Laboratorium 3

1. Cel ćwiczenia
Celem ćwiczenia jest zapoznanie się z obsługą C. Instrukcje **switch**, **pętle, preinkrementacja**, **postinkrementacja.**

Instrukcja wyboru **switch**
Instrukcja switch (‘przełącz’) jest w pewien sposób podobna do if: jej przeznaczeniem jest także wybór jednego z wariantów kodu podczas działania programu. Pomiędzy obiema konstrukcjami istnieją jednak dość znaczne różnice. O ile if podejmuje decyzję na podstawie prawdziwości lub fałszywości jakiegoś warunku, o tyle switch bierze pod uwagę wartość podanego wyrażenia. Z tego też powodu może dokonywać wyboru spośród większej liczby możliwości niż li tylko dwóch (prawdy lub fałszu). Najlepiej widać to na przykładzie:

(Przykład 1 Język C++)

```
#include <iostream>

int main()
{
float fLiczba1;
float fLiczba2;

std::cout << "Podaj wspolczynnik a: ";
std::cin >> fLiczba1;

std::cout << "Podaj wspolczynnik b: ";
std::cin >> fLiczba2;

int nOpcja;
std::cout << "Wybierz dzialanie:" << std::endl;
std::cout << "1. Dodawanie" << std::endl;
std::cout << "2. Odejmowanie" << std::endl;
std::cout << "3. Mnozenie" << std::endl;
std::cout << "4. Dzielenie" << std::endl;
std::cout << "0. Wyjscie" << std::endl;
std::cout << "Twoj wybor: ";
std::cin >> nOpcja;
switch (nOpcja)
{
case 1: std::cout << fLiczba1 << " + " << fLiczba2 << " = "
<< fLiczba1 + fLiczba2; break;
case 2: std::cout << fLiczba1 << " - " << fLiczba2 << " = "
<< fLiczba1 - fLiczba2; break;
case 3: std::cout << fLiczba1 << " * " << fLiczba2 << " = "
```

```
<< fLiczba1 * fLiczba2; break;
case 4:
if (fLiczba2 == 0.0)
std::cout << "Dzielnik nie moze byc zerem!";
else
std::cout << fLiczba1 << " / " << fLiczba2 << " = "
<< fLiczba1 / fLiczba2;
break;
case 0: std::cout << "Dziekujemy :)"; break;
default: std::cout << "Nieznana opcja!";
}
}
```

Pętle (ang. *loops*), zwane też **instrukcjami iteracyjnymi**, stanowią podstawę prawie wszystkich algorytmów. Lwia część zadań wykonywanych przez programy komputerowe opiera się w całości lub częściowo właśnie na pętlach.
Pętla to element języka programowania, pozwalający na wielokrotne, kontrolowane wykonywanie wybranego fragmentu kodu.
Liczba takich powtórzeń (zwanych **cyklami** lub **iteracjami** pętli) jest przy tym ograniczona w zasadzie tylko inwencją i rozsądkiem programisty. Te potężne narzędzia dają więc możliwość zrealizowania niemal każdego algorytmu.
Pętla **do** wykona zawsze co najmniej jeden przebieg.
Pętla **while** może nie wykonać się **ani razu**, jeżeli jej warunek będzie od początku nieprawdziwy.

(Przykład 2 Język C++)

```
#include <iostream>

int main()
{
int nLiczba;
do
{
std::cout << "Wprowadz liczbe wieksza od 10: ";
std::cin >> nLiczba;
} while (nLiczba <= 10);
std::cout << "Dziekuje za wspolprace :)";
}
```

Z instrukcją **break** (‘przerwij’) spotkaliśmy się już przy okazji konstrukcji **switch**. Korzystaliśmy z niej, aby zagwarantować wykonanie kodu odpowiadającego tylko jednemu wariantowi case. **break** powodowała bowiem przerwanie bloku **switch** i przejście do następnej linijki po nim.
Rola tej instrukcji w kontekście pętli nie zmienia się ani na jotę: jej wystąpienie wewnątrz bloku **do**, **while** lub for powoduje dokładnie ten sam efekt. Bez względu na prawdziwość lub nieprawdziwość

warunku pętli jest ona błyskawicznie przerywana, a punkt wykonania programu przesuwa się do kolejnego wiersza za nią.

(Przykład 3 Język C++)

```
#include <iostream>

// Break – przerwanie pętli
int main()
{
int nLiczba;
do
{
std::cout << "Wprowadz liczbe wieksza od 10" << std::endl;
std::cout << "lub zero, by zakonczyc program: ";
std::cin >> nLiczba;
if (nLiczba == 0) break;
} while (nLiczba <= 10);
std::cout << "Nacisnij dowolny klawisz.";
}
```

Umieszczenie operatora ++ (--) **przed** wyrażeniem nazywamy **preinkrementacją** (**predekrementacją**). W takiej sytuacji **najpierw** dokonywane jest zwiększenie (zmniejszenie) jego wartości o 1. Nowa wartość jest potem zwracana jako wynik.

Kiedy napiszemy operator ++ (--) **po** wyrażeniu, mamy do czynienia z **postinkrementacją** (**postdekrementacją**). W tym przypadku najpierw następuje zwrócenie wartości, która dopiero **potem** jest zwiększana (zmniejszana) o jeden.

2. Przykłady

Przykład4
Napisz program, który wymaga wpisania liczby całkowitej. Następnie korzystając z pętli for, należy dokonać odliczania w dół do zera, wyświetlając każdą z odliczanych wartości w oddzielnym wierszu (użyj pętli „for”).

```
#include <stdio.h>

int main(void){
int i,j;
printf("Wprowadz liczbe:\n");
scanf("%d",&i);
for(j=i;j>0;j--)printf("%d\n",j);
return 0;
}
```

Przykład5
Napisać program będący komputerową wersją gry „Zgadnij sekretną liczbę”. Użytkownik ma dziesięć szans na zgadnięcia sekretnej liczby. Jeśli wprowadzona liczba odpowiada liczbie wybranej przez programistę jako sekretna, wówczas pojawia się komunikat „DOBRZE” i następuje koniec gry. W przeciwnym wypadku program informuje, czy wprowadzona liczba jest większa, czy mniejsza od sekretnej i gra toczy się dalej. Gra kończy się, jeśli gracz zgadnie liczbę lub skończy się limit dziesięciu prób. Aby uatrakcyjnić grę, program może zwracać liczbę prób zużytą przed poprawnym trafieniem (użyj pętli „for” i instrukcji if, else).

```
#include <stdio.h>

int main(void){
int sekret; //sekretna liczba
int proba;
int i;

sekret=1325;
proba=0;

for(i=0;i<10&&proba!=sekret;i++)  {
printf("Zgadnij liczbe:\n");
scanf("%d",&proba);
if(proba==sekret){
printf("DOBRZE\n");
printf("%d jest sekretna liczba.\n", sekret);
                        }
else{
printf("...Niestety nie masz racji...");
if(proba>sekret) printf("Proponowana liczba jest za duza\n");
else printf("Proponowana liczba jest za mala\n");
      }
                                              }
return 0;
}
```

Przykład6
Napisać program, który oblicza całkowitą powierzchnię mieszkalną domu na podstawie wymiarów każdego z pokoi. Program powinien zapytać o liczbę pokoi, a następnie prosić o wymiary każdego z nich. Na ekranie należy wypisać wartość całkowitej powierzchni mieszkalnej.

```
#include <stdio.h>

int main(void){
int pokoje, dlugosc, szerokosc, razem;
int i;
printf("Liczba pokoi?\n");
```

```
scanf("%d", &pokoje);
razem=0;
for(i=pokoje;i>0;i--){
printf("Wprowadz dlugosc pokoju %d:\n",i);
scanf("%d",&dlugosc);
printf("Wprowadz szerokosc pokoju %d:\n",i);
scanf("%d",&szerokosc);
razem=razem+dlugosc*szerokosc;
}
printf("Powierzchnia calkowita: %d : ", razem);
return 0;
}
```

3. Program ćwiczenia

Zad1
Napisać program, obliczający pole powierzchni koła lub prostokąta. Należy użyć (instrukcji if, else). Użyć funkcji getchar() np. ch=getchar();

Zad2
Korzystając z instrukcji **switch**, napisać program, który wczytuje znaki z klawiatury i czeka na znaki, nowej linii oraz znaki cofnięcia. W momencie wprowadzenia takiego znaku powinna zostać wyświetlona jego nazwa. Na przykład po naciśnięciu klawisza ENTER powinien pojawić się napis „ENTER” po naciśnięciu BACKSPACE pojawić się ma napis „BACKSPACE”. Program powinien kończyć działanie po wpisaniu znaku „k” (zastosować „switch” i „do while”).

Zad3
Napisać program wczytujący napis, a następnie wyświetlający go na ekranie w odwrotnej kolejności.

Zad4
Napisać program cyklicznie wczytujący napisy. Po wczytaniu każdego napisu należy dołączyć go do innego napisu, pamiętanego w zmiennej **dluginap**. Na końcu każdego napisu należy dodawać znaki nowej linii. Jeśli użytkownik wpisze „**koniec**”, należy przerwać wczytywanie napisów i wyświetlić **dluginap** (która będzie zawierać zapis wszystkich wprowadzonych ciągów znaków). Podobnie należy zatrzymać wczytywanie, gdy dołączanie do **dluginap** kolejnego napisu będzie niemożliwe ze względu na przekroczenie zakresu tablicy **dluginap**.

Zad5
Napisać program wczytujący liczbę i sprawdzający czy jest podzielna przez 3. Program powinien wykorzystywać instrukcję **switch** i **case**.
Program może wypisywać:
printf ("Liczba %d dzieli sie przez 3\n", a);
printf ("Liczba %d nie dzieli sie przez 3\n", a);