



# Podstawy Programowania

dr inż. Tomasz Marciniak

# Wykład 5

- Unie
- Operacje na plikach
- dyrektywy kompilatora

# Unia

- Jest strukturą, której wszystkie składowe umieszczone są w tym samym obszarze pamięci,
- Deklaracja unii podobna jest do deklaracji struktury

# Deklaracja unii

```
union test  
{  
    int x1;  
    unsigned char t[4];  
};
```

# Rozmiar unii

- Dla unii zostaje przydzielona taka ilość pamięci, aby mogła pomieścić największy typ zmiennej wchodzącej w skład unii,

union zmien

{

int x1;

char zn;

4 bajty

};

# Definicja zmiennej

- union test  
  {  
    long x1;  
    unsigned char t[4];  
  } zm1,zm2;
- union test zm1,zm2;

# Inicjowanie unii

```
union test
{
    int x1;
    unsigned char t[4];
}ut;
```

```
int main(){
    ut.x1=123; ←
```

```
union test
{
    int x1;
    unsigned char t[4];
}ut={123}; ←
```

```
int main(){
}
```

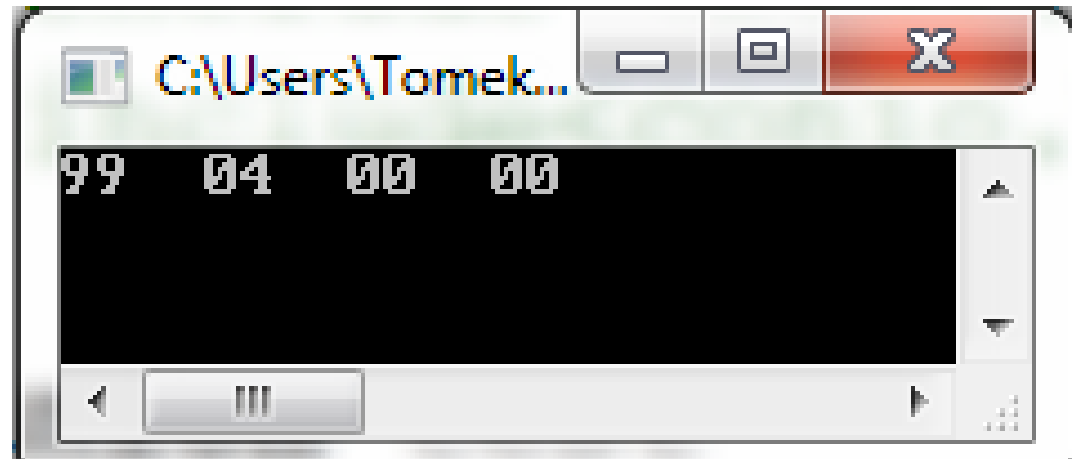
# Przykład

```
1  #include <stdio.h>
2  #include<conio.h>
3
4  union test{
5      int x1;
6      unsigned char t[4];
7  };
8
9  int main(){
10     union test ut;
11     ut.x1=1123;
12     printf("%02d %02d %02d %02d", ut.t[0], ut.t[1], ut.t[2], ut.t[3]);
13     getch();
14 }
```



# Przykład

```
1  #include <stdio.h>
2  #include<conio.h>
3
4  union test{
5      int x1;
6      unsigned char t[4];
7  };
8
9  int main(){
10     union test ut;
11     ut.x1=1123;
12     printf("%02d %02d %02d %02d", ut.t[0], ut.t[1], ut.t[2], ut.t[3]);
13     getch();
14 }
```

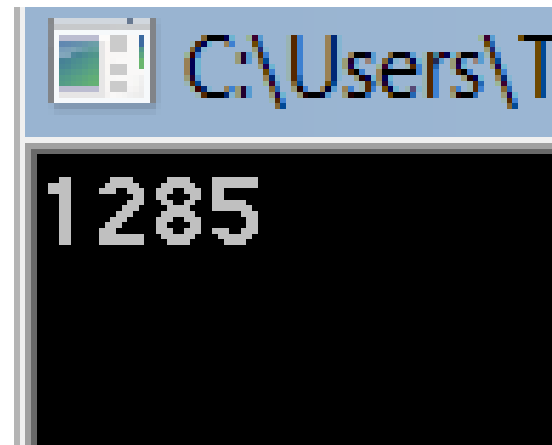


# Przykład

```
1  #include <stdio.h>
2  #include<conio.h>
3
4  union test{
5      int x1;
6      unsigned char t[4];
7  };
8
9  int main(){
10     union test ut;
11     ut.t[0]=5;
12     ut.t[1]=5;
13     ut.t[2]=0;
14     ut.t[3]=0;
15     printf("%d", ut.x1);
16     getch();
17 }
```

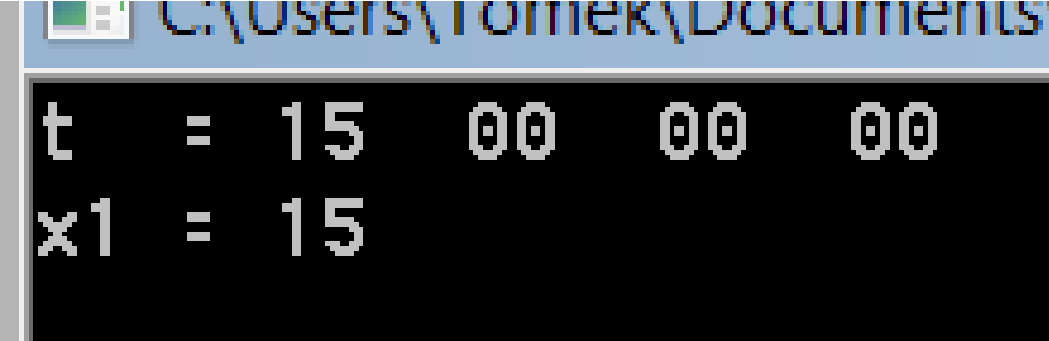
# Przykład

```
1  #include <stdio.h>
2  #include <conio.h>
3
4  union test{
5      int x1;
6      unsigned char t[4];
7  };
8
9  int main(){
10     union test ut;
11     ut.t[0]=5;
12     ut.t[1]=5;
13     ut.t[2]=0;
14     ut.t[3]=0;
15     printf("%d", ut.x1);
16     getch();
17 }
```



# Przykład – inicjowanie

```
1  #include <stdio.h>
2  #include<conio.h>
3
4  union test{
5      int x1;      ←
6      unsigned char t[4];
7  }ut={15};      ←
8
9  int main(){
10     printf("t  = %02d %02d %02d %02d \n", ut.t[0], ut.t[1], ut.t[2], ut.t[3]);
11     printf("x1 = %d", ut.x1);
12     getch();
13 }
```



```
C:\Users\Tomek\Documents>
t  = 15 00 00 00
x1 = 15
```

# Przykład – inicjowanie

```
1  #include <stdio.h>
2  #include <conio.h>
3
4  union test{
5      int x1;
6      unsigned char t[4];
7  }ut={15,0,0,0};
8
9  int main(){
10     printf("t = %02d %02d %02d %02d \n", ut.t[0], ut.t[1], ut.t[2], ut.t[3]);
11     printf("x1 = %d", ut.x1);
12     getch();
13 }
```

anie\p... [Error] too many initializers for 'test'

# Przykład – inicjowanie

```
1  #include <stdio.h>
2  #include<conio.h>
3
4  union test{
5      unsigned char t[4];
6      int x1;
7  }ut={15,0,0,0};
8
9  int main(){
10     printf("t  = %02d  %02d  %02d  %02d \n", ut.t[0], ut.t[1], ut.t[2], ut.t[3]);
11     printf("x1 = %d", ut.x1);
12     getch();
13 }
```



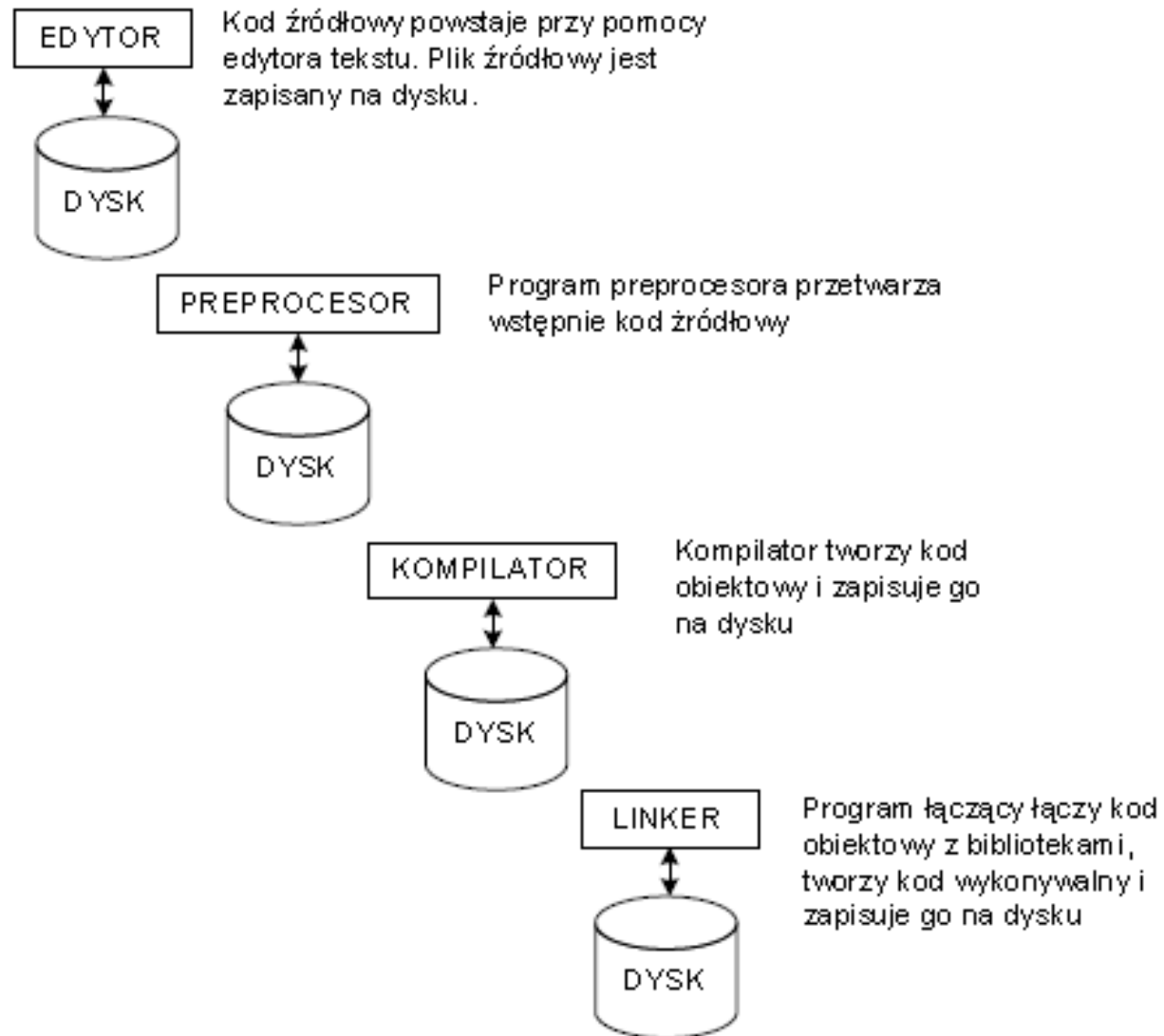
t = 15 00 00 00  
x1 = 15

# Przykład – inicjowanie

```
1  #include <stdio.h>
2  #include<conio.h>
3
4  union test{
5      unsigned char t[4];
6      int x1;
7      unsigned short s[2];
8  }ut={15,0,0,0};
9
10 int main(){
11     printf("t  = %02d  %02d  %02d  %02d \n", ut.t[0], ut.t[1], ut.t[2], ut.t[3]);
12     printf("x1 = %d \n", ut.x1);
13     printf("s  = %04d  %04d \n", ut.s[0], ut.s[1]);
14     getch();
15 }
```

```
t   = 15   00   00   00
x1  = 15
s   = 0015  0000
```

# Fazy przetwarzania programu



Źródło: „Język C – podstawy programowania”, P. Mikołajczak, UMCS 2011



# Dyrektywy preprocesora

- Odpowiadają za wstępne przetwarzanie programu,
- Dołączanie plików,
- Zamiana skrótów symbolicznych na ich definicje;
- Kod warunkowy.

# #include

- #include <conio.h>
- #include "plik2.h "
- #include "c:\temp\plik3.h"

# #include

Przeszukuje katalogi systemowe

- #include <conio.h>
- #include "plik2.h"
- #include "c:\temp\plik3.h"

Przeszukuje bieżący  
katalog roboczy

Przeszukuje katalog c:\temp

# #define

- Tworzy stałe symboliczne
- Tworzy makroinstrukcje

# #define

- #define PI 3.14
- #define roz\_tab 100
- #define MAX(x,y) ((x)>(y)?(x):(y))

# #ifdef

- Można realizować kompilację warunkową,
- Używamy:

#ifdef / #ifndef

#else

#endif

# #ifdef - konstrukcja

`#ifdef` / `#ifndef` *nazwa*

*ciąg instrukcji*

`#else`

*ciąg instrukcji*

`#endif`

# #ifdef - konstrukcja

`#ifdef` / `#ifndef` *nazwa*

*ciąg instrukcji*

`#else`

*ciąg instrukcji*

`#endif`

Kiedy w praktyce możemy zastosować kompilację warunkową ?



# Przykład I

```
#include <stdio.h>
#include <conio.h>
#define klient I
union test
{
    int xI;
    unsigned char t[4];
}ut={123};

int main(){
#ifdef klient I
    printf("%08d",ut.xI);
#else
    printf("%02d %02d %02d %02d", ut.t[0], ut.t[1], ut.t[2], ut.t[3]);
#endif
    getch();
}
```

# Przykład I

```
#include <stdio.h>
```

```
#include <conio.h>
```

```
#define klient1
```

```
union test
```

```
{
```

```
    int x1;
```

```
    unsigned char t[4];
```

```
    }ut={123};
```

```
int main(){
```

```
#ifdef klient1
```

```
    printf("%08d",ut.x1);
```

```
#else
```

```
    printf("%02d %02d %02d %02d", ut.t[0], ut.t[1], ut.t[2], ut.t[3]);
```

```
#endif
```

```
    getch();
```

```
}
```

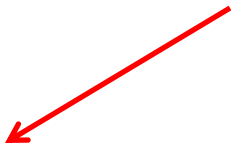
Definicja stałej klient1



# Przykład I

```
#include <stdio.h>
#include <conio.h>
#define klient1
union test
{
    int x1;
    unsigned char t[4];
}ut={123};

int main(){
#ifdef klient1
    printf("%08d",ut.x1);
#else
    printf("%02d %02d %02d %02d", ut.t[0], ut.t[1], ut.t[2], ut.t[3]);
#endif
    getch();
}
```



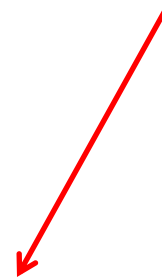
Jeśli zdefiniowaliśmy klient1 to drukuj int

# Przykład I

```
#include <stdio.h>
#include <conio.h>
#define klient1
union test
{
    int x1;
    unsigned char t[4];
}ut={123};

int main(){
#ifdef klient1
    printf("%08d",ut.x1);
#else
    printf("%02d %02d %02d %02d", ut.t[0], ut.t[1], ut.t[2], ut.t[3]);
#endif
    getch();
}
```

Jeśli nie zdefiniowaliśmy klient1 to drukuj tablicę



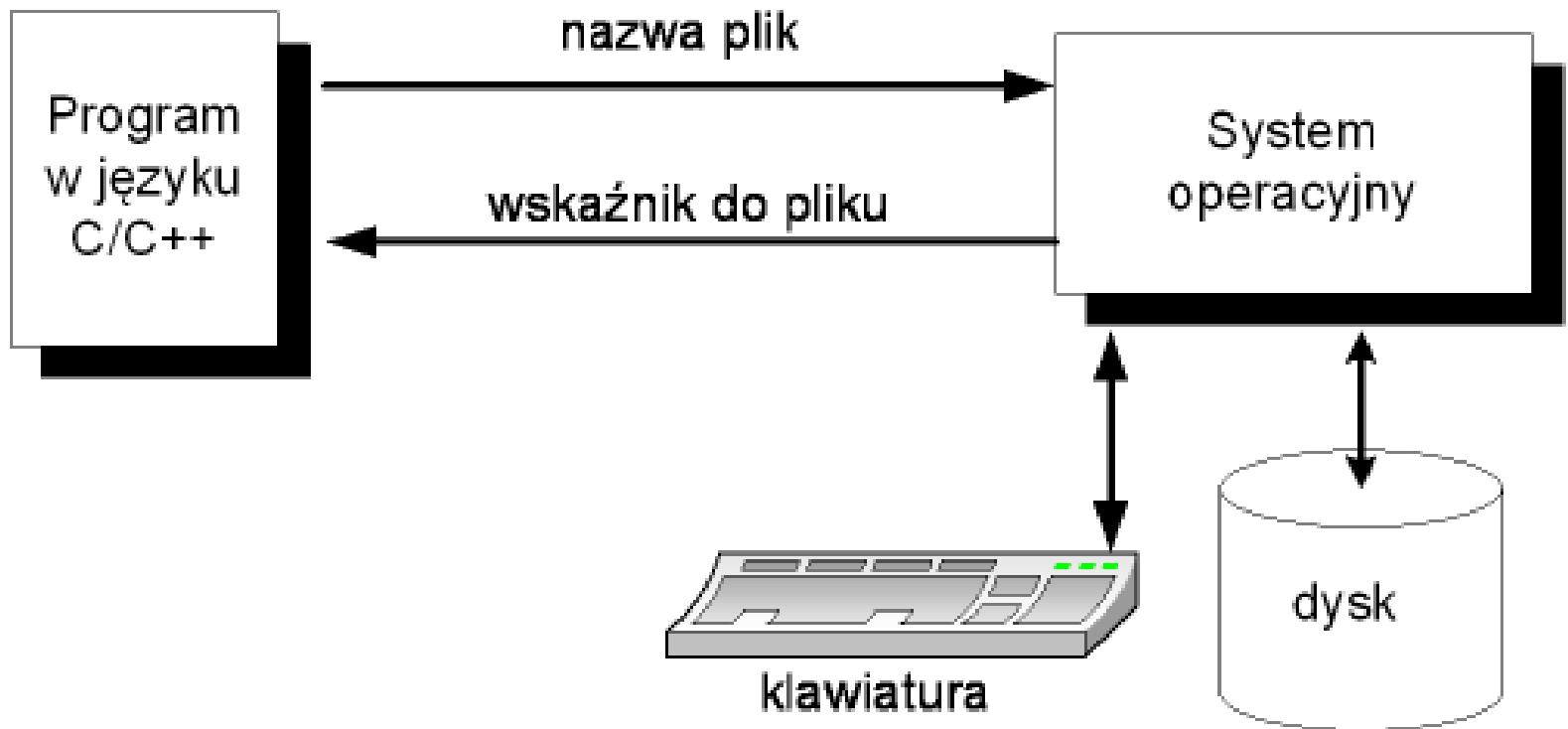
# Inne przykłady zastosowań

- Pisanie programu pod różne wersje systemu,
- Pisanie programu pod różne systemy,
- Umieszczanie w wersji testowej wydruków testowych.

# Pliki

- Umożliwiają przechowywanie danych na dysku;
- Dostęp przez wskaźnik zdefiniowany w `stdio.h`;
- Plik może zostać otwarty w trybie tekstowym lub binarnym

# System plików



# Tryb tekstowy-przykład

```
#include <stdio.h>
#include <conio.h>
int main()
{
    FILE *f;
    char zn;
    f = fopen ("test35.txt","w");
    printf("\nWprowadz tekst :\n");
    while ( (zn=getche()) != '\r')
        putc(zn,f);
    fclose(f);
    return 0;
}
```



# Tryb tekstowy-przykład

```
#include <stdio.h>
```

```
#include <conio.h>
```

```
int main()
```

```
{
```

```
    FILE *f;
```

```
    char zn;
```

```
    f = fopen ("test35.txt","w");
```

```
    printf("\nWprowadz tekst :\n");
```

```
    while ( (zn=getche()) != '\r')
```

```
        putc(zn,f);
```

```
    fclose(f);
```

```
    return 0;
```

```
}
```

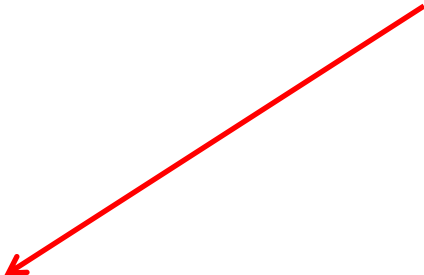
Definicja wskaźnika na plik



# Tryb tekstowy-przykład

```
#include <stdio.h>
#include <conio.h>
int main()
{
    FILE *f;
    char zn;
    f = fopen ("test35.txt","w");
    printf("\nWprowadz tekst :\n");
    while ( (zn=getche()) != '\r')
        putc(zn,f);
    fclose(f);
    return 0;
}
```

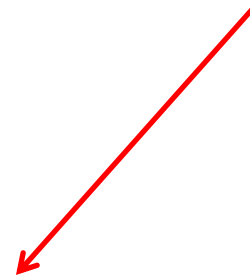
Definicja zmiennej  
znakowej



# Tryb tekstowy-przykład


```
#include <stdio.h>
#include <conio.h>
int main()
{
    FILE *f;
    char zn;
    f = fopen ("test35.txt","w");
    printf("Wprowadz tekst :\n");
    while ( (zn=getche()) != '\r')
        putc(zn,f);
    fclose(f);
    return 0;
}
```

Otwarcie pliku o nazwie „test35.txt” do zapisu



# Tryb tekstowy-przykład

```
#include <stdio.h>
#include <conio.h>
int main()
{
    FILE *f;
    char zn;
    f = fopen ("test35.txt","w");
    printf("\Wprowadz tekst :\n");
    while ( (zn=getche()) != '\r')
        putc(zn,f);
    fclose(f);
    return 0;
}
```



Zapis wprowadzonego znaku  
do pliku do chwili  
przyciśnięcia ENTER

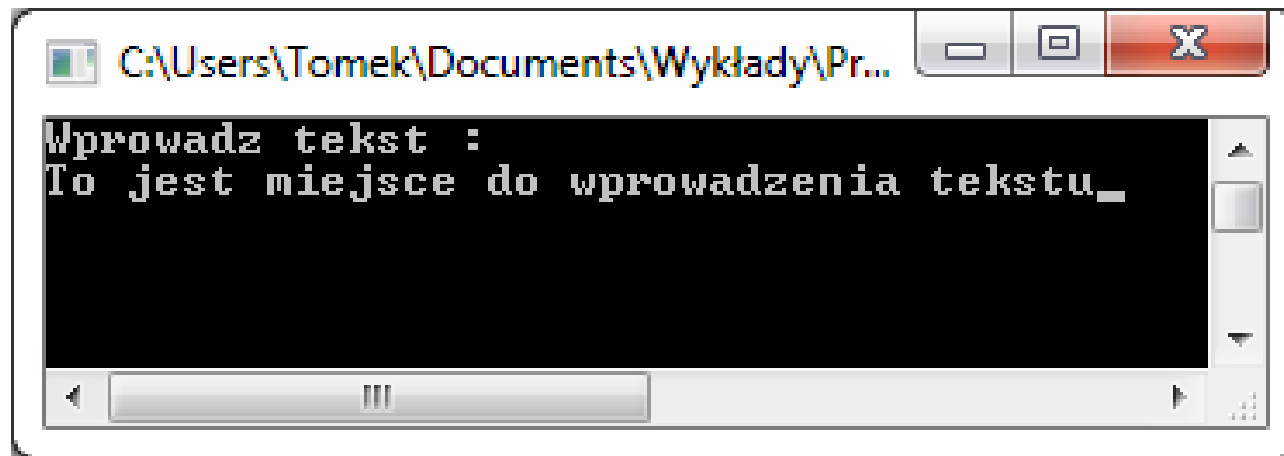
# Tryb tekstowy-przykład

```
#include <stdio.h>
#include <conio.h>
int main()
{
    FILE *f;
    char zn;
    f = fopen ("test35.txt","w");
    printf("\Wprowadz tekst :\n");
    while ( (zn=getche()) != '\r')
        putc(zn,f);
    fclose(f);
    return 0;
}
```

Zamknięcie pliku



# Tryb tekstowy-przykład c.d.




p33.c | p34.c | p35.c | test35.txt

To jest miejsce do wprowadzenia tekstu

# Co musimy zapamiętać

```
f = fopen ("test35.txt","w");
```



Nazwa pliku wraz  
ze ścieżką

Tryb otwarcia pliku

# Nazwa pliku w fopen

- „test35.txt”
- „c:\\test35.txt”
- „c:\\temp\\test35.txt”



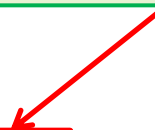
# Parametry otwarcia pliku

Tryb	Opis
"r"	Otwiera plik tekstowy do czytania
"w"	Otwiera plik tekstowy do zapisu. Jeżeli plik istnieje, usuwa zawartość i umieszcza nowe dane. Jeżeli plik nie istnieje, zostaje utworzony.
"a"	Otwiera plik do zapisu. Jeżeli plik istnieje, dopisuje nowe dane na końcu istniejących danych. Jeżeli plik nie istnieje zostaje utworzony.
"r+"	Otwiera plik tekstowy do uaktualnienia, zezwala na zapis i czytanie
"w+"	Otwiera plik tekstowy do uaktualnienia, zezwala na zapis i czytanie. Jeżeli plik istnieje, usuwa zawartość. Jeżeli plik nie istnieje jest tworzony.
"a+"	Otwiera plik tekstowy do uaktualnienia, zezwala na zapis i czytanie. Jeżeli plik istnieje, dopisuje nowe dane na końcu. Jeżeli plik nie istnieje jest tworzony. Odczyt obejmuje cały plik, zapis polega na dodawaniu nowego tekstu.
"rb", "wb", "ab", "rb+", "r+b", "wb+", "w+b", "ab+", "a+b"	Wymienione specyfikatory mają takie samo znaczenie jak powyższe, dotyczą <b>plików binarnych</b>

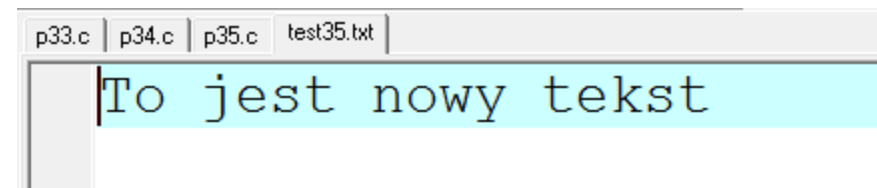
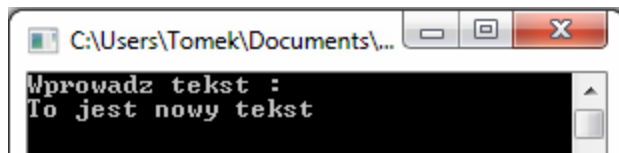
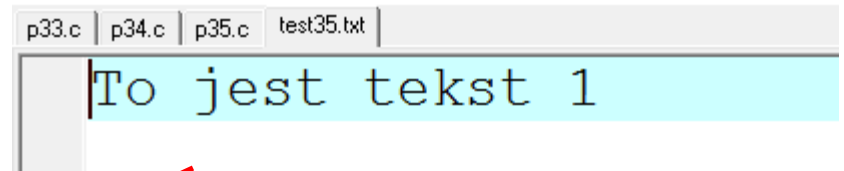
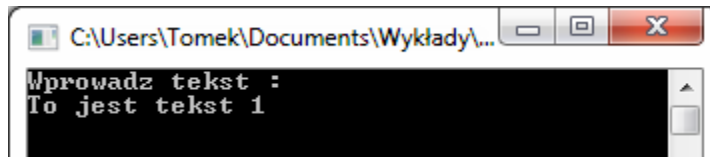
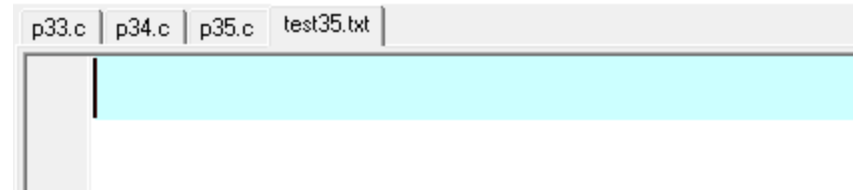
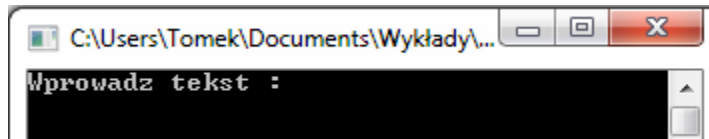
# Tryb tekstowy – przykład 2

```
#include <stdio.h>
#include <conio.h>
int main()
{
    FILE *f;
    char zn;
    f = fopen ("test35.txt","w");
    printf("Wprowadz tekst :\n");
    while ( (zn=getche()) != '\r')
        putc(zn,f);
    fclose(f);
    return 0;
}
```

Modyfikujemy tryb otwarcia pliku



# `f = fopen ("test35.txt","w");`



# f = fopen ("test35.txt","a");

```
C:\Users\Tomek\Documents\Wykłady\...  
Wprowadz tekst :
```

```
p33.c | p34.c | p35.c | test35.txt  
|
```

```
C:\Users\Tomek\Documents\Wykłady\...  
Wprowadz tekst :  
To jest tekst 1
```

```
p33.c | p34.c | p35.c | test35.txt  
|  
To jest tekst 1
```

```
C:\Users\Tomek\Documents\...  
Wprowadz tekst :  
To jest nowy tekst
```

```
p33.c | p34.c | p35.c | [*] test35.txt  
|  
To jest tekst 1  
To jest nowy tekst
```

# Kontrola otwarcia pliku

- Jak sprawdzić czy plik został poprawnie otwarty ???
- Pamiętamy .... od funkcji fopen otrzymujemy wskaźnik na plik;
- Czyli wystarczy sprawdzić czy mamy „prawidłowy” wskaźnik.

# Kontrola otwarcia pliku

```
f = fopen("test35.txt","r");  
if ( f == NULL) ←  
{  
    printf("\n Nie moge otworzyc pliku");  
    exit(1);  
}else{  
    //operacje na pliku  
    fclose(f);  
}
```

# Wprowadzanie łańcuchów

```
f = fopen("test35.txt","a");  
char napis[100];  
while (strlen ( gets(napis)) > 0 )  
{  
    fputs(napis,f);  
    fputs("\n",f);  
}
```

# Czytanie łańcuchów z pliku

```
while (fgets ( napis, 80, f) != NULL )  
    printf("%s", napis);
```



# Formatowanie zapisu do pliku

```
int a; float b;  
fprintf(f, "\n%d %f", a, b);
```

```
int num;  
fscanf(f, "%d", &num);
```

# Zapis i odczyt do pliku w trybie binarnym

- `size_t fread(void *ptr, size_t size, size_t n, FILE *stream);`

Wskaźnik na miejsce z danymi

- `size_t fwrite(void *ptr, size_t size, size_t n, FILE`

# Zapis i odczyt do pliku w trybie binarnym

- `size_t fread(void *ptr, size_t size, size_t n, FILE *stream);`

Długość jednostkowa pozycji danych np. `sizeof(int)`

- `size_t fwrite(void *ptr, size_t size, size_t n, FILE`

# Zapis i odczyt do pliku w trybie binarnym

- `size_t fread(void *ptr, size_t size, size_t n, FILE *stream);`



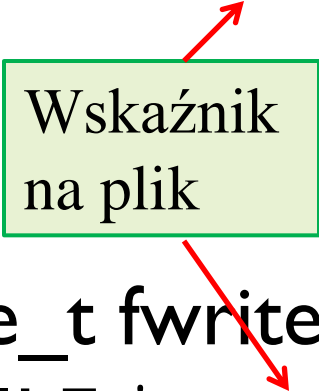
Ilość  
danych

- `size_t fwrite(void *ptr, size_t size, size_t n, FILE *stream);`

# Zapis i odczyt do pliku w trybie binarnym

- `size_t fread(void *ptr, size_t size, size_t n, FILE *stream);`

Wskaźnik  
na plik



- `size_t fwrite(void *ptr, size_t size, size_t n, FILE *stream);`

# Przykład

```
#include <stdio.h>
#include <conio.h>
int tab[5] = {1,2,3,4,5} ;
int main()
{
    FILE *f;
    f = fopen("test.dat", "wb");
    fwrite(tab, sizeof(tab), 1, f);
    fclose(f);
    puts("zapisano");
    getch();
    return 0;
}
```

# Przykład

```
#include <stdio.h>
```

```
#include <conio.h>
```

```
int tab[5] = {1,2,3,4,5} ;
```

```
int main()
```

```
{
```

```
    FILE *f;
```

```
    f = fopen("test.dat", "wb");
```

```
    fwrite(tab, sizeof(tab), 1, f);
```

```
    fclose(f);
```

```
    puts("zapisano");
```

```
    getch();
```

```
    return 0;
```

```
}
```

```
    fwrite(tab, sizeof(int), 5, f);
```





# KONIEC