Óbudai Egyetem
Kandó Kálmán Villamosmérnöki Kar

C programozási nyelv
Utasítások I.

Dr. Schuster György

2017. december 28.

`if` utasítás

`if` utasítás

Adott feltételtől függően egy programrészlet végrehajtásra kerül.

`if` utasítás

Adott feltételtől függően egy programrészlet végrehajtásra kerül.

```
if(kifejezés) ...
```

`if` utasítás

Adott feltételtől függően egy programrészlet végrehajtásra kerül.

```
if(kifejezés) ...
```

Ha a kifejezés igaz

`if` utasítás

Adott feltételtől függően egy programrészlet végrehajtásra kerül.

```
if(kifejezés) ...
```

Ha a kifejezés igaz (relációs jellegű),

`if` utasítás

Adott feltételtől függően egy programrészlet végrehajtásra kerül.

```
if(kifejezés) ...
```

Ha a kifejezés igaz (relációs jellegű), akkor az ”**igaz**” ág végrehajtásra kerül.

`if` utasítás

Adott feltételtől függően egy programrészlet végrehajtásra kerül.

```
if(kifejezés) ...
```

Ha a kifejezés igaz (relációs jellegű), akkor az ”**igaz**” ág végrehajtásra kerül.
Ha a kifejezés hamis,

`if` utasítás

Adott feltételtől függően egy programrészlet végrehajtásra kerül.

```
if(kifejezés) ...
```

Ha a kifejezés igaz (relációs jellegű), akkor az ”**igaz**” ág végrehajtásra kerül.
Ha a kifejezés hamis, akkor az ”**igaz**” ágat a program nem hajtja végre.

`if` utasítás

Adott feltételtől függően egy programrészlet végrehajtásra kerül.

```
if(kifejezés) ...
```

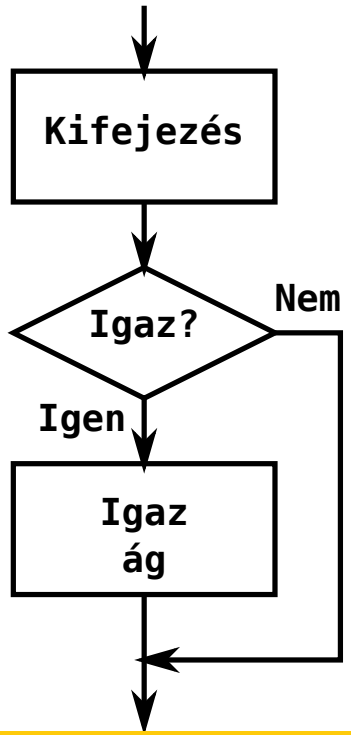
Ha a kifejezés igaz (relációs jellegű), akkor az ”**igaz**” ág végrehajtásra kerül.
Ha a kifejezés hamis, akkor az ”**igaz**” ágat a program nem hajtja végre.
Folyamatábrával:

`if` utasítás

Adott feltételtől függően egy programrészlet végrehajtásra kerül.

```
if(kifejezés) ...
```

Ha a kifejezés igaz (relációs jellegű), akkor az ”**igaz**” ág végrehajtásra kerül.
Ha a kifejezés hamis, akkor az ”**igaz**” ágat a program nem hajtja végre.
Folyamatábrával:



`if` utasítás az igaz ág

Az igaz ág egy logikai blokk,

`if` utasítás az igaz ág

Az igaz ág egy logikai blokk, amely lehet:

`if` utasítás az igaz ág

Az igaz ág egy logikai blokk, amely lehet:

- egyetlen pontosvesszővel (`;`) lezárt kifejezés,

`if` utasítás az igaz ág

Az igaz ág egy logikai blokk, amely lehet:

- egyetlen pontosvesszővel (`;`) lezárt kifejezés,
  `if(relkif) ...;`

`if` utasítás az igaz ág

Az igaz ág egy logikai blokk, amely lehet:

- egyetlen pontosvesszővel (`;`) lezárt kifejezés,
  `if(relkif) ...;`
- több kifejezés, amelyet `{ }` operátor pár fog közre.

`if` utasítás az igaz ág

Az igaz ág egy logikai blokk, amely lehet:

- egyetlen pontosvesszővel (`;`) lezárt kifejezés,
  `if(relkif) ...;`
- több kifejezés, amelyet `{ }` operátor pár fog közre.

```
if(relkif)
  {
  ...;
  ...;
  ...;
    ⋮
  }
```

`if` utasítás tipikus hibák

`if` utasítás tipikus hibák

Fölösleges pontosvessző!!!

`if` utasítás tipikus hibák

Fölösleges pontosvessző!!! Nagyon ”sunyi” hiba.

`if` utasítás tipikus hibák

Fölösleges pontosvessző!!! Nagyon ”sunyi” hiba.

- pontosvessző az `if` zárójele után:

`if` utasítás tipikus hibák

Fölösleges pontosvessző!!! Nagyon ”sunyi” hiba.

- pontosvessző az `if` zárójele után:

```
if(relkif);
  {
    ⋮
  }
```

`if` utasítás tipikus hibák

Fölösleges pontosvessző!!! Nagyon ”sunyi” hiba.

- pontosvessző az `if` zárójele után:

```
if(relkif);
  {
    ⋮
  }
```

  Nehéz észrevenni!!

- pontosvessző az igaz ág lezáró }-le után:

`if` utasítás tipikus hibák

Fölösleges pontosvessző!!! Nagyon ”sunyi” hiba.

- pontosvessző az `if` zárójele után:

```
if(relkif);
  {
    ⋮
  }
```

  Nehéz észrevenni!!

- pontosvessző az igaz ág lezáró }-le után:

```
if(relkif)
  {
    ⋮
  };
```

`if` utasítás **tipikus hibák**

Fölösleges pontosvessző!!! Nagyon ”sunyi” hiba.

- pontosvessző az `if` zárójele után:

```
if(relkif);
  {
    ⋮
  }
```

  Nehéz észrevenni!!

- pontosvessző az igaz ág lezáró }-le után:

```
if(relkif)
  {
    ⋮
  };
```

  `else` esetén hibaüzenet.

Mi is az igaz

Mi is az igaz

Egy kifejezés igaz, ha értéke nem 0.

Mi is az igaz

Egy kifejezés igaz, ha értéke nem 0.

Egy kifejezés hamis, ha értéke 0.

Mi is az igaz

Egy kifejezés igaz, ha értéke nem 0.

Egy kifejezés hamis, ha értéke 0.

Tehát a

`if(a!=0)` ... kifejezés helyettesíthető az

Mi is az igaz

Egy kifejezés igaz, ha értéke nem 0.

Egy kifejezés hamis, ha értéke 0.

Tehát a

`if(a!=0)` ... kifejezés helyettesíthető az `if(a)` ... kifejezéssel.

Mi is az igaz

Egy kifejezés igaz, ha értéke nem 0.

Egy kifejezés hamis, ha értéke 0.

Tehát a

`if(a!=0)` ... kifejezés helyettesíthető az `if(a)` ... kifejezéssel.

és

Mi is az igaz

Egy kifejezés igaz, ha értéke nem 0.

Egy kifejezés hamis, ha értéke 0.

Tehát a

`if(a!=0)` ... kifejezés helyettesíthető az `if(a)` ... kifejezéssel.

és

`if(a==0)` ... kifejezés helyettesíthető az

Mi is az igaz

Egy kifejezés igaz, ha értéke nem 0.

Egy kifejezés hamis, ha értéke 0.

Tehát a

`if(a!=0)` ... kifejezés helyettesíthető az `if(a)` ... kifejezéssel.

és

`if(a==0)` ... kifejezés helyettesíthető az `if(!a)` ... kifejezéssel.

`if - else` szerkeszet

`if - else` szerkeszet

Ha a feltétel **igaz**, akkor egy adott programrészlet kerül végrehajtásra,

`if - else` szerkeszet

Ha a feltétel **igaz**, akkor egy adott programrészlet kerül végrehajtásra, ha a feltétel **hamis**, akkor egy másik.

`if - else` szerkeszet

Ha a feltétel **igaz**, akkor egy adott programrészlet kerül végrehajtásra, ha a feltétel **hamis**, akkor egy másik.
Az igaz ág az `if` ág, mint előzőleg.

`if - else` szerkeszet

Ha a feltétel **igaz**, akkor egy adott programrészlet kerül végrehajtásra, ha a feltétel **hamis**, akkor egy másik.

Az igaz ág az `if` ág, mint előzőleg. A hamis ág az **`else`** ág.

`if - else` szerkeszet

Ha a feltétel **igaz**, akkor egy adott programrészlet kerül végrehajtásra, ha a feltétel **hamis**, akkor egy másik.
Az igaz ág az `if` ág, mint előzőleg. A hamis ág az **`else`** ág. Vagyis:

`if - else` szerkeszet

Ha a feltétel **igaz**, akkor egy adott programrészlet kerül végrehajtásra, ha a feltétel **hamis**, akkor egy másik.
Az igaz ág az `if` ág, mint előzőleg. A hamis ág az **`else`** ág. Vagyis:

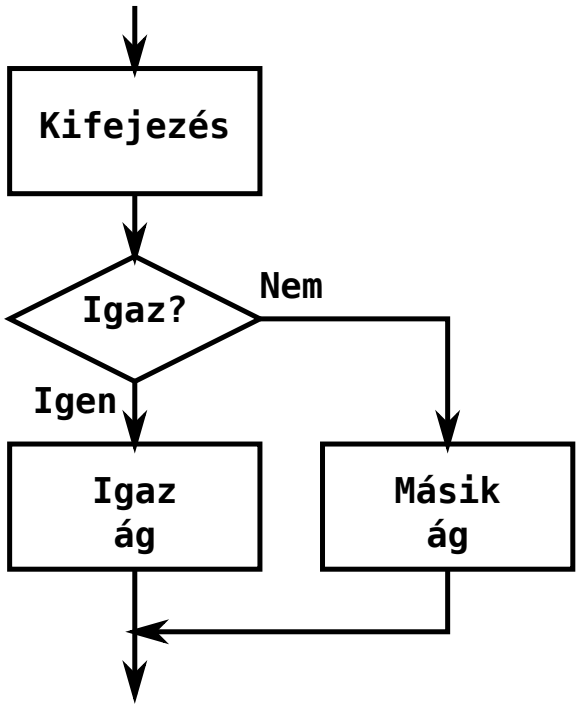
```
if(relkif) igazag
```

`if - else` szerkeszet

Ha a feltétel **igaz**, akkor egy adott programrészlet kerül végrehajtásra, ha a feltétel **hamis**, akkor egy másik.
Az igaz ág az `if` ág, mint előzőleg. A hamis ág az **`else`** ág. Vagyis:

```
if(relkif) igazag

else masikag
```



A másik ág

A másik ág

Az másik ág egy logikai blokk,

A másik ág

Az másik ág egy logikai blokk, amely lehet:

A másik ág

Az másik ág egy logikai blokk, amely lehet:

- egyetlen pontosvesszővel (;) lezárt kifejezés,

A másik ág

Az másik ág egy logikai blokk, amely lehet:

- egyetlen pontosvesszővel (`;`) lezárt kifejezés,
  `else ...;`

A másik ág

Az másik ág egy logikai blokk, amely lehet:

- egyetlen pontosvesszővel (`;`) lezárt kifejezés,
  `else ...;`
- több kifejezés, amelyet `{ }` operátor pár fog közre.

A másik ág

Az másik ág egy logikai blokk, amely lehet:

- egyetlen pontosvesszővel (`;`) lezárt kifejezés,
  ```
  else ...;
  ```
- több kifejezés, amelyet `{ }` operátor pár fog közre.
  ```
  else
    {
    ...;
    ...;
    ...;
      ⋮
    }
  ```

Példa

Beolvasunk egy ASCII karaktert.

Példa

Beolvasunk egy ASCII karaktert. Megnézzük, hogy hexa számot reprezentál-e.

Példa

Beolvasunk egy ASCII karaktert. Megnézzük, hogy hexa számot reprezentál-e.
Ha igen átalakítjuk egy `int` változóvá, amely a hexa szám értéke.

Példa

Beolvasunk egy ASCII karaktert. Megnézzük, hogy hexa számot reprezentál-e.
Ha igen átalakítjuk egy `int` változóvá, amely a hexa szám értéke.
Ha nem -1 -el térünk vissza.

Példa

Beolvasunk egy ASCII karaktert. Megnézzük, hogy hexa számot reprezentál-e.
Ha igen átalakítjuk egy `int` változóvá, amely a hexa szám értéke.
Ha nem -1 -el térünk vissza.
Kihasználjuk, az ASCII kódolás sajátosságait:

Példa

Beolvasunk egy ASCII karaktert. Megnézzük, hogy hexa számot reprezentál-e.
Ha igen átalakítjuk egy `int` változóvá, amely a hexa szám értéke.
Ha nem -1 -el térünk vissza.
Kihasználjuk, az ASCII kódolás sajátosságait:

`'0'`	30_h
`'1'`	31_h
⋮	
`'9'`	39_h

Példa

Beolvasunk egy ASCII karaktert. Megnézzük, hogy hexa számot reprezentál-e.
Ha igen átalakítjuk egy `int` változóvá, amely a hexa szám értéke.
Ha nem -1 -el térünk vissza.
Kihasználjuk, az ASCII kódolás sajátosságait:

`'0'`	30_h
`'1'`	31_h
⋮	
`'9'`	39_h
`'A'`	41_h
`'B'`	42_h
⋮	
`'F'`	46_h

Példa

Beolvasunk egy ASCII karaktert. Megnézzük, hogy hexa számot reprezentál-e.
Ha igen átalakítjuk egy `int` változóvá, amely a hexa szám értéke.
Ha nem -1 -el térünk vissza.
Kihasználjuk, az ASCII kódolás sajátosságait:

`'0'`	30_h
`'1'`	31_h
⋮	
`'9'`	39_h
`'A'`	41_h
`'B'`	42_h
⋮	
`'F'`	46_h
`'a'`	61_h
`'b'`	62_h
⋮	
`'f'`	66_h

Példa

Példa

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

Példa

```
#include <stdio.h>
int main(void)
```

Példa

```
#include <stdio.h>
int main(void)
    {
    char c;
    int i;
```

Példa

```
#include <stdio.h>
int main(void)
    {
    char c;
    int i;
    scanf("%c",&c);
```

Példa

```
#include <stdio.h>
int main(void)
    {
    char c;
    int i;
    scanf("%c",&c);
    if(c>='0' && c<='9') i=c-'0';
```

Példa

```
#include <stdio.h>
int main(void)
    {
    char c;
    int i;
    scanf("%c",&c);
    if(c>='0' && c<='9') i=c-'0';
    else
       {
       if(c>='A' && c<='F') i=c-'A'+10;
```

Példa

```
#include <stdio.h>
int main(void)
    {
    char c;
    int i;
    scanf("%c",&c);
    if(c>='0' && c<='9') i=c-'0';
    else
       {
       if(c>='A' && c<='F') i=c-'A'+10;
       else
          {
          if(c>='a' && c<='f') i=c-'a'+10;
```

Példa

```
#include <stdio.h>
int main(void)
    {
    char c;
    int i;
    scanf("%c",&c);
    if(c>='0' && c<='9') i=c-'0';
    else
       {
       if(c>='A' && c<='F') i=c-'A'+10;
       else
          {
          if(c>='a' && c<='f') i=c-'a'+10;
          else i=-1;
```

Példa

```
#include <stdio.h>
int main(void)
    {
    char c;
    int i;
    scanf("%c",&c);
    if(c>='0' && c<='9') i=c-'0';
    else
       {
       if(c>='A' && c<='F') i=c-'A'+10;
       else
          {
          if(c>='a' && c<='f') i=c-'a'+10;
          else i=-1;
          }
       }
    printf("%i",i);
```

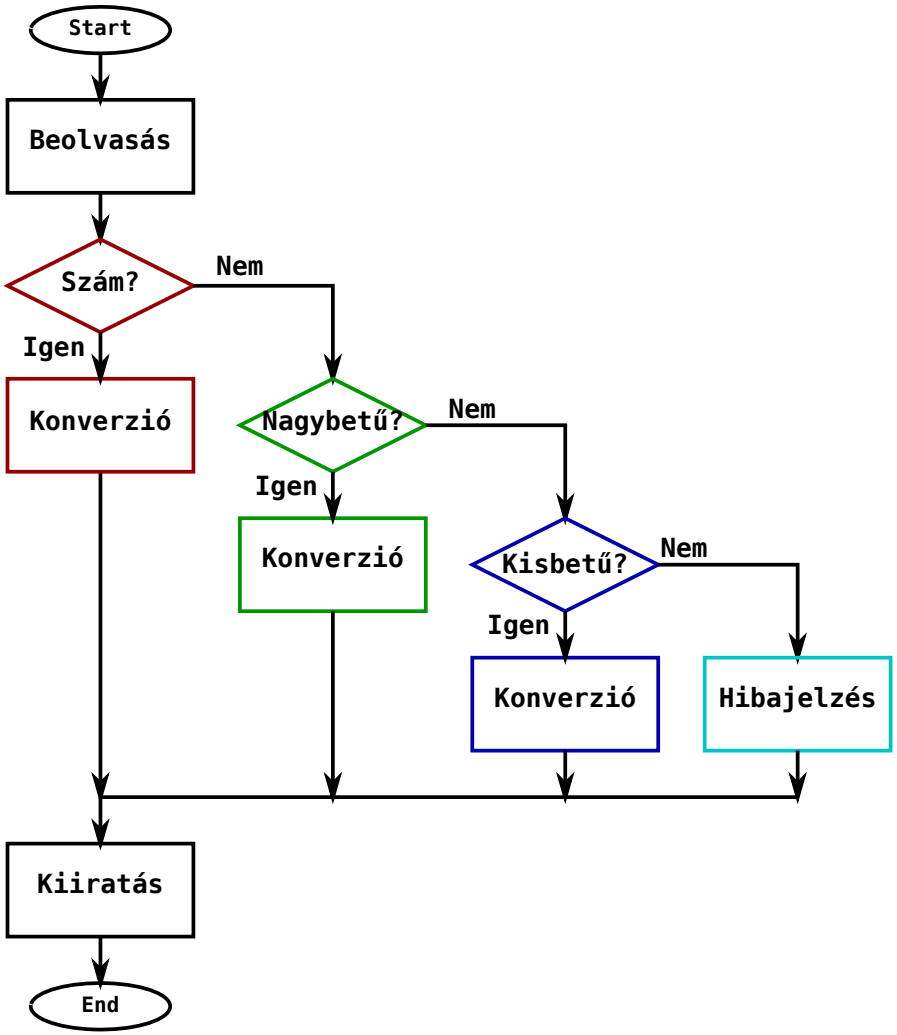
Példa

```
#include <stdio.h>
int main(void)
    {
    char c;
    int i;
    scanf("%c",&c);
    if(c>='0' && c<='9') i=c-'0';
    else
       {
       if(c>='A' && c<='F') i=c-'A'+10;
       else
          {
          if(c>='a' && c<='f') i=c-'a'+10;
          else i=-1;
          }
       }
    printf("%i",i);
    return 0;
```

Példa

```
#include <stdio.h>
int main(void)
    {
    char c;
    int i;
    scanf("%c",&c);
    if(c>='0' && c<='9') i=c-'0';
    else
       {
       if(c>='A' && c<='F') i=c-'A'+10;
       else
          {
          if(c>='a' && c<='f') i=c-'a'+10;
          else i=-1;
          }
       }
    printf("%i",i);
    return 0;
    }
```

Példa


Start
Beolvasás
Szám?
Nem
Igen
Konverzió
Nagybetű?
Nem
Igen
Konverzió
Kisbetű?
Nem
Igen
Konverzió
Hibajelzés
Kiiratás
End

`switch-case` szerkezet

`switch-case` szerkezet

Egy adott változó értékétől függő többszörös elágazást tesz lehetővé.

`switch-case` szerkezet

Egy adott változó értékétől függő többszörös elágazást tesz lehetővé.
A változó csak **egész jellegű lehet!**

`switch-case` szerkezet

Egy adott változó értékétől függő többszörös elágazást tesz lehetővé.
A változó csak **egész jellegű lehet!**

```
switch(valt)
```

`switch-case` szerkezet

Egy adott változó értékétől függő többszörös elágazást tesz lehetővé.
A változó csak **egész jellegű lehet!**

```
switch(valt)
  {
  case ert1:   ...;
```

`switch-case` szerkezet

Egy adott változó értékétől függő többszörös elágazást tesz lehetővé.
A változó csak **egész jellegű lehet!**

```
switch(valt)
  {
  case ert1:   ...;
               ...;
               ...;
```

switch-case szerkezet

Egy adott változó értékétől függő többszörös elágazást tesz lehetővé.
A változó csak **egész jellegű lehet!**

```
switch(valt)
  {
  case ert1:   ...;
               ...;
               ...;
               break;
```

switch-case szerkezet

Egy adott változó értékétől függő többszörös elágazást tesz lehetővé.
A változó csak **egész jellegű lehet!**

```
switch(valt)
  {
  case ert1:   ...;
               ...;
               ...;
               break;
  case ert2:   ...;
```

switch-case szerkezet

Egy adott változó értékétől függő többszörös elágazást tesz lehetővé.
A változó csak **egész jellegű lehet!**

```
switch(valt)
  {
  case ert1:   ...;
               ...;
               ...;
               break;
  case ert2:   ...;
               ...;
               ...;
               break;
                 ⋮
```

switch-case szerkezet

Egy adott változó értékétől függő többszörös elágazást tesz lehetővé.
A változó csak **egész jellegű lehet!**

```
switch(valt)
  {
  case ert1:   ...;
               ...;
               ...;
               break;
  case ert2:   ...;
               ...;
               ...;
               break;
                 :
  default:     ...;
               ...;
  }
```

`break` hatása

`break` hatása

A program az `ert1`-re ugrik.

`break` hatása

A program az `ert1`-re ugrik.

```
case ert1:   ...;
             ...;
             ...;
```

`break` hatása

A program az `ert1`-re ugrik.

```
case ert1:   ...;
             ...;
             ...;
             break;
```

`break` hatása

A program az `ert1`-re ugrik.

```
case ert1:  ...;
            ...;
            ...;
            break;
case ert2:  ...;
            ...;
            ...;
```

`break` hatása

A program az `ert1`-re ugrik.

```
case ert1:  ...;
            ...;
            ...;
            break;
case ert2:  ...;
            ...;
            ...;
```

A **`break`** hatására kilép a `switch-case` szerkezetből.

`break` hatása

A program az `ert1`-re ugrik.

```
case ert1:  ...;
            ...;
            ...;
            break;
case ert2:  ...;
            ...;
            ...;
```

A **`break`** hatására kilép a `switch-case` szerkezetből.

```
case ert1:  ...;
            ...;
            ...;
```

`break` hatása

A program az `ert1`-re ugrik.

```
case ert1:  ...;
            ...;
            ...;
            break;
case ert2:  ...;
            ...;
            ...;
```

A **`break`** hatására kilép a `switch-case` szerkezetből.

```
case ert1:  ...;
            ...;
            ...;
case ert2:  ...;
            ...;
            ...;
```

`break` hatása

A program az `ert1`-re ugrik.

```
case ert1:   ...;
             ...;
             ...;
             break;
case ert2:   ...;
             ...;
             ...;
```

A **`break`** hatására kilép a `switch-case` szerkezetből.

```
case ert1:   ...;
             ...;
             ...;
case ert2:   ...;
             ...;
             ...;
```

Ha nincs **`break`**, a program folytatódik.

default

default

Mi történik akkor ha a változó értéke egyik `case`-nek sem felel meg?

default

Mi történik akkor ha a változó értéke egyik `case`-nek sem felel meg?

- nem kell semmit csinálni,

default

Mi történik akkor ha a változó értéke egyik `case`-nek sem felel meg?

- nem kell semmit csinálni, kilépünk a `switch-case`-ből,

default

Mi történik akkor ha a változó értéke egyik `case`-nek sem felel meg?

- nem kell semmit csinálni, kilépünk a `switch-case`-ből, nem kell a **`default`** utasítás,

default

Mi történik akkor ha a változó értéke egyik `case`-nek sem felel meg?

- nem kell semmit csinálni, kilépünk a `switch-case`-ből, nem kell a **`default`** utasítás,
- valamit kell csinálni,

default

Mi történik akkor ha a változó értéke egyik `case`-nek sem felel meg?

- nem kell semmit csinálni, kilépünk a `switch-case`-ből, nem kell a **`default`** utasítás,
- valamit kell csinálni, ekkor használjuk a **`default`**-ot,

default

Mi történik akkor ha a változó értéke egyik `case`-nek sem felel meg?

- nem kell semmit csinálni, kilépünk a `switch-case`-ből, nem kell a **`default`** utasítás,
- valamit kell csinálni, ekkor használjuk a **`default`**-ot, ide írjuk azokat a kifejezéseket, amelyeket ekkor kell végrehajtani.

default

Mi történik akkor ha a változó értéke egyik `case`-nek sem felel meg?

- nem kell semmit csinálni, kilépünk a `switch-case`-ből, nem kell a **`default`** utasítás,
- valamit kell csinálni, ekkor használjuk a **`default`**-ot, ide írjuk azokat a kifejezéseket, amelyeket ekkor kell végrehajtani.

Tehát a `default` nem kötelező.

default

Mi történik akkor ha a változó értéke egyik `case`-nek sem felel meg?

- nem kell semmit csinálni, kilépünk a `switch-case`-ből, nem kell a **`default`** utasítás,
- valamit kell csinálni, ekkor használjuk a **`default`**-ot, ide írjuk azokat a kifejezéseket, amelyeket ekkor kell végrehajtani.

Tehát a `default` nem kötelező.
A `default` helye a `switch-case` végén van.

Példa

Példa

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

Példa

```
#include <stdio.h>
int main(void)
```

Példa

```
#include <stdio.h>
int main(void)
    {
    char c; int i;
```

Példa

```
#include <stdio.h>
int main(void)
    {
    char c; int i;
    scanf("%c",&c);
```

Példa

```
#include <stdio.h>
int main(void)
    {
    char c; int i;
    scanf("%c",&c);
    switch(c)
```

Példa

```
#include <stdio.h>
int main(void)
    {
    char c; int i;
    scanf("%c",&c);
    switch(c)
       {
       case '0':  case '1':  case '2':  case '3':
       case '4':  case '5':  case '6':  case '7':
       case '8':  case '9':  i=c-'0'; break;
```

Példa

```
#include <stdio.h>
int main(void)
    {
    char c; int i;
    scanf("%c",&c);
    switch(c)
       {
       case '0':  case '1':  case '2':  case '3':
       case '4':  case '5':  case '6':  case '7':
       case '8':  case '9':  i=c-'0'; break;
       case 'A': case 'B': case 'C':
       case 'D': case 'E': case 'F': i=c-'A'+10; break;
```

Példa

```
#include <stdio.h>
int main(void)
    {
    char c; int i;
    scanf("%c",&c);
    switch(c)
       {
       case '0': case '1': case '2': case '3':
       case '4': case '5': case '6': case '7':
       case '8': case '9': i=c-'0'; break;
       case 'A': case 'B': case 'C':
       case 'D': case 'E': case 'F': i=c-'A'+10; break;
       case 'a': case 'b': case 'c':
       case 'd': case 'e': case 'f': i=c-'a'+10; break;
```

Példa

```
#include <stdio.h>
int main(void)
    {
    char c; int i;
    scanf("%c",&c);
    switch(c)
       {
       case '0':  case '1':  case '2':  case '3':
       case '4':  case '5':  case '6':  case '7':
       case '8':  case '9':  i=c-'0'; break;
       case 'A': case 'B': case 'C':
       case 'D': case 'E': case 'F': i=c-'A'+10; break;
       case 'a':  case 'b':  case 'c':
       case 'd':  case 'e':  case 'f':  i=c-'a'+10; break;
       deafult:  i=-1;
```

Példa

```
#include <stdio.h>
int main(void)
    {
    char c; int i;
    scanf("%c",&c);
    switch(c)
       {
       case '0': case '1': case '2': case '3':
       case '4': case '5': case '6': case '7':
       case '8': case '9': i=c-'0'; break;
       case 'A': case 'B': case 'C':
       case 'D': case 'E': case 'F': i=c-'A'+10; break;
       case 'a': case 'b': case 'c':
       case 'd': case 'e': case 'f': i=c-'a'+10; break;
       deafult: i=-1;
       }
    printf("%i",i);
```

Példa

```
#include <stdio.h>
int main(void)
    {
    char c; int i;
    scanf("%c",&c);
    switch(c)
       {
       case '0': case '1': case '2': case '3':
       case '4': case '5': case '6': case '7':
       case '8': case '9': i=c-'0'; break;
       case 'A': case 'B': case 'C':
       case 'D': case 'E': case 'F': i=c-'A'+10; break;
       case 'a': case 'b': case 'c':
       case 'd': case 'e': case 'f': i=c-'a'+10; break;
       deafult: i=-1;
       }
    printf("%i",i);
    return 0;
```

Példa

```
#include <stdio.h>
int main(void)
    {
    char c; int i;
    scanf("%c",&c);
    switch(c)
       {
       case '0': case '1': case '2': case '3':
       case '4': case '5': case '6': case '7':
       case '8': case '9': i=c-'0'; break;
       case 'A': case 'B': case 'C':
       case 'D': case 'E': case 'F': i=c-'A'+10; break;
       case 'a': case 'b': case 'c':
       case 'd': case 'e': case 'f': i=c-'a'+10; break;
       deafult: i=-1;
       }
    printf("%i",i);
    return 0;
    }
```

Kérdések

Mire való az `if` utasítás?

Kérdések

Mire való az `if` utasítás?

Adott feltételnek megfelelően elágazást tesz lehetővé.

Kérdések

Mire való az `if` utasítás?

Adott feltételnek megfelelően elágazást tesz lehetővé.

Milyen jellegű kifejezés van az `if` utasítás argumentumában?

Kérdések

Mire való az `if` utasítás?

Adott feltételnek megfelelően elágazást tesz lehetővé.

Milyen jellegű kifejezés van az `if` utasítás argumentumában?

Tulajdonképpen bármilyen, de a kiértékelése mindig igaz - nem igaz módon történik.

Kérdések

Mire való az `if` utasítás?

Adott feltételnek megfelelően elágazást tesz lehetővé.

Milyen jellegű kifejezés van az `if` utasítás argumentumában?

Tulajdonképpen bármilyen, de a kiértékelése mindig igaz - nem igaz módon történik.

Az `if` utasítás esetén mikor lép az igaz logikai blokkba a program?

Kérdések

Mire való az `if` utasítás?

Adott feltételnek megfelelően elágazást tesz lehetővé.

Milyen jellegű kifejezés van az `if` utasítás argumentumában?

Tulajdonképpen bármilyen, de a kiértékelése mindig igaz - nem igaz módon történik.

Az `if` utasítás esetén mikor lép az igaz logikai blokkba a program?

Ha az `if` argumentumának kiértékelése igaz értéket ad.

Kérdések

Mire való az `if` utasítás?

Adott feltételnek megfelelően elágazást tesz lehetővé.

Milyen jellegű kifejezés van az `if` utasítás argumentumában?

Tulajdonképpen bármilyen, de a kiértékelése mindig igaz - nem igaz módon történik.

Az `if` utasítás esetén mikor lép az igaz logikai blokkba a program?

Ha az `if` argumentumának kiértékelése igaz értéket ad.

Mi lehet `if` esetén az igaz ág?

Kérdések

Mire való az `if` utasítás?
Adott feltételnek megfelelően elágazást tesz lehetővé.
Milyen jellegű kifejezés van az `if` utasítás argumentumában?
Tulajdonképpen bármilyen, de a kiértékelése mindig igaz - nem igaz módon történik.
Az `if` utasítás esetén mikor lép az igaz logikai blokkba a program?
Ha az `if` argumentumának kiértékelése igaz értéket ad.
Mi lehet `if` esetén az igaz ág?
Egy kifejezés `;`-vel lezárva, vagy {} operátor párok közé zárt kifezés, vagy kifejezések.

Kérdések

Mire való az `if` utasítás?
Adott feltételnek megfelelően elágazást tesz lehetővé.
Milyen jellegű kifejezés van az `if` utasítás argumentumában?
Tulajdonképpen bármilyen, de a kiértékelése mindig igaz - nem igaz módon történik.
Az `if` utasítás esetén mikor lép az igaz logikai blokkba a program?
Ha az `if` argumentumának kiértékelése igaz értéket ad.
Mi lehet `if` esetén az igaz ág?
Egy kifejezés `;`-vel lezárva, vagy {} operátor párok közé zárt kifezés, vagy kifejezések.
Mi a tipikus hiba `if` esetén?

Kérdések

Mire való az `if` utasítás?
Adott feltételnek megfelelően elágazást tesz lehetővé.
Milyen jellegű kifejezés van az `if` utasítás argumentumában?
Tulajdonképpen bármilyen, de a kiértékelése mindig igaz - nem igaz módon történik.
Az `if` utasítás esetén mikor lép az igaz logikai blokkba a program?
Ha az `if` argumentumának kiértékelése igaz értéket ad.
Mi lehet `if` esetén az igaz ág?
Egy kifejezés `;`-vel lezárva, vagy {} operátor párok közé zárt kifezés, vagy kifejezések.
Mi a tipikus hiba `if` esetén?
Az `if` argumentuma mögé kerül egy `;`.

Kérdések

Mire való az `if` utasítás?
Adott feltételnek megfelelően elágazást tesz lehetővé.
Milyen jellegű kifejezés van az `if` utasítás argumentumában?
Tulajdonképpen bármilyen, de a kiértékelése mindig igaz - nem igaz módon történik.
Az `if` utasítás esetén mikor lép az igaz logikai blokkba a program?
Ha az `if` argumentumának kiértékelése igaz értéket ad.
Mi lehet `if` esetén az igaz ág?
Egy kifejezés `;`-vel lezárva, vagy {} operátor párok közé zárt kifezés, vagy kifejezések.
Mi a tipikus hiba `if` esetén?
Az `if` argumentuma mögé kerül egy `;`.
Mi a szerepe az `else` ágnak?

Kérdések

Mire való az `if` utasítás?
Adott feltételnek megfelelően elágazást tesz lehetővé.
Milyen jellegű kifejezés van az `if` utasítás argumentumában?
Tulajdonképpen bármilyen, de a kiértékelése mindig igaz - nem igaz módon történik.
Az `if` utasítás esetén mikor lép az igaz logikai blokkba a program?
Ha az `if` argumentumának kiértékelése igaz értéket ad.
Mi lehet `if` esetén az igaz ág?
Egy kifejezés `;`-vel lezárva, vagy {} operátor párok közé zárt kifezés, vagy kifejezések.
Mi a tipikus hiba `if` esetén?
Az `if` argumentuma mögé kerül egy `;`.
Mi a szerepe az `else` ágnak?
Az `if` igaz ágát követve, akkor kerül végrehajtásra, ha az `if` feltétele hamis.

Kérdések

Mire való az `if` utasítás?
Adott feltételnek megfelelően elágazást tesz lehetővé.
Milyen jellegű kifejezés van az `if` utasítás argumentumában?
Tulajdonképpen bármilyen, de a kiértékelése mindig igaz - nem igaz módon történik.
Az `if` utasítás esetén mikor lép az igaz logikai blokkba a program?
Ha az `if` argumentumának kiértékelése igaz értéket ad.
Mi lehet `if` esetén az igaz ág?
Egy kifejezés `;`-vel lezárva, vagy {} operátor párok közé zárt kifezés, vagy kifejezések.
Mi a tipikus hiba `if` esetén?
Az `if` argumentuma mögé kerül egy `;`.
Mi a szerepe az `else` ágnak?
Az `if` igaz ágát követve, akkor kerül végrehajtásra, ha az `if` feltétele hamis.
Elágazás esetén milyen szabály célszerű követni lebegőpontos számok összehasonlításánál?

Kérdések

Mire való az `if` utasítás?
Adott feltételnek megfelelően elágazást tesz lehetővé.
Milyen jellegű kifejezés van az `if` utasítás argumentumában?
Tulajdonképpen bármilyen, de a kiértékelése mindig igaz - nem igaz módon történik.
Az `if` utasítás esetén mikor lép az igaz logikai blokkba a program?
Ha az `if` argumentumának kiértékelése igaz értéket ad.
Mi lehet `if` esetén az igaz ág?
Egy kifejezés `;`-vel lezárva, vagy {} operátor párok közé zárt kifezés, vagy kifejezések.
Mi a tipikus hiba `if` esetén?
Az `if` argumentuma mögé kerül egy `;`.
Mi a szerepe az `else` ágnak?
Az `if` igaz ágát követve, akkor kerül végrehajtásra, ha az `if` feltétele hamis.
Elágazás esetén milyen szabály célszerű követni lebegőpontos számok összehasonlításánál?
A kerekítési hibák miatt nem célszerű egyenlőséget, vagy nem egyenlőséget vizsgálni.

Kérdések

Mire való az `if` utasítás?
Adott feltételnek megfelelően elágazást tesz lehetővé.
Milyen jellegű kifejezés van az `if` utasítás argumentumában?
Tulajdonképpen bármilyen, de a kiértékelése mindig igaz - nem igaz módon történik.
Az `if` utasítás esetén mikor lép az igaz logikai blokkba a program?
Ha az `if` argumentumának kiértékelése igaz értéket ad.
Mi lehet `if` esetén az igaz ág?
Egy kifejezés `;`-vel lezárva, vagy {} operátor párok közé zárt kifezés, vagy kifejezések.
Mi a tipikus hiba `if` esetén?
Az `if` argumentuma mögé kerül egy `;`.
Mi a szerepe az `else` ágnak?
Az `if` igaz ágát követve, akkor kerül végrehajtásra, ha az `if` feltétele hamis.
Elágazás esetén milyen szabály célszerű követni lebegőpontos számok összehasonlításánál?
A kerekítési hibák miatt nem célszerű egyenlőséget, vagy nem egyenlőséget vizsgálni.

Kérdések

Mi a szerepe a **`switch`** utasításnak?

Kérdések

Mi a szerepe a **`switch`** utasításnak?
Egy egész jellegű változó előre meghatározott értékeire tud ugrásokat adott **`case`** pontokra végrehajtani.

Kérdések

Mi a szerepe a **`switch`** utasításnak?
Egy egész jellegű változó előre meghatározott értékeire tud ugrásokat adott **`case`** pontokra végrehajtani.
Mi lehet egy **`case`** után?

Kérdések

Mi a szerepe a **`switch`** utasításnak?

Egy egész jellegű változó előre meghatározott értékeire tud ugrásokat adott **`case`** pontokra végrehajtani.

Mi lehet egy **`case`** után?

Csak egy érték.

Kérdések

Mi a szerepe a **`switch`** utasításnak?

Egy egész jellegű változó előre meghatározott értékeire tud ugrásokat adott **`case`** pontokra végrehajtani.

Mi lehet egy **`case`** után?

Csak egy érték.

Mi a szerepe a **`default`** utasításnak?

Kérdések

Mi a szerepe a **`switch`** utasításnak?

Egy egész jellegű változó előre meghatározott értékeire tud ugrásokat adott **`case`** pontokra végrehajtani.

Mi lehet egy **`case`** után?

Csak egy érték.

Mi a szerepe a **`default`** utasításnak?

Ha egy **`swith - case`** szerkezet változója egy olyan értéket kap, amelyre nincs **`case`** ág, akkor a program a **`default`** ágra ugrik.

Kérdések

Mi a szerepe a **switch** utasításnak?
Egy egész jellegű változó előre meghatározott értékeire tud ugrásokat adott **case** pontokra végrehajtani.
Mi lehet egy **case** után?
Csak egy érték.
Mi a szerepe a **default** utasításnak?
Ha egy **swith - case** szerkezet változója egy olyan értéket kap, amelyre nincs **case** ág, akkor a program a **default** ágra ugrik.
Mi történik, ha **swith - case** szerkezet változója egy olyan értéket kap, amelyre nincs **case** ág, de nincs **default** ág?

Kérdések

Mi a szerepe a **switch** utasításnak?
Egy egész jellegű változó előre meghatározott értékeire tud ugrásokat adott **case** pontokra végrehajtani.
Mi lehet egy **case** után?
Csak egy érték.
Mi a szerepe a **default** utasításnak?
Ha egy **swith - case** szerkezet változója egy olyan értéket kap, amelyre nincs **case** ág, akkor a program a **default** ágra ugrik.
Mi történik, ha **swith - case** szerkezet változója egy olyan értéket kap, amelyre nincs **case** ág, de nincs **default** ág?
A program kilép a **swith - case** szerkezetből.

Kérdések

Mi a szerepe a **switch** utasításnak?
Egy egész jellegű változó előre meghatározott értékeire tud ugrásokat adott **case** pontokra végrehajtani.
Mi lehet egy **case** után?
Csak egy érték.
Mi a szerepe a **default** utasításnak?
Ha egy **swith - case** szerkezet változója egy olyan értéket kap, amelyre nincs **case** ág, akkor a program a **default** ágra ugrik.
Mi történik, ha **swith - case** szerkezet változója egy olyan értéket kap, amelyre nincs **case** ág, de nincs **default** ág?
A program kilép a **swith - case** szerkezetből.
Mi a szerepe a **break** utasításnak **swith - case** szerkezetben?

Kérdések

Mi a szerepe a **switch** utasításnak?
Egy egész jellegű változó előre meghatározott értékeire tud ugrásokat adott **case** pontokra végrehajtani.
Mi lehet egy **case** után?
Csak egy érték.
Mi a szerepe a **default** utasításnak?
Ha egy **swith - case** szerkezet változója egy olyan értéket kap, amelyre nincs **case** ág, akkor a program a **default** ágra ugrik.
Mi történik, ha **swith - case** szerkezet változója egy olyan értéket kap, amelyre nincs **case** ág, de nincs **default** ág?
A program kilép a **swith - case** szerkezetből.
Mi a szerepe a **break** utasításnak **swith - case** szerkezetben?
Amikor a program az utasítást végrehajtja kilép a **swith - case** szerkezetből.

Kérdések

Mi a szerepe a **switch** utasításnak?
Egy egész jellegű változó előre meghatározott értékeire tud ugrásokat adott **case** pontokra végrehajtani.
Mi lehet egy **case** után?
Csak egy érték.
Mi a szerepe a **default** utasításnak?
Ha egy **swith - case** szerkezet változója egy olyan értéket kap, amelyre nincs **case** ág, akkor a program a **default** ágra ugrik.
Mi történik, ha **swith - case** szerkezet változója egy olyan értéket kap, amelyre nincs **case** ág, de nincs **default** ág?
A program kilép a **swith - case** szerkezetből.
Mi a szerepe a **break** utasításnak **swith - case** szerkezetben?
Amikor a program az utasítást végrehajtja kilép a **swith - case** szerkezetből.
Mi történi, ha a program az adott **case** után nem talál **break** utasítást?

Kérdések

Mi a szerepe a **switch** utasításnak?
Egy egész jellegű változó előre meghatározott értékeire tud ugrásokat adott **case** pontokra végrehajtani.
Mi lehet egy **case** után?
Csak egy érték.
Mi a szerepe a **default** utasításnak?
Ha egy **swith - case** szerkezet változója egy olyan értéket kap, amelyre nincs **case** ág, akkor a program a **default** ágra ugrik.
Mi történik, ha **swith - case** szerkezet változója egy olyan értéket kap, amelyre nincs **case** ág, de nincs **default** ág?
A program kilép a **swith - case** szerkezetből.
Mi a szerepe a **break** utasításnak **swith - case** szerkezetben?
Amikor a program az utasítást végrehajtja kilép a **swith - case** szerkezetből.
Mi történi, ha a program az adott **case** után nem talál **break** utasítást?
A program addig fut a **swith - case** szerkezetben, amig nem talál egy **break**-et, vagy a **swith - case** szerkezet véget nem ér.

Kérdések

Mi a szerepe a **switch** utasításnak?
Egy egész jellegű változó előre meghatározott értékeire tud ugrásokat adott **case** pontokra végrehajtani.
Mi lehet egy **case** után?
Csak egy érték.
Mi a szerepe a **default** utasításnak?
Ha egy **swith - case** szerkezet változója egy olyan értéket kap, amelyre nincs **case** ág, akkor a program a **default** ágra ugrik.
Mi történik, ha **swith - case** szerkezet változója egy olyan értéket kap, amelyre nincs **case** ág, de nincs **default** ág?
A program kilép a **swith - case** szerkezetből.
Mi a szerepe a **break** utasításnak **swith - case** szerkezetben?
Amikor a program az utasítást végrehajtja kilép a **swith - case** szerkezetből.
Mi történi, ha a program az adott **case** után nem talál **break** utasítást?
A program addig fut a **swith - case** szerkezetben, amig nem talál egy **break**-et, vagy a **swith - case** szerkezet véget nem ér.