PODSTAWY JĘZYKA

C++

CZĘŚĆ II

DANIEL DARUL 2023

Pytanie 1
Opisz konstrukcję pętlę while, jaka to pętla, narysuj schemat, przepisz program jako przykład.
Pytanie 2
Opisz instrukcję kbhit()
Pytanie 3
Opisz konstrukcję pętlę do while, jaka to pętla, narysuj schemat, przepisz program jako przykład.
Pytanie 4
Opisz konstrukcję instrukcji warunkowa if z własnym przykładem innym niż w instrukcji, narysuj schemat.
Pytanie 5
Opisz konstrukcję instrukcji warunkowa if...else z własnym przykładem innym niż w instrukcji, narysuj schemat.
Pytanie 6
Na podstawie przykładu zapisz co oznacza zapis SUMA+=(++n);
Pytanie 7
Napisz przykład zagnieżdżonej instrukcji if z własnym przykładem innym niż w instrukcji.
Pytanie 8
Opisz instrukcję Continue oraz Break

Pętle→definicja i rodzaje

Pętla jest to struktura informatyczna, która wykonuje powtarzanie pewnych operacji w programie np. wyświetlaj napis „Programowanie jest piękne" 10 razy.

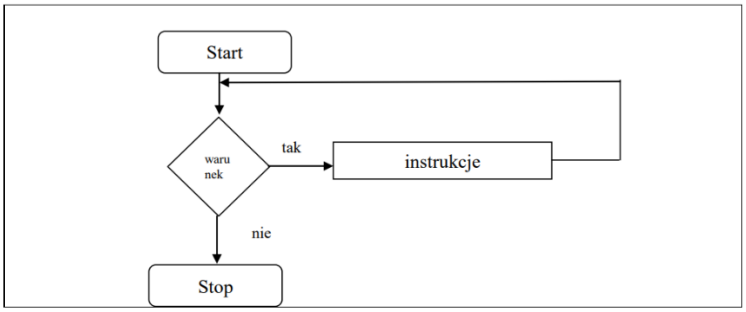
Rodzaje pętli:
a)pętla **while**
b)pętla **do while**
c)petla **for**

PĘTLA while.

Pętlę typu **while** jest pętlą z **kontrolowanym wejściem** tzn. najpierw jest obliczany warunek a po jego spełnieniu wchodzimy do pętli i wykonujemy instrukcje z niej.
Konstrukcja pętli while wygląda następująco:

```
while (Wyrażenie_logiczne)
{
        Instrukcja1;
        ……..
        InstrukcjaN;
}
```

Jeśli Wyrażenie_logiczne ma wartość logiczną zera, to nie zostaną wykonane Instrukcje czyli nie nastąpi wejście do pętli..



Przykład 1

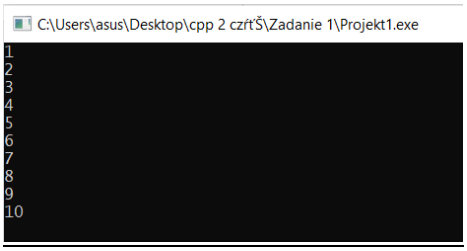
Temat: Wykorzystanie pętlę while do programu piszącego wykrzykniki aż do wciśnięcia dowolnego klawisza.

Funkcja kbhit() oznacza wciśnięcie dowolnego klawisza.

Wykonaj:
1)Przepisz temat.
2)Zapisz teorię+schemat blokowy dotyczącą pętli while.
3)Zapisz, co oznacza funkcja kbhit() , !kbhit(), printf("\n"),
4)Uruchom przykład,
5)Zapisz przykład do zeszytu (pamiętaj o wcięciach).

```
#include <cstdlib>
#include <iostream>
#include <stdio.h>
#include <conio.h>

using namespace std;

int main(int argc, char *argv[])
{
   while(!kbhit())
   {
      printf("!");
   }
   printf("\n");
   system("PAUSE");
   return EXIT_SUCCESS;
}
```

Zadanie 1 Napisz program w C++, który wypisuje liczby całkowite od 1 do 10. Użyj pętli while, aby to zrobić.



Zadanie 2 Napisz program w C++, który używa pętli while do wypisania parzystych liczb od 2 do 10**.**

PĘTLA do...while.

Pętlę typu do while jest pętlą **z kontrolowanym wyjściem** tzn. warunek obliczany po wykonaniu pętli.
Przerwanie pętli powodowane jest przez **NIESPEŁNIENIE WARUNKU.**

Konstrukcja pętli do while wygląda następująco:

```
do
{
        Instrukcja1
        Instrukcja2
……..
        InstrukcjaN
}

while (Wyrażenie_logiczne);
```

Inkrementacja
Operator inkrementacji (++), czyli zwiększenie wartości zmiennej o stałą wartość najczęściej o jeden (1). Np.
i=i+5;
i=i+1; lub w formie skróconej i++;

Dekrementacja
Operator dekrementacji (--), czyli zmniejszenie wartości zmiennej o stałą wartość najczęściej o jeden (1). Np.
x=x+5;
x=x-1; lub w formie skróconej x--;

Formy zapisu:
Przedrostkowa np. (++i --x) najpierw zmienna jest zwiększana o jeden, a następnie ta zwiększona wartość jest brana do obliczeń.

Przyrostkowa (końcówkowa) (i++ x--) najpierw brana jest stara wartość zmiennej do obliczeń a dopiero później jest ona zwiększana o jeden.

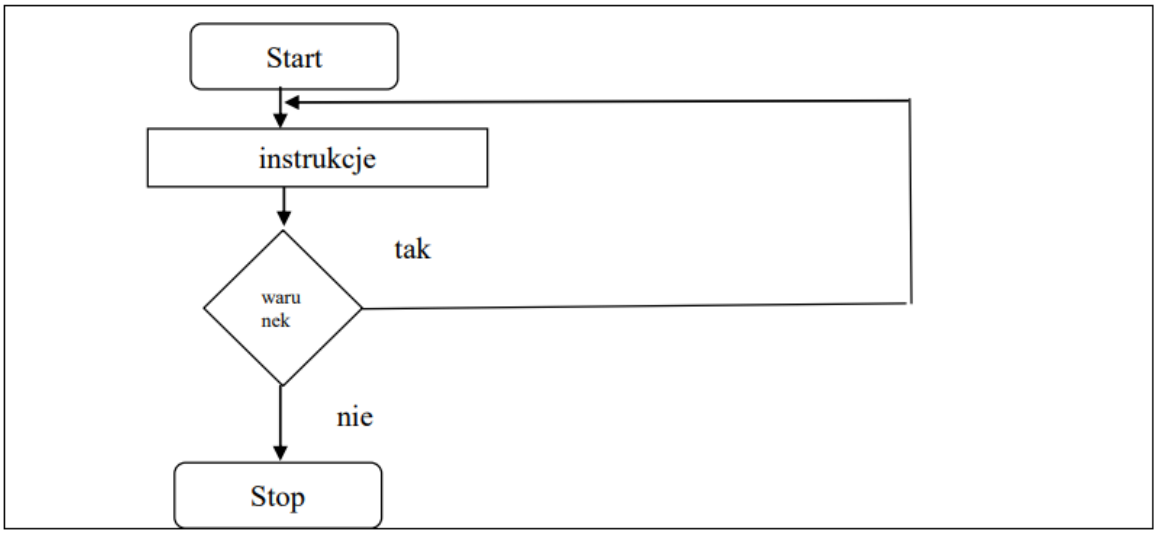
Przykład 2

Temat: Stosujemy pętlę while w programie obliczającym sumę kolejnych liczb naturalnych dopóki SUMA nie przekroczy 1000.

Wykonaj:
1)Przepisz temat,
2)Zapisz, co oznacza instrukcja SUMA+=(++n);,
3)Uruchom przykład,
4)Zapisz w zeszycie daną wyjścia dla programu,
5)Zapisz przykład do zeszytu (pamiętaj o wcięciach).
6)Wykonaj schematy blokowy.

```
#include <cstdlib>
#include <iostream>

using namespace std;

int main(int argc, char *argv[])
{
 float SUMA=0, n=0;
 while (SUMA<1000)
  {
  SUMA+=(++n);          // SUMA = SUMA + n;
                        // ++n zwiększaj zmienna a o jeden
                        //w formie przedrostkowej
                        // inny zapis ++n  to     n=n+1;
  }
  printf("SUMA: %.1f ostatnia liczba: %.2f", SUMA, n);
  cout<<"\n"<<endl;
 system("PAUSE");
 return EXIT_SUCCESS;
}
```

Uwaga:
Podczas procesu **sumowa** w dolnym jezyku programowania zmienna sumujaca musi być zerowana przed przystąpieniem do czynności sumowania np. **suma=0;** .
Podczas procesu **mnożenia** w dolnym jezyku programowania zmienna w którym zapisujemy aktualny iloczyn musi mieć wartość 1 (jeden) przed przystąpieniem do czynności obliczania iloczynu np. **iloczyn=1; .**



Przykład 3

Temat: Program obliczający pole trójkąta ze sprawdzeniem poprawności wczytywania danych z użyciem pętli **do…while**.

```
#include <cstdlib>
#include <iostream>

using namespace std;

int main(int argc, char *argv[])
{
 float bok_a, wysokosc, pole;
 do
    {
     cout<<"podaj dlugosc podstawy a=";
     cin>> bok_a;
     }
 while (bok_a<=0);
 do
    {
     cout<<"podaj dlugosc wysokosci=";
     cin>> wysokosc;
     }
 while (wysokosc<=0);
 pole = bok_a*wysokosc/2;
 cout<<"pole="<<pole<<" m^2";
  cout<<"\n"<<endl;
 system("PAUSE");
 return EXIT_SUCCESS;
}
```

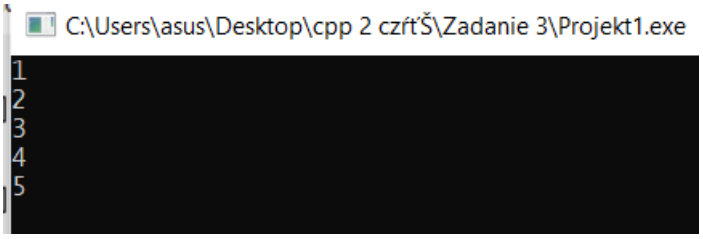
Koniec przykładu

Przykład 4

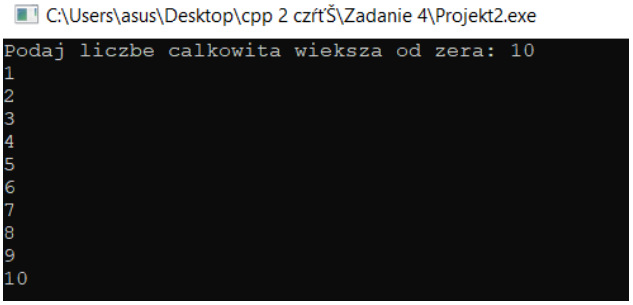
Temat: Demonstracja pętli **do…while** do zapewnienia powtarzalności programu.

```
#include <cstdlib>
#include <iostream>
#include <stdio.h>
#include <conio.h>

using namespace std;

int main(int argc, char *argv[])
{
   char znak;
  do
     {
       printf("Umiem powtarzać  program\n");
        printf("\nczy powtórzyć t(tak)/n(nie) wciśnij klawisz\n");
        cin>>znak;
        }
     while (znak=='t');

   return EXIT_SUCCESS;
}
```

Zadanie 3 Napisz program w C++, który używa pętli do-while do wypisania liczb od 1 do 5.



Zadanie 4 Napisz program w C++, który prosi użytkownika o wprowadzenie liczby całkowitej większej od zera, a następnie wypisuje liczby od 1 do wprowadzonej wartości. Użyj pętli do-while do realizacji tego zadania.



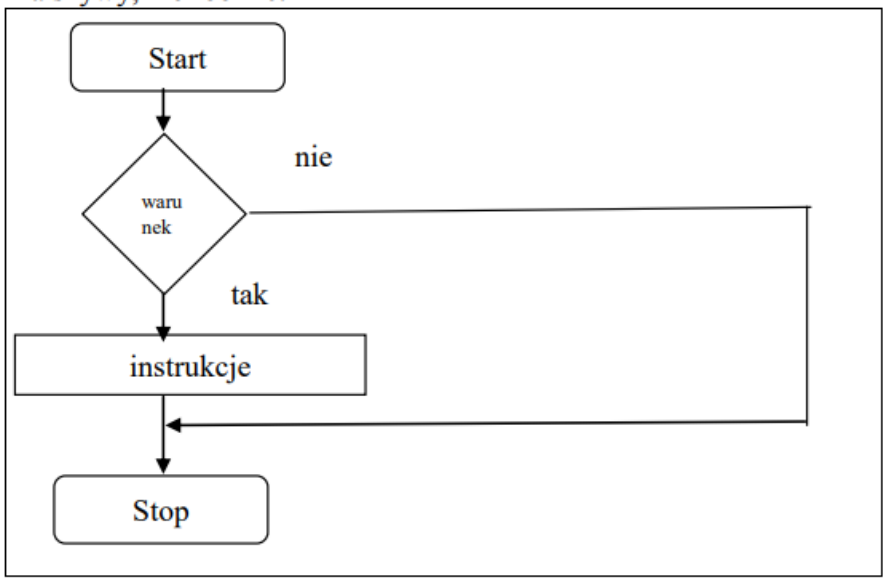
INSTRUKCJA WARUNKOWA if

Instrukcja warunkowa ma postać:

```
if (Wyrażenie)
{
        Instrukcja1;
        ……..
        InstrukcjaN;
}
```

np.
if (a<b) {cout<<"Liczba a jest mniejsza od b"<<endl;}

Jeśli warunek jest prawdziwy, wykonaj instrukcję lub grupę instrukcji. Gdy warunek jest fałszywy, nie rób nic.



INSTRUKCJA WARUNKOWA if else

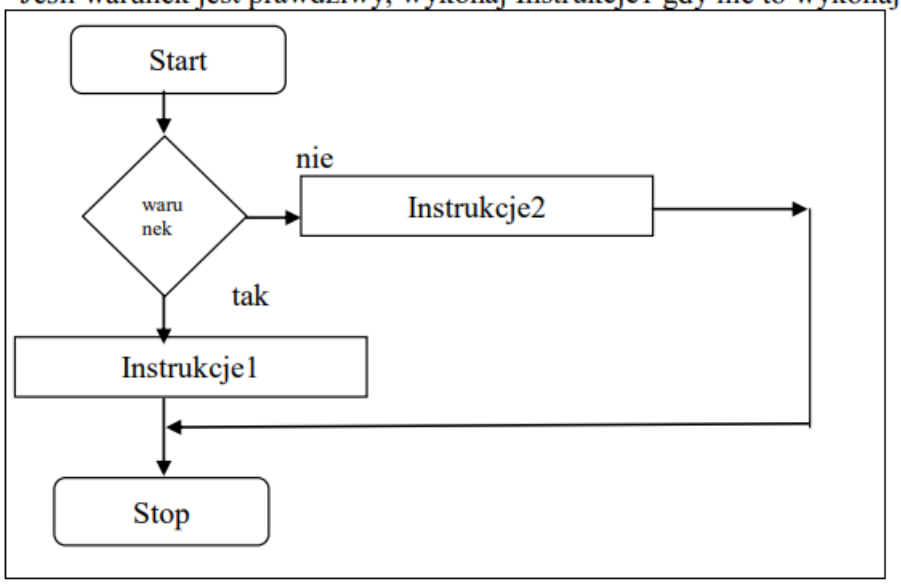
Instrukcja warunkowa ma postać:

```
if (Wyrażenie)
{
        Instrukcja1;
        ……..
        InstrukcjaN;
}
    else
{
        Instrukcja1;
        ……..
        InstrukcjaN;
}
```

np.

```
if (a<b)
        {cout<<"Liczba a jest mniejsza od b"<<endl;}
                                else
        {cout<<"Liczba a jest większa lub równa  od b"<<endl;}
```

Jeśli warunek jest prawdziwy, wykonaj Instrukcje1 gdy nie to wykonaj Instrukcję2. Gdy



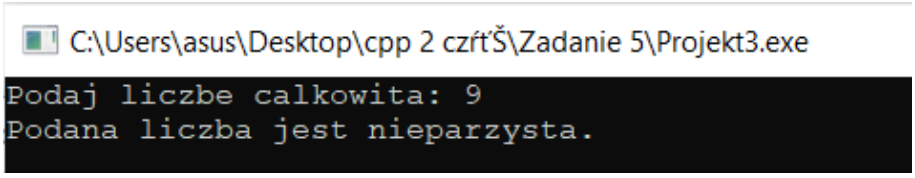
Przykład 5

Temat: Prezentacja instrukcji **if** dla podania komunikatu o nieprawidłowego wczytania danych.

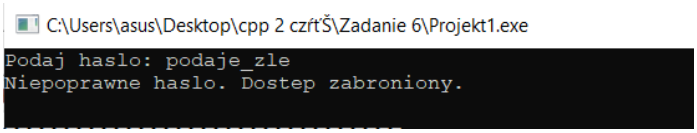
Wykonaj:
1)Przepisz temat.
2)Zapisz teorię +schemat blokowy dotyczącą instrukcji if.
3)Uruchom przykład
4)Wpisz przykład do zeszytu.

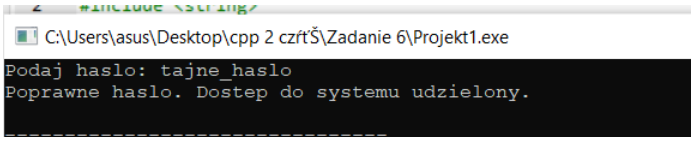
```
#include <cstdlib>
#include <iostream>

using namespace std;

int main(int argc, char *argv[])
{
   system("chcp 1250");


   system("cls");
   int waga;
   do
     {
      printf("Podaj Twoja w [kg] wagę=");
      scanf("%d",&waga);
      if (waga<=0)
        {  printf("waga musi być większa od zera\n");     }
     }
    while(!(waga>0));
    printf("waga wczytana poprawnie\n");
   system("PAUSE");
   return EXIT_SUCCESS;
}
```

Zadanie 5 Napisz program w C++, który pyta użytkownika o podanie liczby całkowitej i sprawdza, czy ta liczba jest parzysta czy nieparzysta. Wykorzystaj instrukcję warunkową if do wyświetlenia odpowiedniego komunikatu w zależności od wyniku.



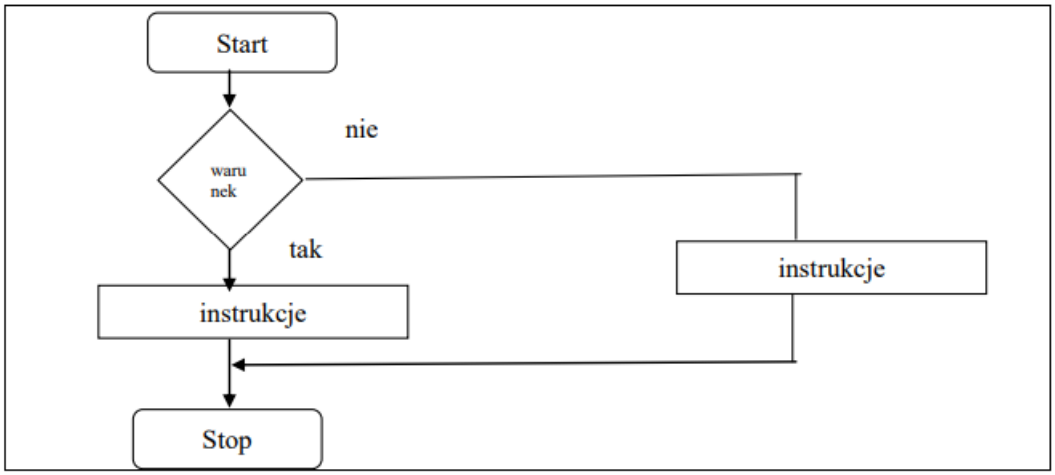
Zadanie 6 Napisz program w C++, który ma ustalone hasło (np. "tajne_haslo") i prosi użytkownika o wprowadzenie hasła. Program sprawdza, czy podane hasło jest poprawne, a następnie wyświetla odpowiedni komunikat.





INSTRUKCJA WARUNKOWA if...else

```
if (Wyrażenie)
{
  Instrukcja1
  ……..
  InstrukcjaN;
}
else
{
   Instrukcja1
   ……..
  InstrukcjaN;
}
```



Jeśli Wyrażenie jest prawdziwy to zostanie wykonana Instrukcja1, w przeciwnym razie wykonana zostanie Instrukcja2. Instrukcje mogą być instrukcjami grupującymi. Słowa kluczowe if i else mogą być stosowane wielokrotnie. Pozwala to tworzyć np. tzw. zagnieżdżenia.

Przykład instrukcji zagnieżdżonych:

```
if (a>0) if (a<100) printf("Dwucyfrowa"); else printf("100+");
```

inaczej:

```
if(a>0) {if(a<100) printf("Dwucyfrowa"); else printf("100+");}
```

Zapis 100+ oznacza "sto i więcej".

INSTRUKCJE break i continue.

Instrukcja **break** powoduje natychmiastowe bezwarunkowe opuszczenie pętli dowolnego typu i przejście do najbliższej instrukcji po zakończeniu pętli. Jeśli w pętli for opuścimy wyrażenie logiczne, to zostanie automatycznie przyjęte 1. Pętla będzie, zatem wykonywana bezwarunkowo w nieskończoność

Instrukcja **continue** powoduje przedwczesne, bezwarunkowe zakończenie wykonania wewnętrznej instrukcji pętli i podjęcie próby realizacji następnego cyklu pętli. Próby, ponieważ najpierw zostanie sprawdzony warunek kontynuacji pętli.

INSTRUKCJE switch i case.

Instrukcja switch dokonuje WYBORU w zależności od stanu wyrażenia przełączającego (selector) jednego z możliwych przypadków - wariantów (case). Każdy wariant jest oznaczony przy pomocy stałej - tzw. ETYKIETY WYBORU. Wyrażenie przełączające może przyjmować wartości typu int. Ogólna postać instrukcji jest następująca:

Należy podkreślić, że po dokonaniu wyboru i skoku do etykiety wykonane zostaną również WSZYSTKIE INSTRUKCJE PONIŻEJ DANEJ ETYKIETY. Jeśli chcemy tego uniknąć, musimy dodać rozkaz break.

#define.

Użycie #define polega na zastąpieniu w tekście programu jednych łańcuchów znaków przez inne.
np.

```
# define Program main()
# define Begin {
# define Writeln printf
# define Readln getch()
# define End }
```

Takie zdefiniowanie pozwala zastąpić instrukcje CPP instrukcjami Pascala.

Przykład 6

Zapisz w zeszycie składnię instrukcji

- INSTRUKCJE switch i case (znaczenie break).
- #define.

Treść programu

Program po podaniu numeru dnia tygodnia odpowie, jaki to dzień tygodnia. W przykładzie zastosowano #define.

Wykonaj:

- Wpisz przykład do komputera oraz przetestuj go.
- Wykonaj schemat blokowy przykładu.

```
#define pisz printf   //dla przypomnienia
#include <cstdlib>
#include <iostream>
#include <stdio.h>
using namespace std;

int main(int argc, char *argv[])
{
   int Numer_Dnia;
   pisz("\nPodaj numer dnia tygodnia\n");
   scanf("%d",&Numer_Dnia);
   switch(Numer_Dnia)
      {
      case 1: {pisz("PONIEDZIALEK");break;}
      case 2: {pisz("WTOREK");break;}
      case 3: {pisz("SRODA");break;}
      case 4: {pisz("CZWARTEK");break;}
      case 5: {pisz("PIATEK");break;}
      case 6: {pisz("SOBOTA");break;}
      case 7: {pisz("NIEDZIELA");break;}
      default: pisz("\nniema takiego dnia tygodnia!!!");
      }
   cout<<endl;
   system("PAUSE");
   return EXIT_SUCCESS;
}
```

Zwróć uwagę, że w przykładzie wariant default zostanie wykonany ZAWSZE nawet, jeśli podasz liczbę większą niż 7.

Skok bezwarunkowy goto (idź do...),

Gdy chcesz, aby program przeszedł do określonego miejsca w programie możesz użyć tak zwanej **etykiety**.

Etykieta skład się z:

- polecenia **goto** nazwa; np. goto skocz;
- oraz nazwy zakończonej dwukropkiem np. **dawaj:** gdzie trzeba skoczyć.

UWAGA: Po każdej etykiecie musi wystąpić co najmniej jedna instrukcja. Jeśli etykieta oznacza koniec programu, to musi po niej wystąpić instrukcja pusta. Należy tu zaznaczyć, że etykieta nie wymaga deklaracji.

np.

```
goto dawaj;

// część programu

:dawaj
```

Różnica między instrukcją getch() a kbhit()

a) kbhit()
Aż do naciśnięcia dowolnego klawisza.
b)!kbhit()
Zapis !kbhit() oznacza "nie naciśnięto klawisza", czyli w buforze klawiatury nie oczekuje znak.
c) getch()
Zwróć uwagę, że funkcja getch() może oczekiwać na klawisz w nieskończoność. Aby uniknąć kłopotliwych sytuacji, czasem znacznie wygodniej jest zastosować kbhit(), szczególnie, jeśli czekamy na dowolny klawisz.

Zadanie 7 Napisz program, który poprosi użytkownika o podanie jego masy (w kilogramach) i wzrostu (w metrach). Na podstawie tych danych, oblicz wskaźnik BMI (Body Mass Index) wg wzoru: BMI = masa / (wzrost * wzrost) i pokaż komunikat z kategorią BMI (np. "Niedowaga", "Prawidłowa waga", "Nadwaga", "Otyłość").

C:\Users\asus\Desktop\cpp 2 czŕťŠ\Zadanie 7\Projekt1.exe

```
Podaj mase (kg): 65
Podaj wzrost (m): 1.72
Prawidlowa waga

--------------------------------
```

C:\Users\asus\Desktop\cpp 2 czŕťŠ\Zadanie 7\Projekt1.exe

```
Podaj mase (kg): 99
Podaj wzrost (m): 1.72
Otylosc
```

C:\Users\asus\Desktop\cpp 2 czŕťŠ\Zadanie 7\Projekt1.exe

```
Podaj mase (kg): 25
Podaj wzrost (m): 1.72
Niedowaga

--------------------------------
```

Zadanie 8 Napisz program w C++, który działa jak prosty kalkulator do wykonywania operacji dodawania, odejmowania, mnożenia i dzielenia na dwóch liczbach. Użytkownik podaje operację (+, -, *, /) oraz dwie liczby, a program wyświetla wynik. Wykorzystaj instrukcję warunkową if do wybrania właściwej operacji.

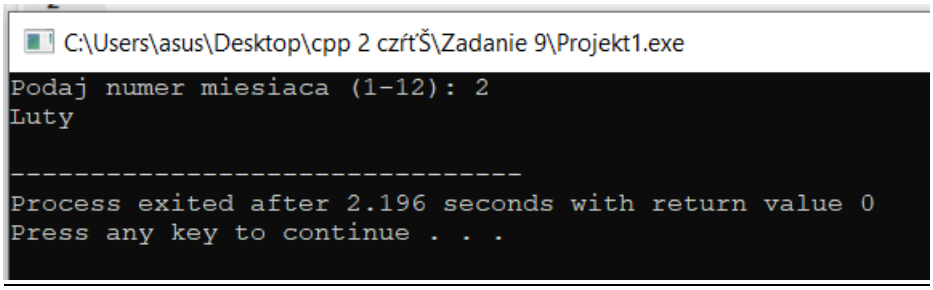
C:\Users\asus\Desktop\cpp 2 czŕťŠ\Zadanie 8\Projekt4.exe

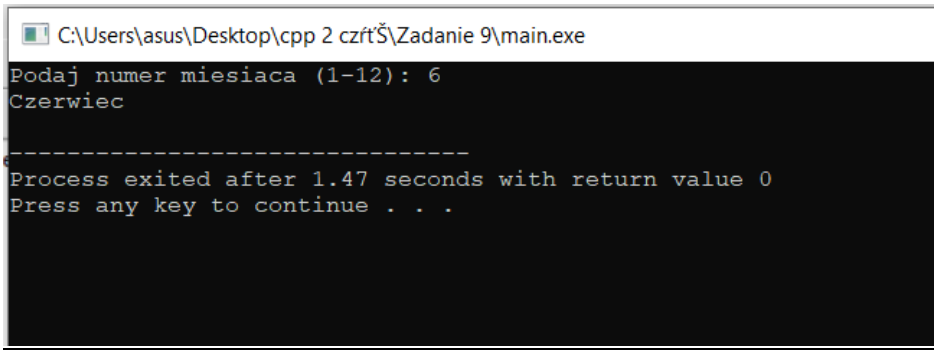
```
Podaj operacje (+, -, *, /): +
Podaj pierwsza liczbe: 3
Podaj druga liczbe: 3
Wynik: 6
```

C:\Users\asus\Desktop\cpp 2 czŕťŠ\Zadanie 8\Projekt4.exe

```
Podaj operacje (+, -, *, /): -
Podaj pierwsza liczbe: 3
Podaj druga liczbe: 3
Wynik: 0
```

Zadanie 9 Napisz program w języku C++, który pyta użytkownika o numer miesiąca (1-12) i następnie wypisuje nazwę tego miesiąca. Musisz użyć instrukcji switch do rozpoznania numeru miesiąca i wypisania odpowiadającej mu nazwy. Upewnij się, że program obsługuje poprawne numery miesięcy (1-12) oraz wyświetla komunikat o błędzie, gdy użytkownik poda nieprawidłowy numer.





Zadanie 10 Napisz program w C++, który symuluje grę w zgadywanie liczby. Program wybierze losowo liczbę z zakresu od 1 do 100, a użytkownik będzie próbował zgadnąć tę liczbę.

Po każdym wprowadzeniu liczby przez użytkownika, program podpowie, czy wybrana liczba jest za duża, za mała, czy trafiona.

Program powinien generować losową liczbę w zakresie od 1 do 100 (użyj funkcji rand()). Poproś użytkownika o zgadnięcie wybranej liczby.

Program powinien sprawdzić, czy podana liczba jest większa, mniejsza lub równa wybranej liczbie. W zależności od wyniku, program wyświetli odpowiedni komunikat.

Gra powinna trwać, dopóki użytkownik nie zgadnie liczby. Po zgadnięciu liczby, program powinien wyświetlić liczbę prób, które użytkownik podjął, zanim zgadł liczbę.