# Výjimky

Strukturované ošetření chyb

# Ošetření chyb při běhu programu

- při běhu programu může dojít k chybě
- možnosti reakce na chybu
  - hlášení chyby a ukončení celého programu
    - nezapomenout na dealokace paměti, zavření souborů atd.
  - ukončení funkce a indikace chyby
    - výpis hlášení apod. - opatření ve volající funkci
  - **vyvolání výjimky**
    - strukturované ošetření chyb
    - rys objektových jazyků

# Ukončení funkce a indikace chyby

- funkce se ukončí a chybu signalizuje návratová hodnota
- příklad:

```
FILE *fopen(char *name, char*mod)
```

  - vrací NULL, pokud soubor nelze otevřít

```
FILE *fi;
fi = fopen("text.txt","rt");
if (fi==NULL)
{
}
```

# Příklad 2 - zásobník

```
class TZasob
{
   int max_vel;
   int pocet_prvku;
   float *pole;
 public:
   int vloz(float x);
   atd.
}
```

```
int TZasob::vloz(float x)
{
  if (pocet_prvku==max_vel)
  return 0;
  pole[pocet_prvku++] = x;
  return 1;
}
```

- funkce vrací hodnotu 1, pokud se podařilo úspěšně vložit prvek na zásobník, hodnotu 0 při pokusu vložit prvek do plného zásobníku

# Použití v hlavním programu

```
void main(void)
{
   TZasob zas(10);
   if (zas.vloz(5.4)==0)
   {
      cout << "Zasobnik je plny";
   }
}
```

- někdy může být s návratovou hodnotou problém
  - jak rozlišit chybový stav od správné hodnoty

```
float TZasob::vyber()
{
  if (pocet_prvku>0)
  {
    return pole[--pocet_prvku];
  }
  else return -1; ←——  ?
}
```

- jedno z řešení
  - nastavení globální chybové proměnné nebo chybové proměnné, která je výstupním parametr metody

```
float TZasob::vyber(int &chyba)
{
  if (pocet_prvku>0)
  {
    chyba = 0;
    return pole[--pocet_prvku];
  }
  else { chyba=1; return -1; }
}
```

- mohou být definovány chybové konstanty, hodnota 0 znamená bez chyby

```
#define OK 0
#define PLNY 1
#define PRAZDNY 2
float TZasob::vyber(int &chyba)
{
  if (pocet_prvku>0)
  {
    chyba = OK;
    return pole[--pocet_prvku];
  }
  else { chyba=PRAZDNY; return -1; }
}
```

- chybová proměnná může být i výčtového typu

```
typedef enum
   { OK, PLNYZAS, PRAZDNYZAS } TChybyZas;

float TZasob::vyber(TChybyZas &chyba)
{
   if (pocet_prvku>0)
   {
      chyba = OK;
      return pole[--pocet_prvku];
   }
   else { chyba=PRAZDNYZAS; return -1; }
}
```

# Použití v hlavním programu

```
void main(void)
{
   TZasob zas(10);
   TChybaZas chyba;
   float x = zas.vyber(chyba);
   if (chyba==PRAZDNYZAS)
   {
      cout << "Zasobnik je prazdny";
   }
}
```

- objekt může mít kód chyby uložen např. v soukromém atributu a vracet jeho hodnotu veřejnou metodou

```
typedef enum
  { OK, PLNYZAS, PRAZDNYZAS } TChybyZas;
class TZasob
{
  int max_vel;
  int pocet_prvku;
  float *pole;
  TChybyZas chyba;
 public:
  void vloz(float x);
  float vyber();
  TChybyZas vrat_chybu();
  atd.
}
```

```
float TZasob::vyber()
{
  if (pocet_prvku>0)
  {
    chyba = OK;
    return pole[--pocet_prvku];
  }
  else { chyba=PRAZDNYZAS; return -1; }
}

TChybaZas TZasob::vrat_chybu()
{
  return chyba;
}
```

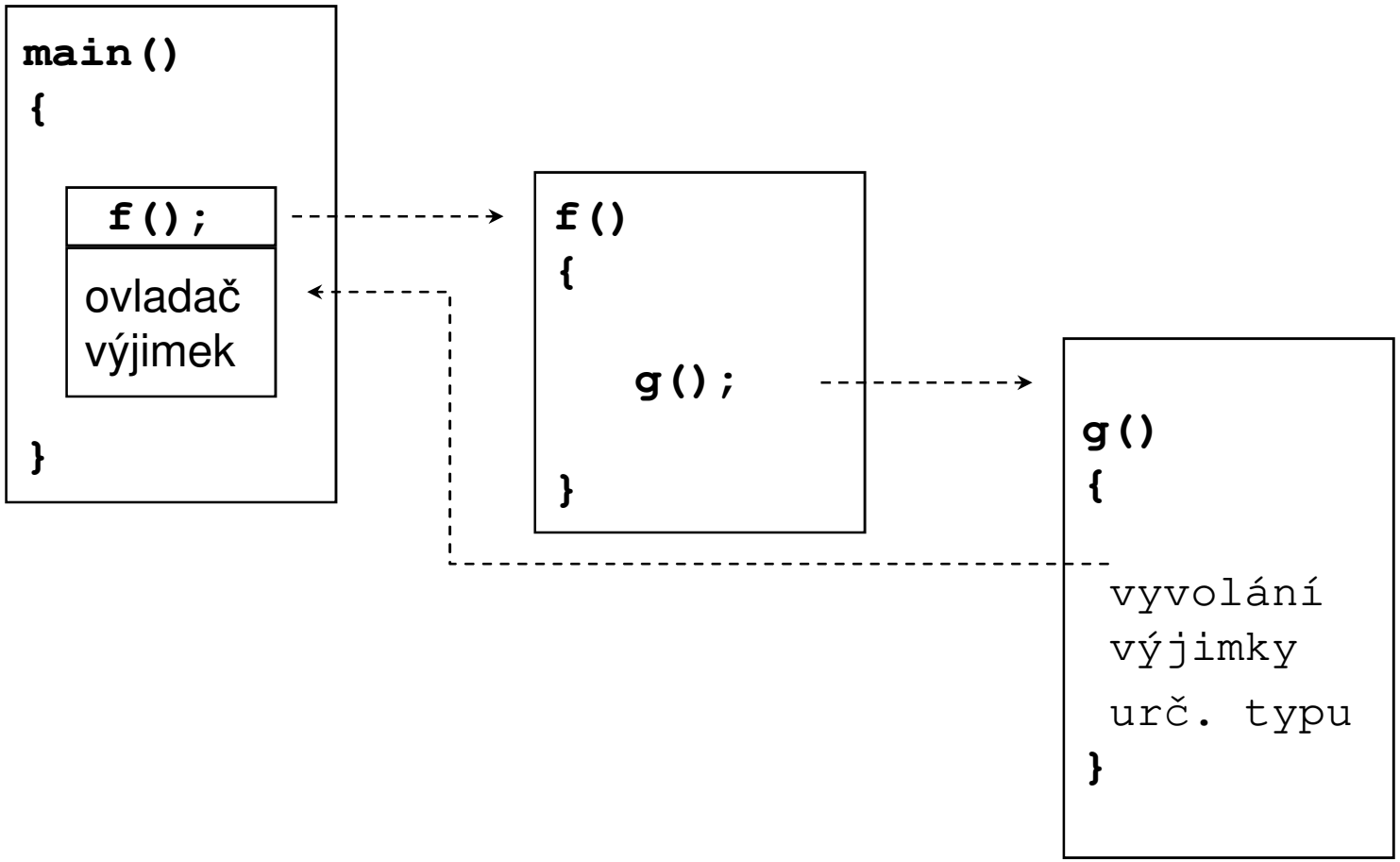
# Použití v hlavním programu

```
void main(void)
{
   TZasob zas(10);
   TChybaZas chyba;
   float x = zas.vyber();
   if (zas.vrat_chybu()==PRAZDNYZAS)
   {
      cout << "Zasobnik je prazdny";
   }
}
```

# Výjimky

- výjimka = exception
- jde o strukturované ošetření chyb
  - existují v C++, Javě, ...
- v případě výskytu chyby se programátor „vyhodí výjimku“ (vyvolá) speciálním příkazem **`throw`**
- vyvolaná výjimka se zachytí a obslouží v části kódu, který se nazývá ovladač výjimek (*exception handlers*)

# Mechanismus



# Vyvolání výjimky

- výjimka se vyvolá (vyhodí) příkazem

  **`throw výraz`**

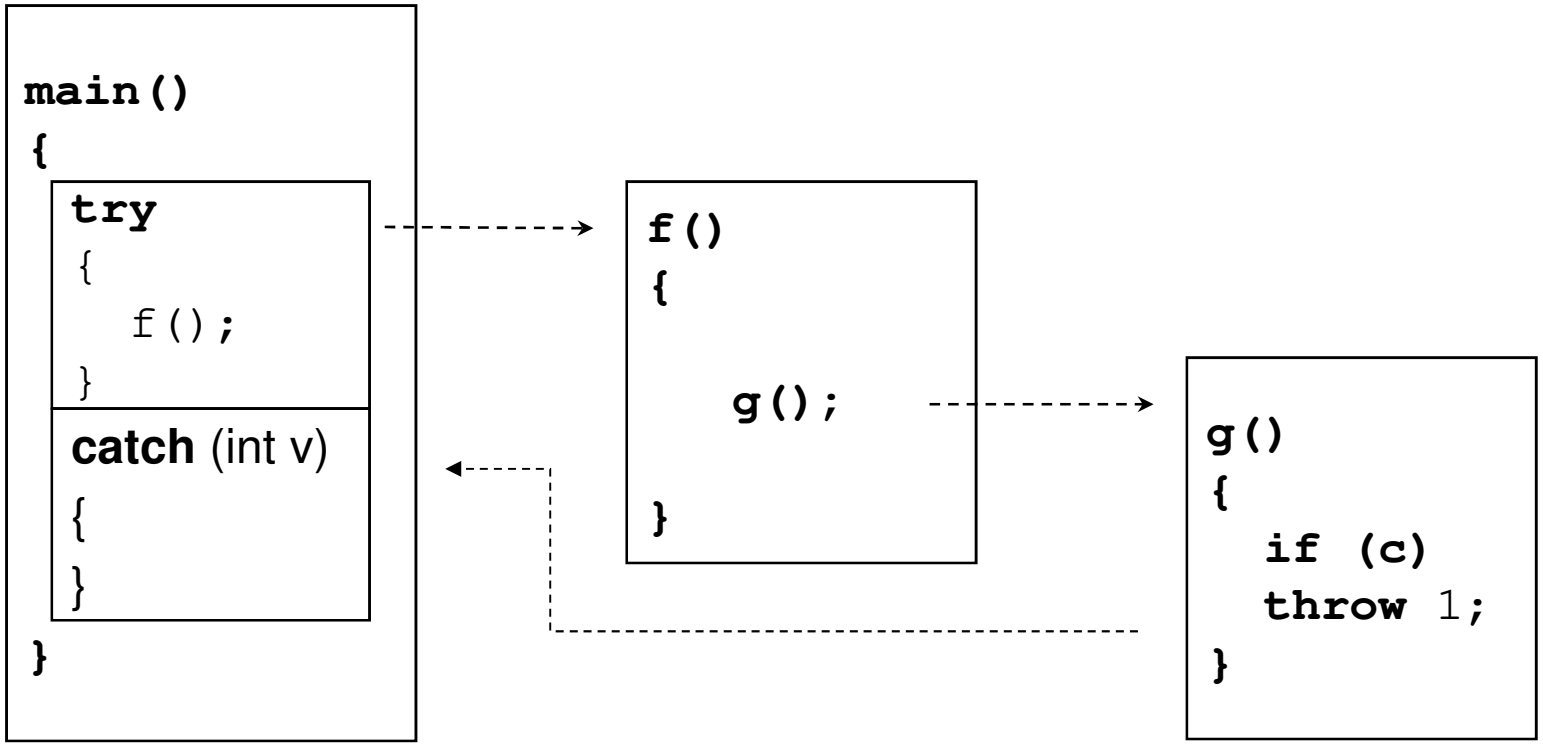
  - tj. vyhodí se k nejbližšímu ovladači výjimek
- výraz rozlišuje typy výjimek
- hodnota výrazu je parametrem výjimky

# Mechanismus použití

- blok, kde se může vyskytnout výjimka se uvozuje příkazem **`try`**
- ovladač výjimky se uvozuje příkazem **`catch`**
  - parametrem **`catch`** je typ výjimky, kterou ovladač obsluhuje
- nezachytí-li se výjimka v nejbližším ovladači, šíří se do dalšího nadřazeného ovladače

```
try
{
    …
}
catch (typ [ident]) { … }
…
catch (typ [ident]) { … }
catch (...) { … }
```

# Mechanismus podrobněji



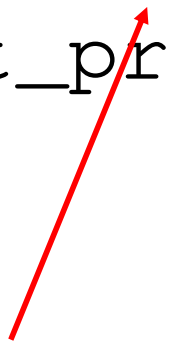
# Názornější příklad …

```
void TZasob::vloz(float x)
{
   if (pocet_prvku==max_vel)throw 1;
   pole[pocet_prvku++] = x;
}
```

při chybě funkce vyhazuje
výjimku typu int s hodnotou 1

# Názornější příklad …

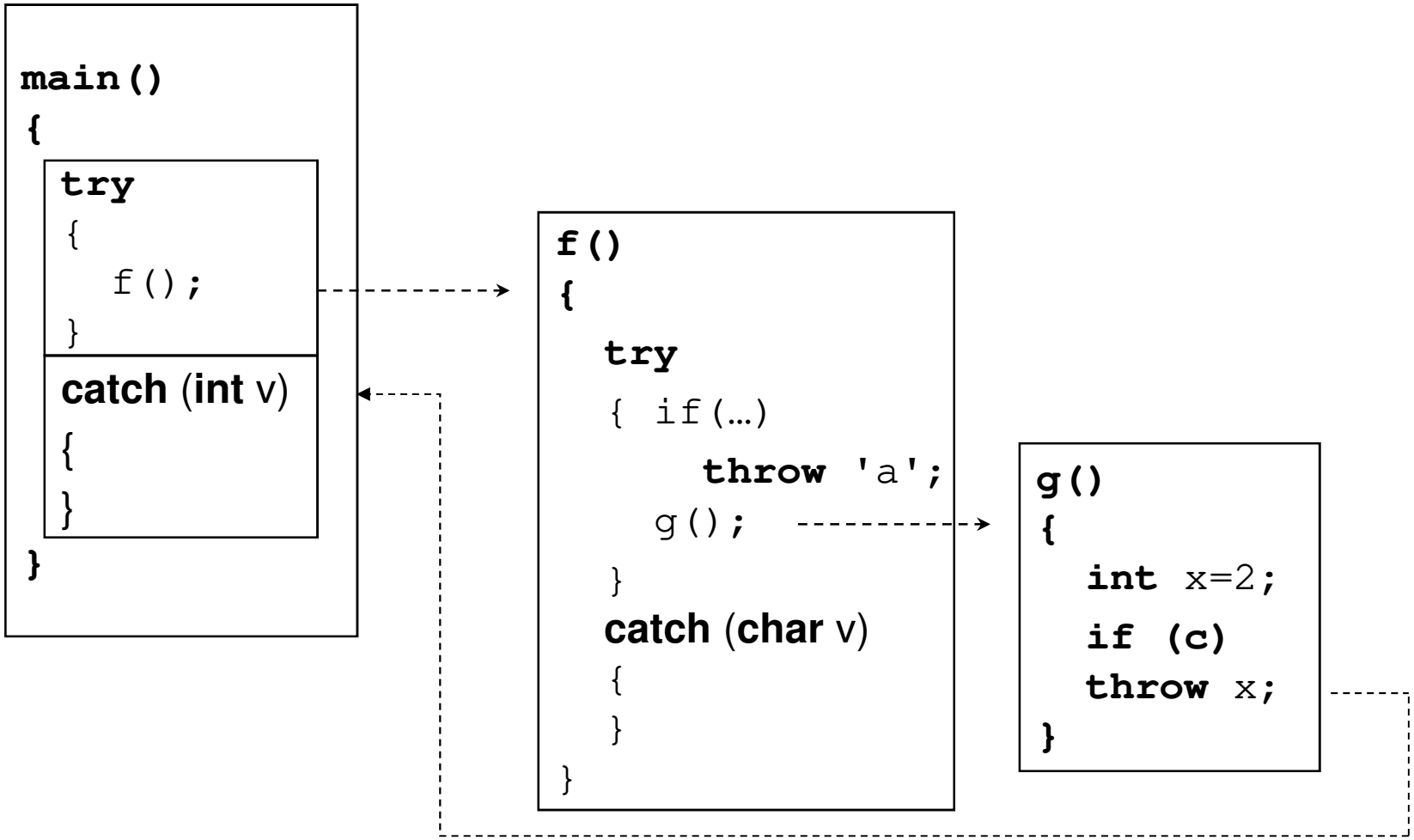
```
float TZasob::vyber()
{
   if (pocet_prvku<=0) throw 2;
   return pole[--pocet_prvku];
}
```



při chybě funkce vyhazuje
výjimku typu int s hodnotou 2

```
void main (void)
{ TZasob zas(10);
  try
  {
    x = zas.vloz(5);
    …
    x = zas.vyber();
  }
  catch (int v)
  {
    if (v==1) cout << "Je plny";
    if (v==2) cout << "Je prazdny";
  }
}
```

# Další příklad



- výjimkou může být často objekt
  - je to výhodné pro rozlišení typu výjimky

```
class TChybaZas
{
  private:
   int kod_chyby;
  public:
   TChybaZas(int kod) { kod_chyby = kod; }
   int vrat_kod() { return kod_chyby; }
}
```

```
#define PLNY 1
#define PRAZDNY 2
float TZasob::vyber()
{
if (pocet_prvku<=0)
  throw TChybaZas(PRAZDNY);
else return pole[--pocet_prvku];
}
```

```
void main (void)
{ TZasob zas(10);
  try
  {
    …
    x = zas.vyber();
  }
  catch (TChybaZas &chyba)
  {
    if (chyba.vrat_kod()==PRAZDNY)
     cout << "Je prazdny";
  }
}
```

je lepší použít referenci

# Příklad – Borland C++

```
{
  TOpenDialog *od = new TOpenDialog(this);
  od -> Filter = "Obrázky (*.bmp)|*.bmp|Všechny soubory
   (*.*)|*.*";
  od ->Execute();
  try
  {
    if (od -> Files -> Count > 0)
    {
      Image1 -> Picture -> LoadFromFile(od -> FileName);
      nahran_obr = true;
      …
    }
  }
  catch (Exception &EFOpenError)
  {
    Application -> MessageBox("Soubor nelze
   otevřít.","Chyba",MB_OK);
  }
  delete od;
}
```

- příklady příkazů **`throw`** a ovladačů výjimek

| příkaz **`throw`** | odpovídající ovladač |
|---|---|
| throw 5; | catch (int n) { … } |
| throw ‘a’; | catch (char c) { … } |
| throw “Help!”; | catch (const char *s) { … } |
| throw TChybaZas(1); | catch (TChybaZas& s) { … } |

# Operátor `new`

- existují dvě varianty operátoru lišící se chováním při nedostatku paměti
  1. vrací hodnotu NULL jako funkce `malloc`

```
void* operator new (std::size_t size,
                    const std::nothrow_t& nothrow_value) throw();
```

  2. vyhazuje výjimku `bad_alloc`

```
void* operator new (std::size_t size) throw (std::bad_alloc);
```

# Operátor `new`

- výjimka:

```
try
{
   MyClass * p1 = new MyClass;
   …
}
catch (std::bad_alloc &a)
{
   cerr >> "Nedostatek pameti";
}
```

# Operátor `new`

- návratová hodnota NULL:

```
MyClass * p2 = new (std::nothrow) MyClass;
if (p2 == NULL)
{
  cerr >> "Nedostatek pameti";
  return -1;
}
```

- je možné střídat bloky **`try{}`** a **`catch{}`**

```
void main (void)
{
   try
   { …
   }
   catch (int v)
   { …
   }
   try
   { …
   }
   catch (int v)
   { …
   }
}
```

- je možné vnořovat bloky **`try{}`** a **`catch{}`**

```
void main (void)
{
   try
   {
      try
      {
      }
      catch (…)
      {
      }
   }
   catch (…)
   { …
   }
}
```

- při deklaraci prototypu funkce (metody) je možné specifikovat dle standardu C++98, jakou výjimku může (přímo nebo nepřímo) vyhodit

```
int moje(int p) throw(int);
```
  - funkce může vyhazovat výjimku typu **int**

```
int moje(int p) throw();
```
  - funkce nesmí vyhazovat žádnou výjimku

```
int moje(int p);
```
  - funkce může vyhazovat libovolnou výjimku

- dle standardu C++11 se používá nové klíčové slovo `noexept`

```
int moje(int p) noexcept;
```

  – funkce nesmí vyhazovat žádnou výjimku

```
int moje(int p);
```

  – funkce může vyhazovat libovolnou výjimku

# Poznámka

- v jazyce Java existuje stejný mechanismus výjimek
- je přidáno klíčové slovo **`finally`**
  - do tohoto bloku se píše kód, který se má provést v obou případech, tj. když byla i nebyla vyhozena výjimka
  - některé překladače C++ dovolují použít i `__finally` (s podtržítkem)

```
void main (void)
{
   try
   { …
   }
   catch (int v)
   { …
   }
   _finally
   { …
   }
}
```

# Úkol

Doplňte ke třídě implementující zásobník definici třídy představující výjimku, např. TChybaZasobniku, která nese atribut informující o typu chyby (výběr z prázdného zásobníku, snaha o vložení prvku do zaplněného zásobníku). Doplňte do hlavního programu obsluhu výjimek.