        break;
    }
    default:
    {
        for(int i=3;i>0;i--)
        {
            czyszc();
            printf("Podano nie prawidlowa wartosc! Program odswiezy sie za: %d \n",i);
            sleep(1);
        }
        czyszc();
        powrot();
        break;
    }
}
}

```

Commented [MZ126]: „1” – otwiera funkcję główną.

Commented [MZ127]: „2” – wyświetla menu zakończenia programu.

Commented [MZ128]: W przypadku błędnego znaku funkcja odświeży się.

```
void brak_dostepnego_kola()
```

```
{  
    for(int k=5;k>0;k--)  
    {  
        czysc();  
        printf("Wybrano zuzyte kolo ratunkowe! Pytanie sie odswiez za: %d",k);  
        sleep(1);  
    }  
}
```

Commented [MZ129]: Używana gdy gracz chce użyć koła, które jest niedostępne.

Commented [MZ130]: Wyświetla stosowny komunikat.

Główna funkcja main()

```
#include <stdio.h>  
#include <stdlib.h>  
#include <string.h>  
#include <time.h>  
#include <stdbool.h>  
#include "f_wygladu.h"  
#include "menu.h"  
#include "gra.h"  
#include "ranking.h"  
#include "dodaj_pytania.h"  
#include "autorzy.h"  
#include "zakonc.h"
```

```
int main()
```

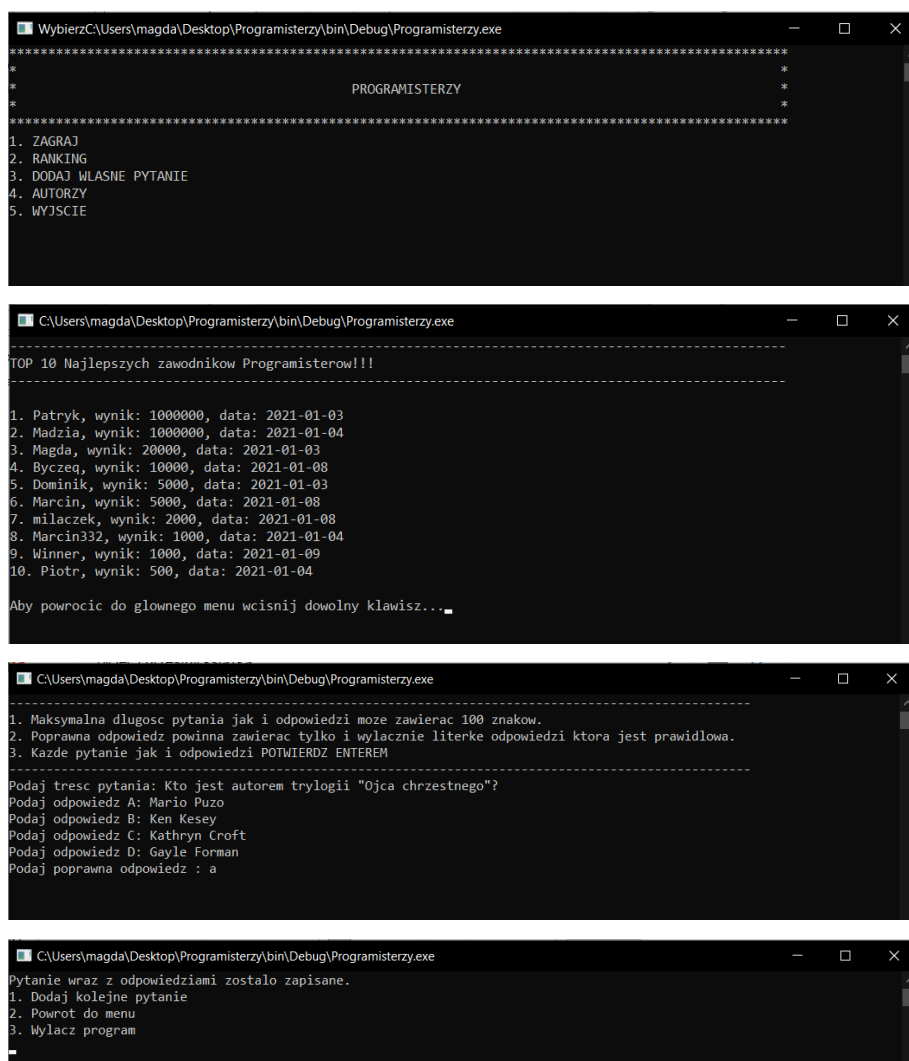
```
{  
    wypisz_logo();  
    wypisz_listewyborow();  
    return 0;  
}
```

Commented [MZ131]: Funkcja wypisuje logo gry.

Commented [MZ132]: Otwiera funkcje z głównym menu gry.

}

JAK OSTATECZNIE WYGLĄDA GRA?



```
WybierzC:\Users\magda\Desktop\Programisterzy\bin\Debug\Programisterzy.exe
*****
*                                     *
*                               PROGRAMISTERZY                               *
*                                     *
*****
1. ZAGRAJ
2. RANKING
3. DODAJ WLASNE PYTANIE
4. AUTORZY
5. WYJSCIE

C:\Users\magda\Desktop\Programisterzy\bin\Debug\Programisterzy.exe
-----
TOP 10 Najlepszych zawodnikow Programisterow!!!
-----
1. Patryk, wynik: 1000000, data: 2021-01-03
2. Madzia, wynik: 1000000, data: 2021-01-04
3. Magda, wynik: 20000, data: 2021-01-03
4. Byczek, wynik: 10000, data: 2021-01-08
5. Dominik, wynik: 5000, data: 2021-01-03
6. Marcin, wynik: 5000, data: 2021-01-08
7. milaczek, wynik: 2000, data: 2021-01-08
8. Marcin332, wynik: 1000, data: 2021-01-04
9. Winner, wynik: 1000, data: 2021-01-09
10. Piotr, wynik: 500, data: 2021-01-04

Aby powrocic do glownego menu wcisnij dowolny klawisz...

C:\Users\magda\Desktop\Programisterzy\bin\Debug\Programisterzy.exe
-----
1. Maksymalna dlugosc pytania jak i odpowiedzi moze zawierac 100 znakow.
2. Poprawna odpowiedz powinna zawierac tylko i wylaczenie literke odpowiedzi ktora jest prawdziwa.
3. Kazde pytanie jak i odpowiedzi POTWIERDZ ENTEREM
-----
Podaj tresc pytania: Kto jest autorem trylogii "Ojca chrzestnego"?
Podaj odpowiedz A: Mario Puzo
Podaj odpowiedz B: Ken Kesey
Podaj odpowiedz C: Kathryn Croft
Podaj odpowiedz D: Gayle Forman
Podaj poprawna odpowiedz : a

C:\Users\magda\Desktop\Programisterzy\bin\Debug\Programisterzy.exe
Pytanie wraz z odpowiedziami zostalo zapisane.
1. Dodaj kolejne pytanie
2. Powrot do menu
3. Wylacz program
_
```

```
C:\Users\magda\Desktop\Programisterzy\bin\Debug\Programisterzy.exe
AUTORZY:
1. Patryk Wojtowicz, student Politechniki Bialostockiej.
2. Magda Zaborowska, studentka Politechniki Bialostockiej.

Wcisnij dowolny klawisz aby wrocic do menu kontekstowego...
```

```
C:\Users\magda\Desktop\Programisterzy\bin\Debug\Programisterzy.exe
Pytanie nr 1
-----
Jaki ptak ma oczy osadzone frontalnie i otoczone promieniscie ulozonymi piorami?
A. Brzegowka
B. Dymowka
C. Plomkowka
D. Bogatka
-----
Dostepne kola ratunkowe to:
1. 50 na 50
2. Zamiana pytania
3. Telefon do przyjaciela
-----
Twój wybór: 
```

```
C:\Users\magda\Desktop\Programisterzy\bin\Debug\Programisterzy.exe
Wybrano kolo ratunkowe 50 na 50!
Pytanie nr 1
-----
Jaki ptak ma oczy osadzone frontalnie i otoczone promieniscie ulozonymi piorami?
B. Dymowka
D. Bogatka
-----
Twój wybór: d
Prawidlowa odpowiedz! Twój aktualny wynik to: 500
Kontynuowac? (Wcisnij dowolny klawisz)
```

```
C:\Users\magda\Desktop\Programisterzy\bin\Debug\Programisterzy.exe
Pytanie nr 2
-----
Ile jest znakow zodiaku?
A. 12
B. 15
C. 16
D. 10
-----
Dostepne kola ratunkowe to:
2. Zamiana pytania
3. Telefon do przyjaciela
-----
Twój wybór: a
Prawidlowa odpowiedz! Twój aktualny wynik to: 1000
$$$ Gratulacje! Osiagnoles gwarantowany prog! $$$
$$$ Od tego momentu w przypadku przegranej nie wychodzisz z pustymi rekami! $$$
-----
Kontynuowac? (Wcisnij dowolny klawisz)
```

```
C:\Users\magda\Desktop\Programisterzy\bin\Debug\Programisterzy.exe
Pytanie nr 7
-----
Malarz miał namalować na drzwiach numery od 1 do 100. Ile dziewczątek musiał namalować?
A. 1
B. 11
C. 15
D. 20
-----
Dostępne kółka ratunkowe to:
3. Telefon do przyjaciela
-----
Twój wybór: d

Prawidłowa odpowiedź! Twój aktualny wynik to: 40000
$$$ Gratulacje! Osiągnąłeś kolejny gwarantowany prog! $$$
Kontynuować? (Wcisnij dowolny klawisz)
```

```
C:\Users\magda\Desktop\Programisterzy\bin\Debug\Programisterzy.exe
Niestety przegrałeś! Zgarniasz jedynie 40,000 PESOS MEKSYKAŃSKICH!

Czy chcesz zapisać swój wynik?
1. Tak
2. Nie
_
```

```
C:\Users\magda\Desktop\Programisterzy\bin\Debug\Programisterzy.exe
Niestety przegrałeś! Zgarniasz jedynie 40,000 PESOS MEKSYKAŃSKICH!

Czy chcesz zapisać swój wynik?
1. Tak
2. Nie

Podaj swój pseudonim: Pepe pan dziobak
```

```
C:\Users\magda\Desktop\Programisterzy\bin\Debug\Programisterzy.exe
Wybrano kółko ratunkowe telefon do przyjaciela. Pamiętaj że masz 35 sekund.
-----
TY: | Cześć Kamil!
PRZYJACIEL: | Cześć!
TY: | Słuchaj mam takie pytanie: Jaka część mowy odpowiada na pytania: kto, co?
TY: | Możliwe odpowiedzi to:
TY: | A. Przysłów
TY: | B. Czasownik
TY: | C. Rzeczownik
TY: | D. Przysłówek
TY: | Która odpowiedź według ciebie jest prawidłowa?
PRZYJACIEL: | Rzeczywiście padło na trudne pytanie. Wahać się między dwiema odpowiedziami.
PRZYJACIEL: | Jednak wydaje mi się, że poprawna odpowiedź brzmi: Przysłów
PRZYJACIEL: | Trzymamy za Ciebie kciuki! Powodzenia!
-----
```