

Základy procedurálneho programovania 2



FIIT STU, Mlynská dolina
Aula Minor, pondelok 9:00

letný semester
2016/2017

Ideme podľa plánu

dátum	prednáška	8:00	9:00	cvičenie	obsah
20.3.	6		Čítanie kódu, Hľadanie chýb v kóde	6	Projekt 1: odovzdanie
27.3.	7	Test 3	Riešenie testu 3, Spájané zoznamy	7	Projekt 2 Tezeus a Minotaurus
3.4.	8		Tezeus, Bitové operácie	8	
10.4.	9	Test 4	Riešenie testu 4, Rekurzia, Minotaurus	9	
17.4.	Veľká noc			X	
24.4.	10		Ďalšie prvky jazyka C	10	Projekt 2: odovzdanie, konzultovanie
1.5.	Sviatok			11	
8.5.	Sviatok			12	
9.5.	11		Opakovanie	X	
15.5.	12	Predtermín?		X	

Opakovanie – Hlavička funkcie

- V nasledujúcich príkladoch napíš hlavičku funkcie a vysvetli ako bude s údajmi pracovať.
 - a) Funkcia **vloz**, ktorá do (pôvodného) reťazca vloží iný (nový) reťazec od pozície k, predpokladajte že pôvodný reťazec nie je dostatočne veľký pre nové znaky. Nepoužívajte globálne premenné.
- Vstupy: `char *povodny, const char *vkladany, int k`
- Vystupy: `char *novySpojeny`
- **`char* vloz(char *povodny, char *vkladany, int k)`**

Opakovanie – Hlavička funkcie (2)

- V nasledujúcich príkladoch napíš hlavičku funkcie a vysvetli ako bude s údajmi pracovať.

b) Funkcia **prvocisla**, ktorá pre uzavretý interval (od, do) prirodzených čísel zistí, ktoré z čísel sú prvočísla. Výsledok vráti ako jednorozmerné pole čísel (int). Nepoužívajte globálne premenné.

- Vstup: int cislo od=1, int cislo do=100M,
- Výsledok naplnené pole čísel: int *prvy, int pocet
- **int *prvocislo(int od, int _do, int *pocet);**

Opakovanie – Smerník

- Smerník – premenná, ktorá obsahuje adresu
- Smerník môže mať dátový typ, napr. smerník na int (adresa pamäte v ktorej je vyhradené miesto pre int)
- Dátový typ „smerník na typ“ píšeme s hviezdičkou typ*, napr. **smerník na int** píšeme **int***
- Príklad:
int i = 30; i je meno pre pamäť, kde je int
int* p = &i; p je adresa premennej i
- Pre priradenie využitím smerníka použijeme operátor hviezdička (tzv. dereferencovanie smerníku):
***p = 20;** je to isté ako **i = 20;**

Opakovanie – Vymeň hodnotu (swap)

- Úloha: **Napíš program, ktorý vymení hodnotu dvoch premenných.**

- **Jednorázový príkaz „niekde“ v programe:**

```
int tmp, u = 10, v = 20;  
tmp = u; u = v; v = tmp;
```

- **Znovupoužiteľná funkcia:**

```
void swap(int *x, int *y)  
{  
    int tmp = *x;  
    *x = *y;  
    *y = tmp;  
}
```

```
// použitie  
int u = 10, v = 20;  
swap(&u, &v);
```

Základy procedurálneho programovania 2

Čítanie kódu, Hľadanie chýb v kóde

20.3.2017

letný semester
2016/2017

Hľadáme chybu...

```
1 #include <stdio.h>
2
3 int main()
4 {
5     int x = 10, y = 10;
6     printf("x=%d, y=%d\n", x, y);
7     swap(&x, &y);
8     printf("x=%d, y=%d\n", x, y);
9     return 0;
10 }
```

- 7:3: warning: implicit declaration of function 'swap' [-Wimplicit-function-declaration]
- Hľadá nejakú implicitnú funkciu swap... Kompilátor predpokladá, že asi niekde existuje nejaká funkcia swap, a len som zabudol na ňu uviesť odkaz...

Hľadáme chybu...

```
1 #include <stdio.h>
2
3 int main()
4 {
5     int x = 10, y = 10;
6     printf("x=%d, y=%d\n", x, y);
7     swap(&x, &y);
8     printf("x=%d, y=%d\n", x, y);
9     return 0;
10 }
```

- 7: undefined reference to `swap'
- Keďže linker nenašiel implementáciu swap, tak o tom podal správu...
- Chýba nám implementácia funkcie swap...

Hľadáme chybu...

```
1 #include <stdio.h>
2
3 int swap(int *i, int *j)
4 {
5     int tmp = *i;
6     *i = *j;
7     *j = tmp;
8 }
9
10 int main()
11 {
12     int x = 10, y = 10;
13     printf("x=%d, y=%d\n", x, y);
14     swap(&x, &y);
15     printf("x=%d, y=%d\n", x, y);
16     return 0;
17 }
```

- 8:1: warning: control reaches end of non-void function [-Wreturn-type]
- Funkcia swap má v hlavičke uvedenú návratovú hodnotu (int) ... Omylom?
- Kompilátor nevidí explicitné volanie return, tak vrátil upozornenie...

Hľadáme chybu...

```
1 #include <stdio.h>
2
3 void swap(int *i, int *j)
4 {
5     int tmp = i;
6     i = j;
7     j = tmp;
8 }
9
10 int main()
11 {
12     int x = 10, y = 10;
13     printf("x=%d, y=%d\n", x, y);
14     swap(&x, &y);
15     printf("x=%d, y=%d\n", x, y);
16     return 0;
17 }
```

- 5:13: warning: initialization makes integer from pointer without a cast [enabled by default]
- Priradenie (riadok 5) priradzuje medzi nekompatibilnými typmi premenných... omyl? alebo úmysel?

Hľadáme chybu...

```
1 #include <stdio.h>
2
3 void swap(int *i, int *j)
4 {
5     int tmp = i;
6     i = j;
7     j = tmp;
8 }
9
10 int main()
11 {
12     int x = 10, y = 10;
13     printf("x=%d, y=%d\n", x, y);
14     swap(&x, &y);
15     printf("x=%d, y=%d\n", x, y);
16     return 0;
17 }
```

- 7:5: warning:
assignment makes
pointer from integer
without a cast
[enabled by default]
- Priradenie (riadok 7)
priraduje medzi
nekompatibilnými
typmi premenných...
omyl? alebo úmysel?

Hľadáme chybu...

```
1 #include <stdio.h>
2
3 void swap(int *i, int *j)
4 {
5     int tmp = *i;
6     *i = *j;
7     *j = *tmp;
8 }
9
10 int main()
11 {
12     int x = 10, y = 10;
13     printf("x=%d, y=%d\n", x, y);
14     swap(&x, &y);
15     printf("x=%d, y=%d\n", x, y);
16     return 0;
17 }
```

- 7:8: error: invalid type argument of unary '*' (have 'int')
- Unárny operátor * dostal argument nesprávneho typu ...
- Tmp nie je smerník a preto ho nemožno dereferencovať ...

Hľadáme chybu...

```
1 #include <stdio.h>
2
3 void swap(int *i, int *j)
4 {
5     int tmp = *i;
6     *i = *j;
7     *j = tmp;
8 }
9
10 int main()
11 {
12     int x = 10, y = 10;
13     printf("x=%d, y=%s\n", x, y);
14     swap(&x, &y);
15     printf("x=%d, y=%d\n", x, y);
16     return 0;
17 }
```

- | 3:3: warning: format '%s' expects argument of type 'char *', but argument 3 has type 'int' [-Wformat]
- Formátovací reťazec obsahuje položku %s, ale v zozname ďalších parametrov nie je druhý typu reťazec (char*)... Kompilátor tam očakáva reťazec (je tam int)

Hľadáme chybu...

```
1 #include <stdio.h>
2
3 void swap(int *i, int *j)
4 {
5     int tmp = *i;
6     *i = *j;
7     *j = tmp;
8 }
9
10 int main()
11 {
12     int x = 10, y = 10;
13     printf("x=%d, y=%d\n", x);
14     swap(&x, &y);
15     printf("x=%d, y=%d\n", x, y);
16     return 0;
17 }
```

- 13:3: warning: format '%d' expects a matching 'int' argument [-Wformat]
- Formátovací reťazec obsahuje dve položky %d, ale v zozname parametrov nie sú dve položky ... Kompilátor tam očakával int (nie je tam nič)

Hľadáme chybu...

```
1 #include <stdio.h>
2
3 void swap(int *i, int *j)
4 {
5     int tmp = *i;
6     *i = *j;
7     *j = tmp;
8 }
9
10 int main()
11 {
12     int x = 10, y = 10;
13     printf("x=%d, y=d\n", x, y);
14     swap(&x, &y);
15     printf("x=%d, y=%d\n", x, y);
16     return 0;
17 }
```

- |3:3: warning: too many arguments for format [-Wformat-extra-args]
- Formátovací reťazec obsahuje len jednu položku %d, ale v zozname parametrov sú dve položky ... Kompilátor sa sťažuje, že tam je príliš veľa argumentov.

Hľadáme chybu...

```
1 #include <stdio.h>
2
3 void swap(int *i, int *j)
4 {
5     int tmp = *i;
6     *i = *j;
7     *j = tmp;
8 }
9
10 int main()
11 {
12     int x = 10, y = 10;
13     printf("x=%d, y=%d\n", x, y);
14     swap(&x);
15     printf("x=%d, y=%d\n", x, y);
16     return 0;
17 }
```

- |4:3: error: too few arguments to function 'swap'
- Pri volaní funkcie som uviedol málo argumentov ...
Kompilátor vie, že pre funkciu swap by mali byť dva argumenty, ale uviedol som ich príliš málo (too few)...

Hľadáme chybu...

```
1 #include <stdio.h>
2
3 void swap(int *i, int *j)
4 {
5     int tmp = *i;
6     *i = *j;
7     *j = tmp;
8 }
9
10 int main()
11 {
12     int x = 10;
13     printf("x=%d, y=%d\n", x, y);
14     swap(&x, &y);
15     printf("x=%d, y=%d\n", x, y);
16     return 0;
17 }
```

- 13:29: error: 'y' undeclared (first use in this function)
- Používam identifikátor y a kompilátor nevie čo to je? Je to funkcia, je to premenná? Akého typu?

Hľadáme chybu...

```
1 #include <stdio.h>
2
3 void swap(int *i, int *j)
4 {
5     int tmp = *i;
6     *i = *j;
7     *j = tmp;
8 }
9
10 int main()
11 {
12     int x = 10, y = 10;
13     printf("x=%d, y=%d\n", x, y);
14     swap(&x, &y);
15     printf("x=%d, y=%d\n", x, y);
16     return;
17 }
```

- | 6:3: warning: 'return' with no value, in function returning non-void [-Wreturn-type]
- Funkcia by mala vracať celé číslo (int) ale nevracia nič ...
- Prečo je to len warning? Pretože podľa štandardu sa vráti hodnota poslednej operácie.

Hľadáme chybu...

```
1 #include <stdio.h>
2
3 void swap(int *i, int *j)
4 {
5     int tmp = *i;
6     *i = *j;
7     *j = tmp;
8 }
9
10 int main()
11 {
12     int x = 10, y = 10;
13     printf("x=%d, y=%d\n", x, y);
14     swap(&x, &y);
15     printf("x=%d, y=%d\n", x, y);
16     return 0;
17 }
```

- 14:3: error: expected ';' before 'swap'
- Kompilátor očakáva bodkočiarku pred volaním swap...
- Čo to má z volaním swap? Nič, to iba predchádzajúci príkaz nie je správne ukončený...

Hľadáme chybu...

```
1 #include <stdio.h>
2
3 void swap(int *i, int *j)
4 {
5     int tmp = *i;
6     *i = *j;
7     *j = tmp;
8
9
10 int main()
11 {
12     int x = 10, y = 10;
13     printf("x=%d, y=%d\n", x, y);
14     swap(&x, &y);
15     printf("x=%d, y=%d\n", x, y);
16     return 0;
17 }
```

- 10:5: warning: 'main' is normally a non-static function [-Wmain]
- Kompilátor očakáva štandardný main, a tuto je „nejaký“ statický? huh?

Hľadáme chybu...

```
1 #include <stdio.h>
2
3 void swap(int *i, int *j)
4 {
5     int tmp = *i;
6     *i = *j;
7     *j = tmp;
8
9
10 int main()
11 {
12     int x = 10, y = 10;
13     printf("x=%d, y=%d\n", x, y);
14     swap(&x, &y);
15     printf("x=%d, y=%d\n", x, y);
16     return 0;
17 }
```

- Ďalšia chyba to vyjasní
- 17:1: error: expected declaration or statement at end of input
- Kompilátor očakáva že nájde deklaráciu alebo vetu na konci vstupu
- Dôležité je to „na konci vstupu“, čo znamená že koniec vstupu nie je korektný ... Zvyčajne sa vyskytuje vtedy, keď sú zle uvedené zložené zátvorky, čo môže byť hocikde v program ...

Hľadáme chybu...

```
1 #include <stdio.h>
2
3 void swap(int *i, int *j)
4 {
5     int tmp = *i;
6     *i = *j;
7     *j = tmp;
8
9
10 int main()
11 {
12     int x = 10, y = 10;
13     printf("x=%d, y=%d\n", x, y);
14     swap(&x, &y);
15     printf("x=%d, y=%d\n", x, y);
16     return 0;
17 }
```

- Ďalšia chyba to vyjasní
- 17:1: error: expected declaration or statement at end of input
- Zvyčajne sa vyskytuje vtedy, keď sú zle uvedené zložené zátvorky, čo môže byť hocikde v program ...
- Miesto, kde by to asi mohlo byť naznačuje umiestnenie ostatných chýb a upozornení

Hľadáme chybu...

```
1 #include <stdio.h>
2
3 void swap(int *i, int *j)
4 {
5     int tmp = *i;
6     *i = *j;
7     *j = tmp;
8 }
9
10 int main()
11 {
12     int x = 10, y = 10;
13     printf("x=%d, y=%d\n", x, y);
14     swap(&x, &y);
15     printf("x=%d, y=%d\n", x, y);
16     return 0;
17 }
```

- 13:10: warning: missing terminating " character
- 13:3: error: missing terminating " character
- 14:15: error: expected ')' before ';' token
- ...
- Zle ukončený reťazec
- Vznikne veľa ďalších chýb a upozornení...

Hľadáme chybu...

```
1 #include <stdio.h>
2
3 void swap(int *i, int *j)
4 {
5     int tmp = *i;
6     *i = *j;
7     *j = tmp;
8 }
9
10 int main()
11 {
12     int x = 10, y = 10;
13     printf('x=%d, y=%d\n', x, y);
14     swap(&x, &y);
15     printf("x=%d, y=%d\n", x, y);
16     return 0;
17 }
```

- 13:10: warning: character constant too long for its type [enabled by default]
- ...
- Znaková konštanta je príliš dlhá (v jednoduchých apostrofoch môže byť uvedený len jeden znak)

Hľadáme chybu...

```
1 #include <stdio.h>
2
3 void swap(int *i, int *j)
4 {
5     int tmp = *i;
6     *i = *j;
7     *j = tmp;
8 }
9
10 int main()
11 {
12     int x = 10, y = 10;
13     printf("x=%d, y=%d\n", x.cislo);
14     swap(&x, &y);
15     printf("x=%d, y=%d\n", x, y);
16     return 0;
17 }
```

- 13:27: error: request for member 'cislo' in something not a structure or union
- V premennej pristupujeme k zložke cislo, ale táto premenná nie je štruktúra ...

Hľadáme chybu...

```
1 #include <stdio.h>
2
3 void swap(int *i, int *j)
4 {
5     int tmp = *i;
6     *i = *j;
7     *j = tmp;
8 }
9
10 int main()
11 {
12     int x = 10, y = 10;
13     printf("x=%d, y=%d\n", x->cislo, y);
14     swap(&x, &y);
15     printf("x=%d, y=%d\n", x, y);
16     return 0;
17 }
```

- 13:27: error: invalid type argument of '->' (have 'int')
- Operátor šípka používame na premennej nesprávneho typu (int), očakáva smerník...

Hľadáme chybu...

```
1 #include <stdio.h>
2
3 void swap(int *i, int *j)
4 {
5     int tmp = *i;
6     *i = *j;
7     *j = tmp;
8 }
9
10 int main()
11 {
12     int x = 10, y = 10;
13     swap(&x, &y);
14     if (x = y)
15         printf("x=%d, y=%d\n", x, y);
16     return 0;
17 }
```

- |4:3: warning: suggest parentheses around assignment used as truth value [-Wparentheses]
- Kompilátor odporúča použiť zátvorky okolo priradenia, ktoré sa používa ako pravdivostná hodnota
- Zvyčajе je to vtedy, keď pri podmienke omylom uvediem = namiesto ==

Hľadáme chybu...

```
1 #include <stdio.h>
2
3 void swap(int *i, int *j)
4 {
5     int *tmp = i;
6     i = j;
7     j = tmp;
8 }
9
10 int main()
11 {
12     int x = 10, y = 20;
13     printf("x=%d, y=%d\n", x, y);
14     swap(&x, &y);
15     printf("x=%d, y=%d\n", x, y);
16     return 0;
17 }
```

- Ako nájsť chybu ktorú nenájde kompilátor...
- Spôsobom „blackbox“ program považujem za tajomnú čiernu skrinku (neviem ako vnútri pracuje, len očakávam správny výsledok)
- Skúšam rôzne vstupy a zisťujem či sú správne

Hľadanie chýb, ktoré nenájde kompilátor...

- Spôsobom „whitebox“: program považujem za otvorený kód, ktorý si môžem prečítať a analyzovať
- Dve základné stratégie čítania kódu (pre porozumenie):
 - Top-down (zhora nadol)
 - Bottom-up (zdola nahor)

```
1 #include <stdio.h>
2
3 void swap(int *i, int *j)
4 {
5     int *tmp = i;
6     i = j;
7     j = tmp;
8 }
9
10 int main()
11 {
12     int x = 10, y = 20;
13     printf("x=%d, y=%d\n", x, y);
14     swap(&x, &y);
15     printf("x=%d, y=%d\n", x, y);
16     return 0;
17 }
```

Top-down vs bottom-up

- https://en.wikipedia.org/wiki/Top-down_and_bottom-up_design
- Software Comprehension – A Review & Research Direction
<https://www.st.cs.uni-saarland.de/edu/empirical-se/2006/PDFs/brien03.pdf>
- <http://www.cse.dmu.ac.uk/~mward/msc-se-2015/01-program-comprehension.pdf>
- Kniha: <http://www.literateprogramming.com/em3.pdf>

Čítame program... (s dobrými názvami to ide ľahšie)

```
1 #include <stdio.h>
2 int x(int y)
3 {
4     int i;
5     for (i = 2; i*i <= y; i++)
6         if (y % i == 0)
7             return -1;
8     return y > 1;
9 }
10 int main()
11 {
12     int a, b, i;
13     scanf("%d %d", &a, &b);
14     for (i = a; i <= b; i++)
15         if (x(i) > 0)
16             printf("%d\n", i);
17     return 0;
18 }
```

```
1 #include <stdio.h>
2 int zisti_prvocislo(int cislo)
3 {
4     int i;
5     for (i = 2; i*i <= cislo; i++)
6         if (cislo % i == 0)
7             return -1;
8     return cislo > 1;
9 }
10 int main()
11 {
12     int a, b, i;
13     scanf("%d %d", &a, &b);
14     for (i = a; i <= b; i++)
15         if (zisti_prvocislo(i) > 0)
16             printf("%d\n", i);
17     return 0;
18 }
```


Čítame program... (s dobrými názvami to ide ľahšie)

```
3 #include <stdio.h>
4
5 int x(int w, int h)
6 {
7     if (h > 0)
8         return x(h, w%h);
9     return w;
10 }
```

```
3 #include <stdio.h>
4
5 int gcd(int a, int b)
6 {
7     if (b == 0)
8         return a;
9     return gcd(b, a%b);
10 }
```

Čítame program...

```
1 #include <stdio.h>
2 #include <string.h>
3
4 int fn(char *d, int m, char *s, int x)
5 {
6     // sem napis svoje riesenie
7     int l = strlen(d), k=strlen(s);
8     if (x > l || l + k >= m)
9         return 1;
10
11     int i;
12     for (i = l+k; i >= x+k; i--)
13         d[i] = d[i-k];
14     for (i = 0; i < k; i++)
15         d[x+i] = s[i];
16     return 0;
17 }
```

Čítame program... (s dobrými názvami to ide ľahšie)

```
1 #include <stdio.h>
2 #include <string.h>
3
4 int strinsert(char *dst, int len, const char *src, int offset)
5 {
6     // sem napis svoje riesenie
7     int dl = strlen(dst), sl=strlen(src);
8     if (offset > dl || dl + sl >= len)
9         return 1;
10
11     int i;
12     for (i = dl+sl; i >= offset+sl; i--)
13         dst[i] = dst[i-sl];
14     for (i = 0; i < sl; i++)
15         dst[offset+i] = src[i];
16     return 0;
17 }
```

Bez farebného zvýrazňovania...

```
#include <stdio.h>

#define MIN(a,b) (((a)<(b))?(a):(b))

int n, rgb[50][3], dp[20][3];

int main()
{
    int i;
    while (scanf("%d", &n) > 0)
    {
        for(i = 0; i < n; i++)
        {
            scanf("%d %d %d", &rgb[i][0], &rgb[i][1], &rgb[i][2]);
            dp[i][0] = dp[i][1] = dp[i][2] = 0;
        }
        dp[0][0] = rgb[0][0];
        dp[0][1] = rgb[0][1];
        dp[0][2] = rgb[0][2];

        for (i = 1; i < n; i++)
        {
            dp[i][0] = MIN( dp[i-1][1] , dp[i-1][2] ) + rgb[i][0] ;
            dp[i][1] = MIN( dp[i-1][0] , dp[i-1][2] ) + rgb[i][1] ;
            dp[i][2] = MIN( dp[i-1][0] , dp[i-1][1] ) + rgb[i][2] ;
        }
        printf("%d\n", MIN(MIN(dp[n-1][0], dp[n-1][1]), dp[n-1][2]));
    }
    return 0;
}
```

Čo si môžeme všimnúť v útržkoch...

- Čo si tu všimneme?
- Aké sú hlavné prvky / vlastnosti tohto kódu?

```
#include <stdio.h>

int result[50][50];

int solve(int start, int end, int *a)
{
    if (start > end)
        return 0;

    if (result[start][end] < 0)
    {
        int r1 = solve(start+1, end, a);
        int r2 = solve(start+2, end, a) + a[start];
        if (r1 < r2)
            r1 = r2;
        result[start][end] = r1;
    }
    return result[start][end];
}
```

Po 15 rokoch ...

- Čo keď sa rozhodnem čítať svoj zdrojový kód **po 15 rokoch ...**
- Našiel som niečo čo som programoval v roku 2002
- Ako sa v tom budem vyznať?
- Čo od toho očakávate?
- Splnia sa vaše očakávania?
- Let us find out ...
- Ponaučenie do budúcnosti:
treba v programoch zrozumiteľne nazývať premenné a funkcie, a používať vhodne nazvané skratky

Základy procedurálneho programovania 2

Vývojové nástroje

20.3.2017

letný semester
2016/2017

Vývojové nástroje

- Windows
 - Microsoft Visual Studio 2012 – 2017
 - msdnaa.fiit.stuba.sk
- Linux
 - GCC, the GNU Compiler Collection
 - GDB: The GNU Project Debugger
- Mac OSX
 - XCode

MSDN Academic Alliance na FIIT STU Bratislava

Kde je dostupný softvér?

Sťahovanie softvéru a spravovanie inšalačných kľúčov zabezpečuje elektronický systém: **DreamSpark**.

Ako získať konto?

Študenti (bakalárske, inžinierske a doktorandské štúdium)

Študentom všetkých troch stupňov štúdia sú kontá pre prístup vytvorené automaticky a nemusia o ne žiadať. Kontá sú začiatkom každého semestra aktualizované, o čom bude zverejnený oznam na tejto stránke, spravidla v priebehu prvého týždňa semestra.

Zamestnanci

Zamestnanci pokiaľ konto nemajú, môžu oň požiadať e-mailom správcu na adrese *msdnaa [zavináč] fiit.stuba.sk*.

Ako získať prihlasovacie údaje?

Študenti bakalárskeho a inžinierskeho štúdia majú **prihlasovacie meno** rovnaké ako e-mailová adresa na serveri **student**, teda napr. **priezvisko02@student.fiit.stuba.sk**.

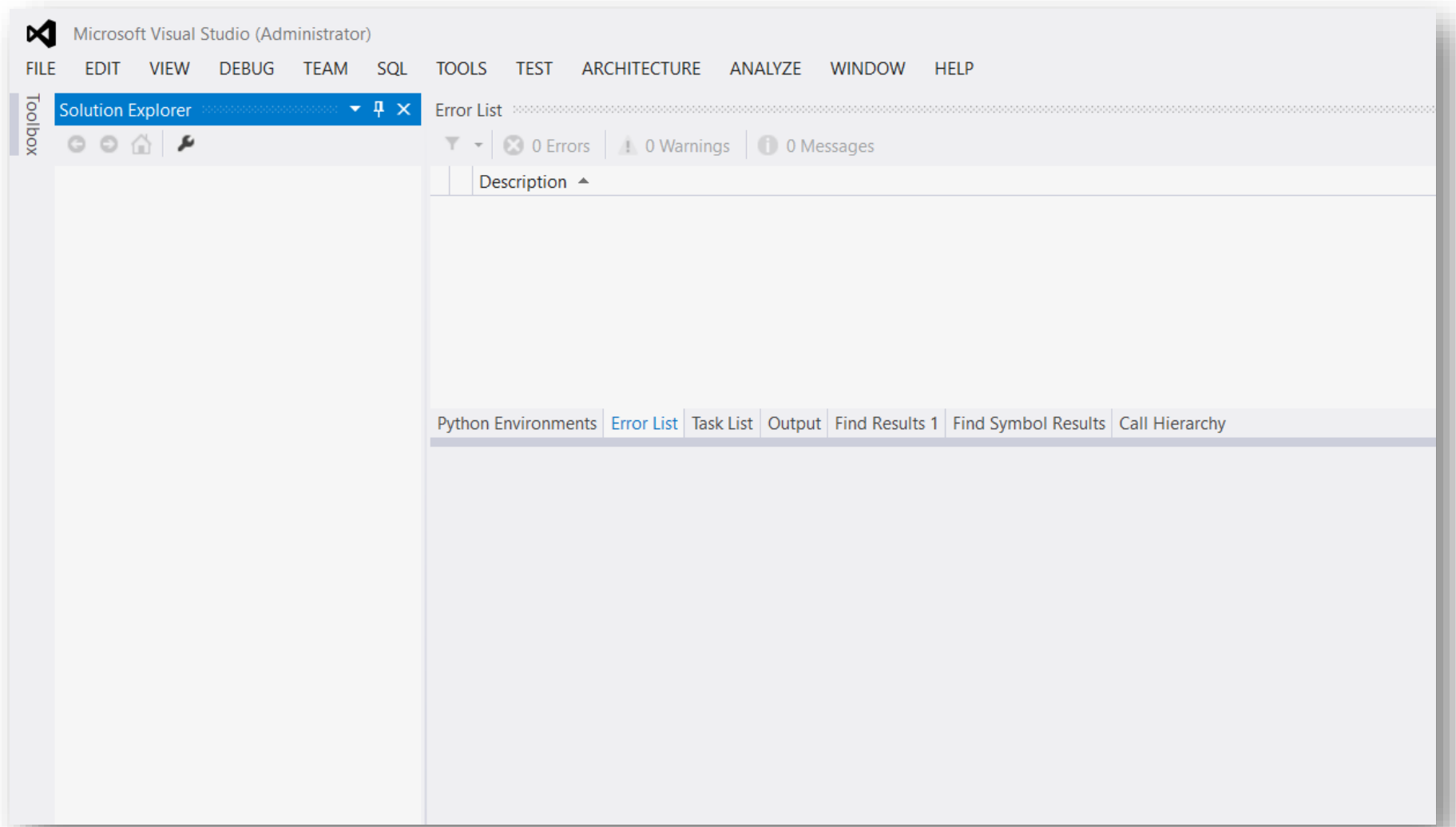
Študenti doktorandského štúdia ktorí konto doteraz nemali, majú prihlasovacie meno rovnaké ako univerzitná e-mailová adresa, teda napr. **xpriezvisko@stuba.sk**. Kontá vytvorené v minulosti zostávajú naďalej platné.

Pri prvom prihlásení je potrebné dokončiť tzv. registráciu.

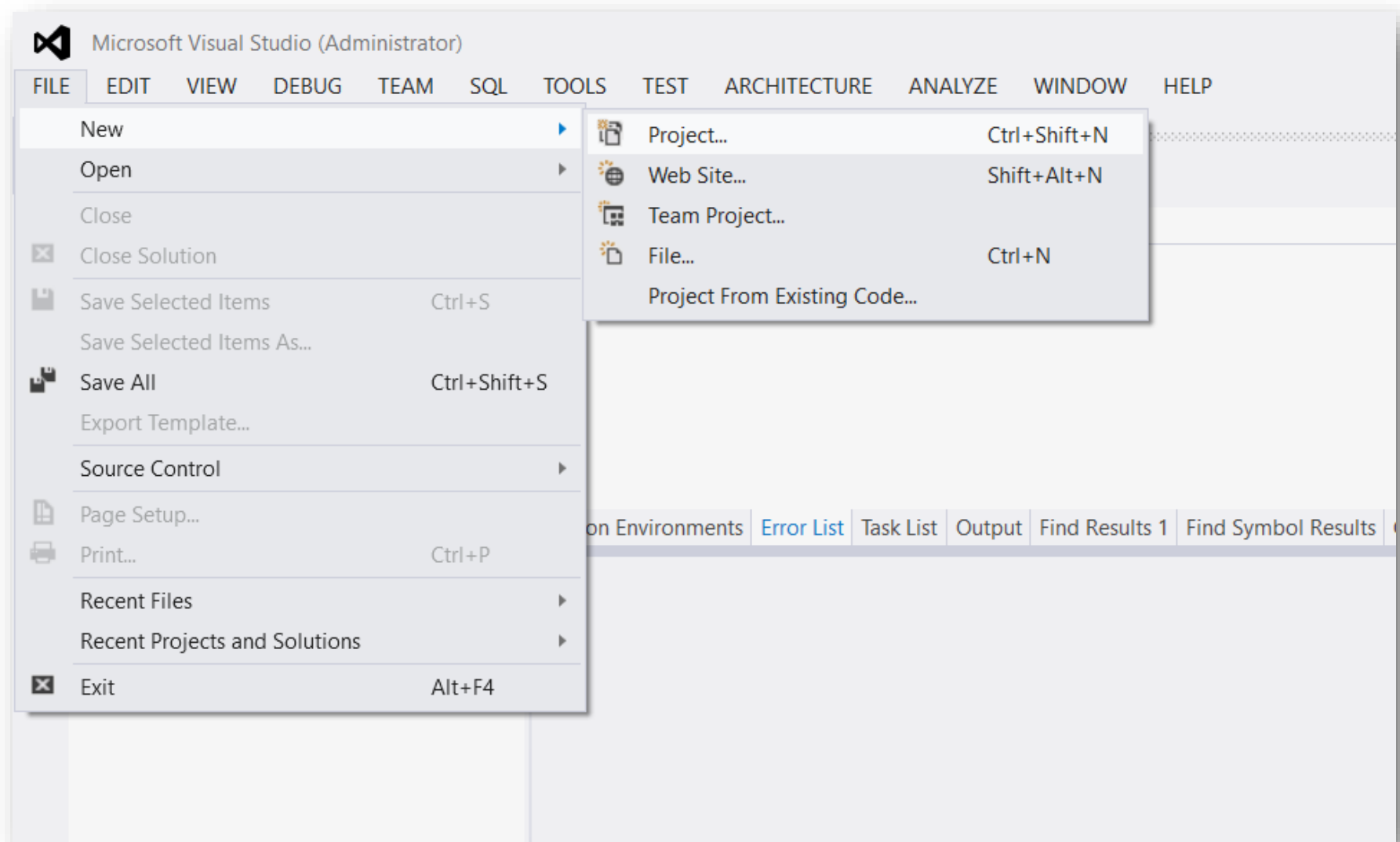
Ako postupovať v prípade zabudnutého hesla?

Aktuálne nastavené heslo je možné zaslať e-mailom priamo cez systém **DreamSpark**, možnosť "Forgot username or password?". Je potrebné použiť e-mailovú adresu zadanú v systéme, teda e-mailovú adresu na server **student**, kam bude heslo následne zaslané.

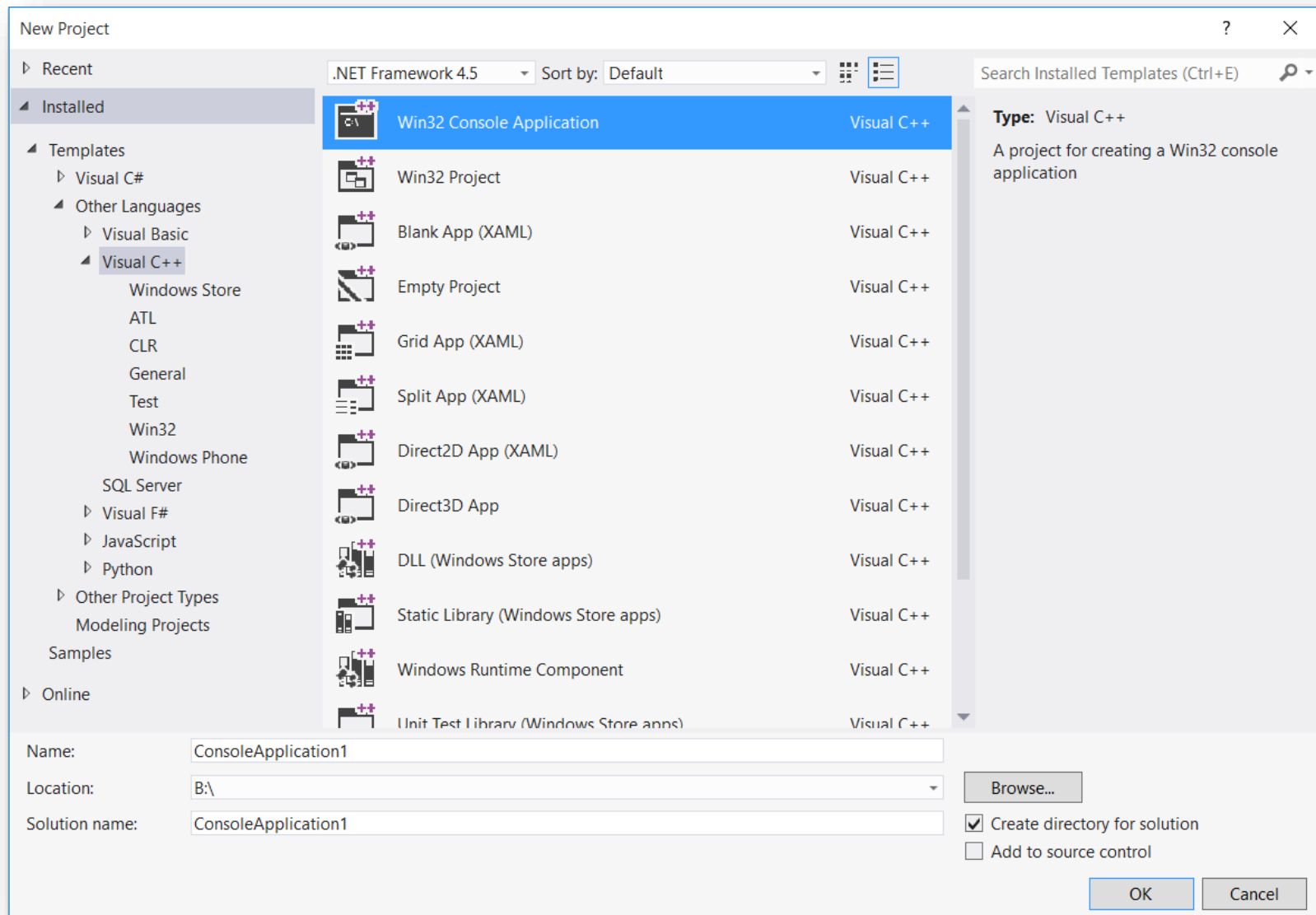
Microsoft Visual Studio 2012



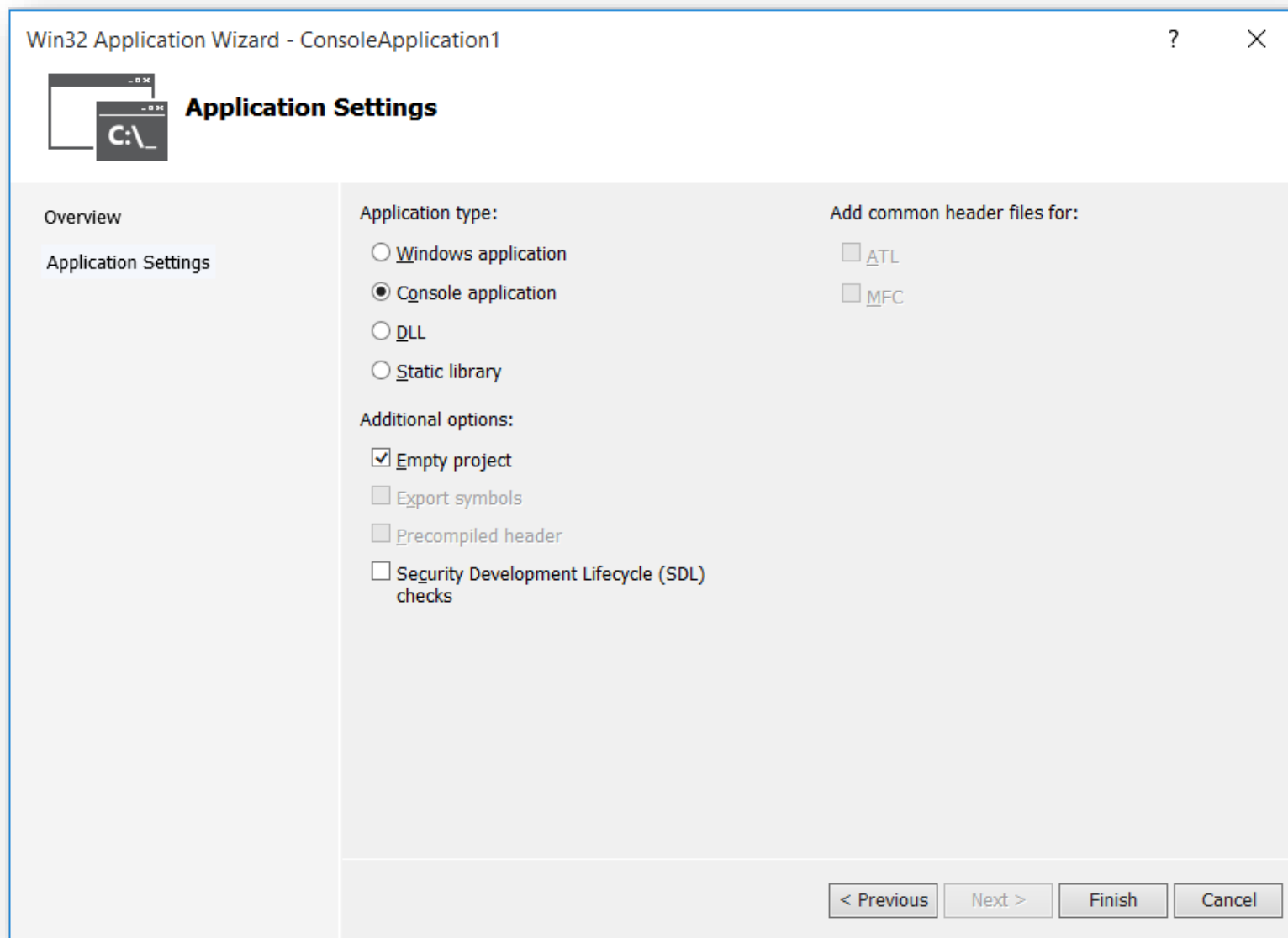
MS VS2012 – Vytvorenie projektu



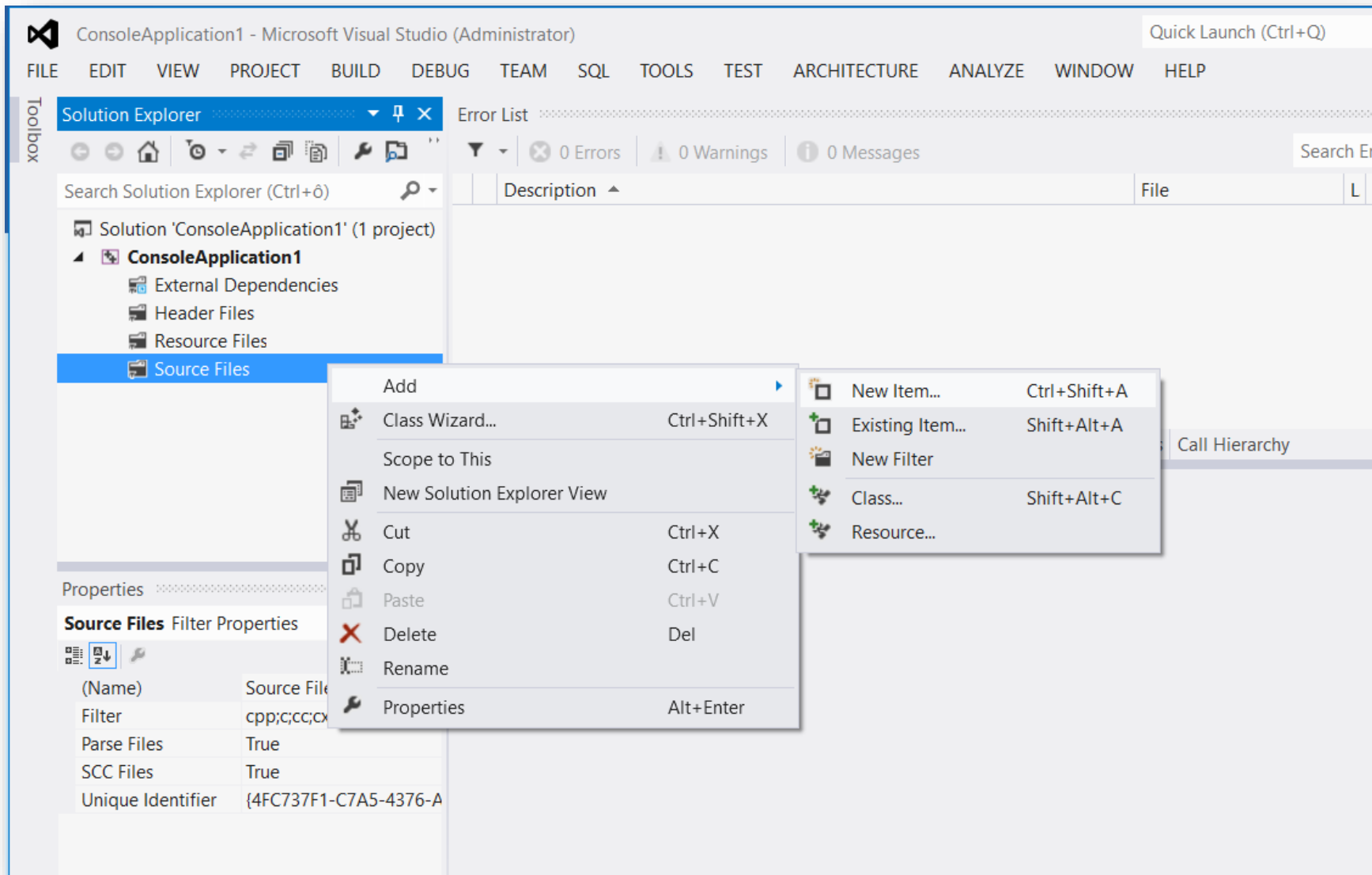
MS VS2012 – Vytvorenie projektu



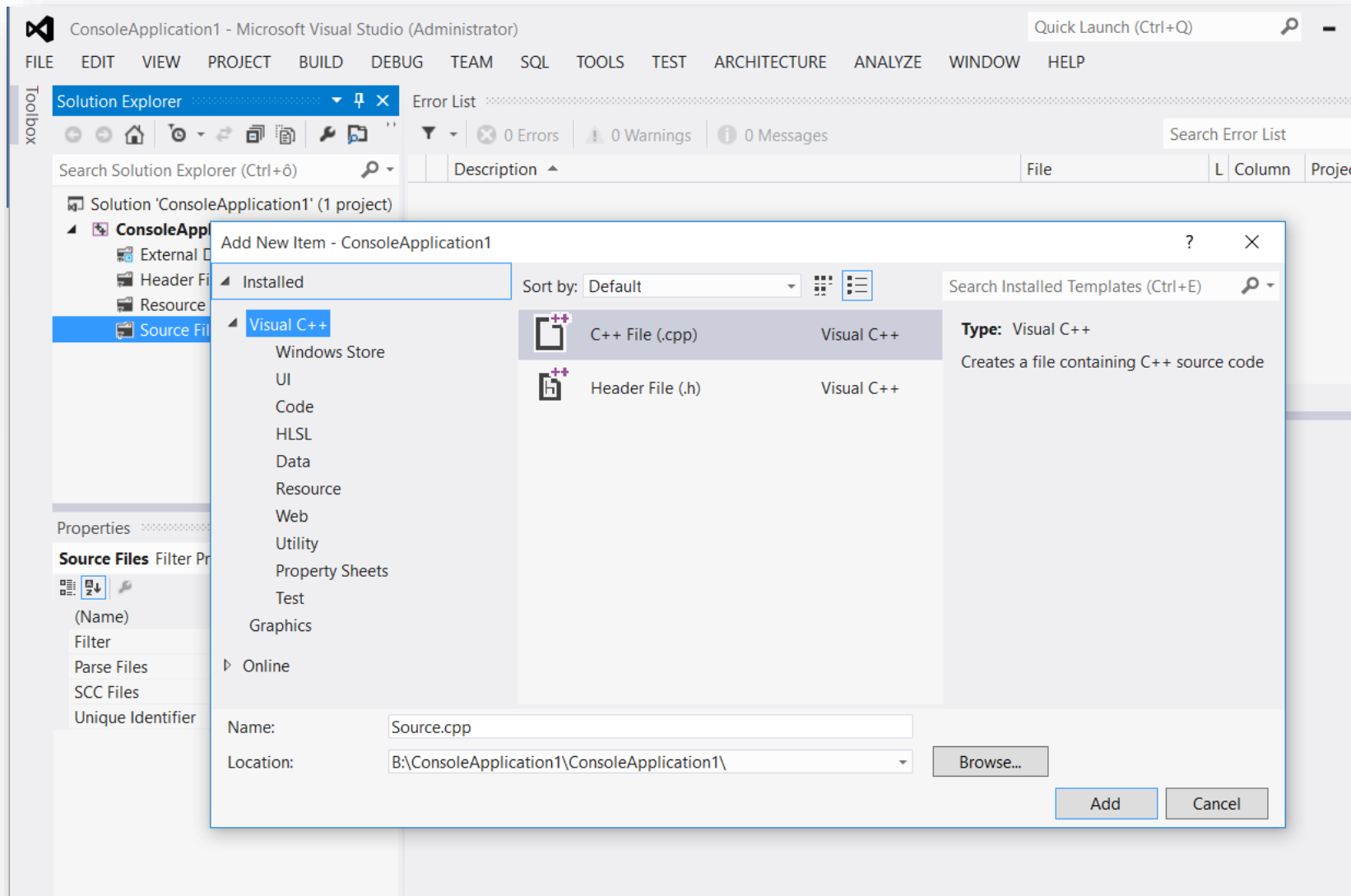
MS VS2012 – Vytvorenie projektu



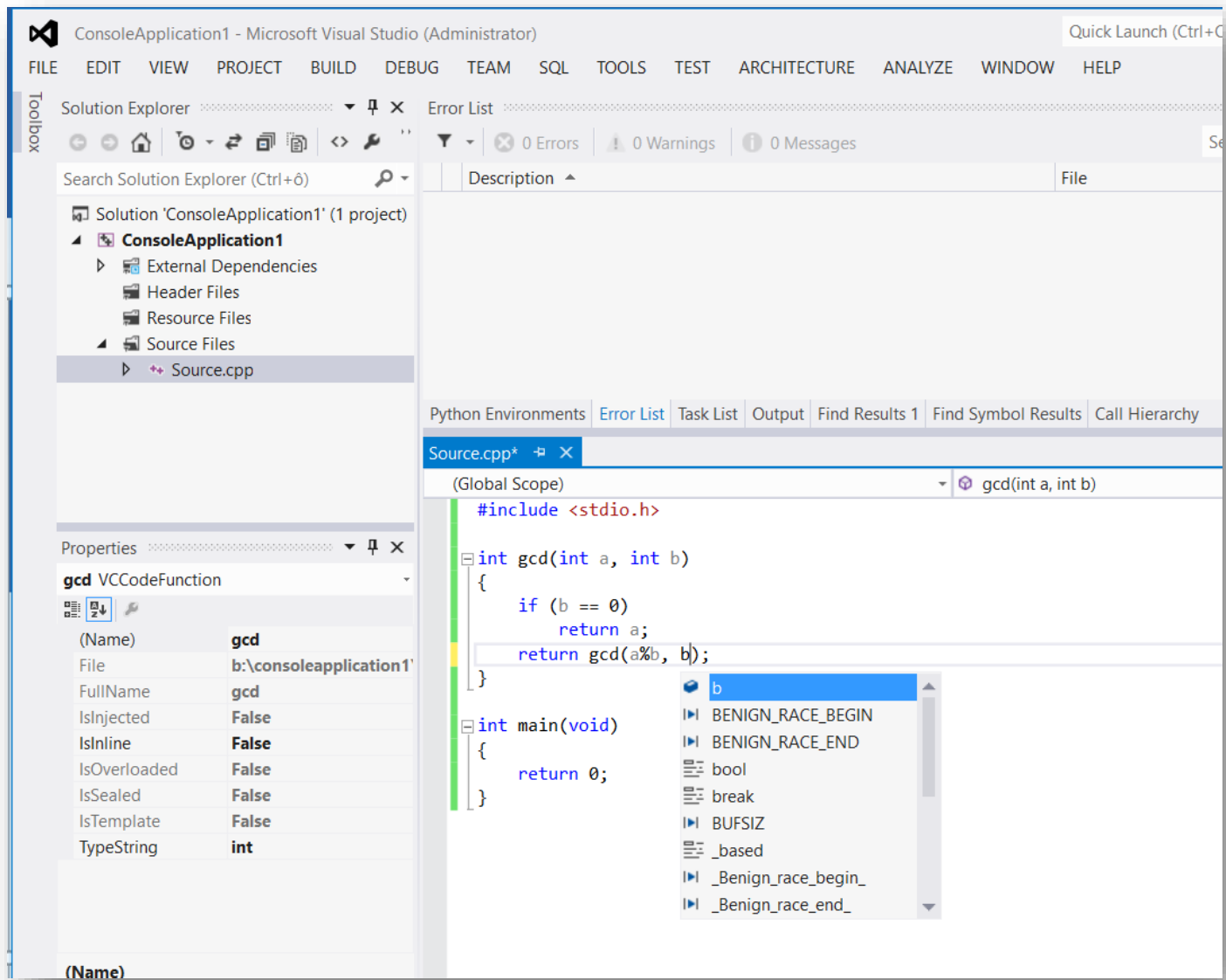
MS VS2012 – Vytvorenie zdrojového súboru



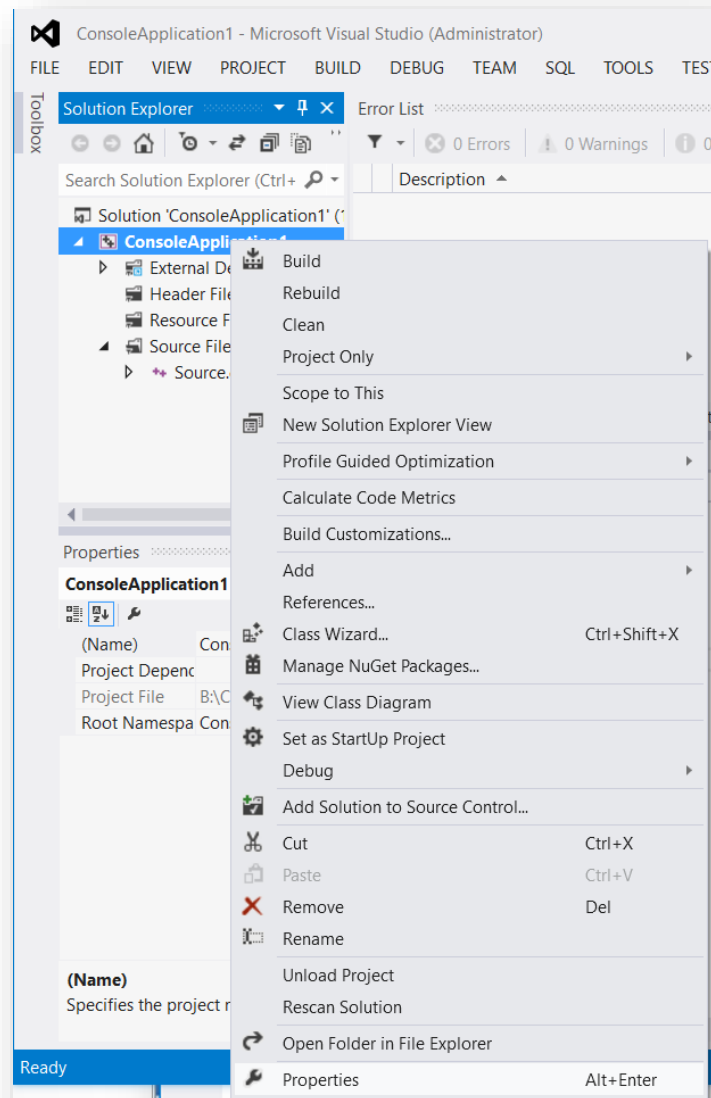
MS VS2012 – Vytvorenie zdrojového súboru



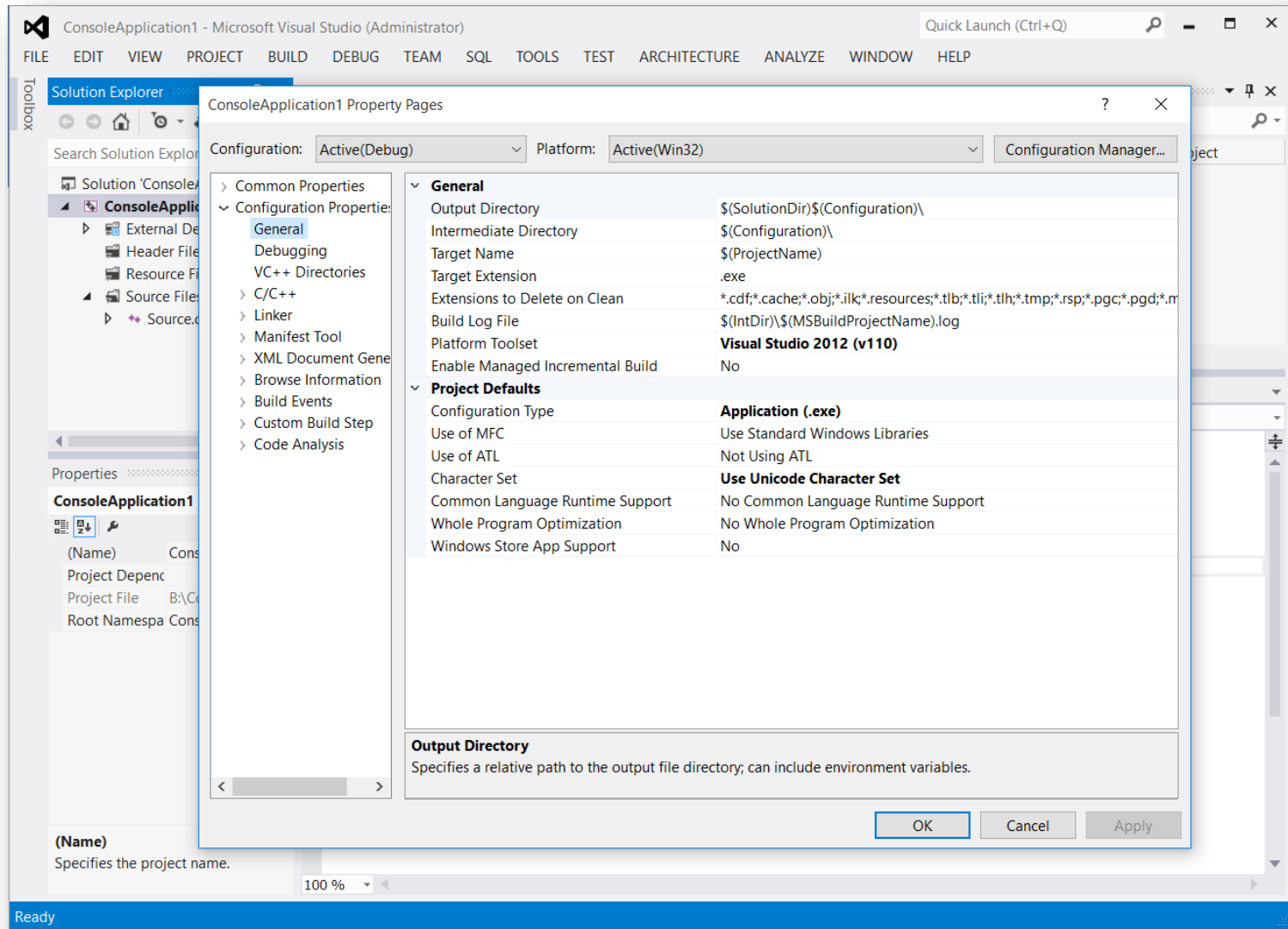
MS VS2012 – Programovanie



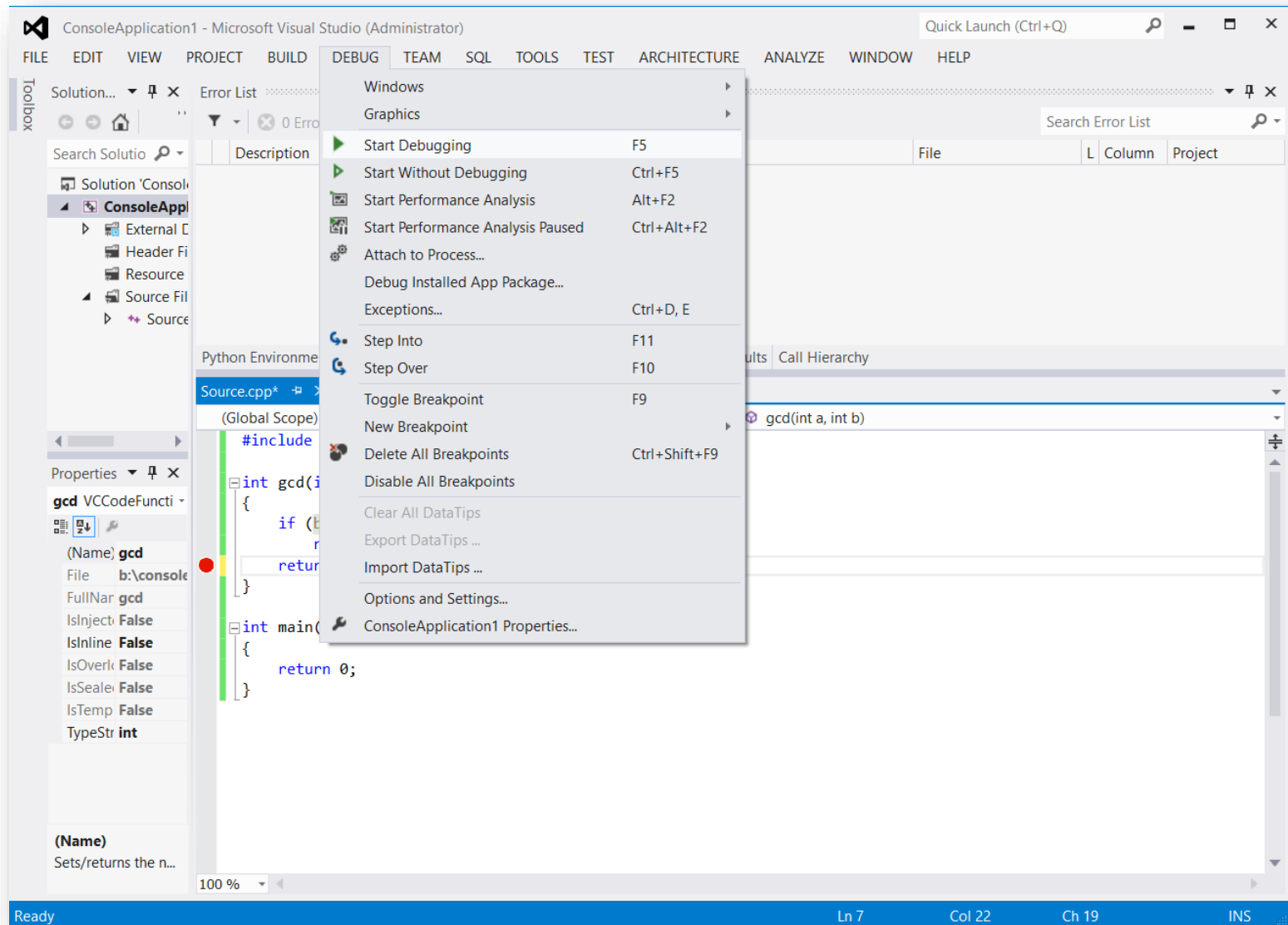
MS VS2012 – Nastavenie projektu



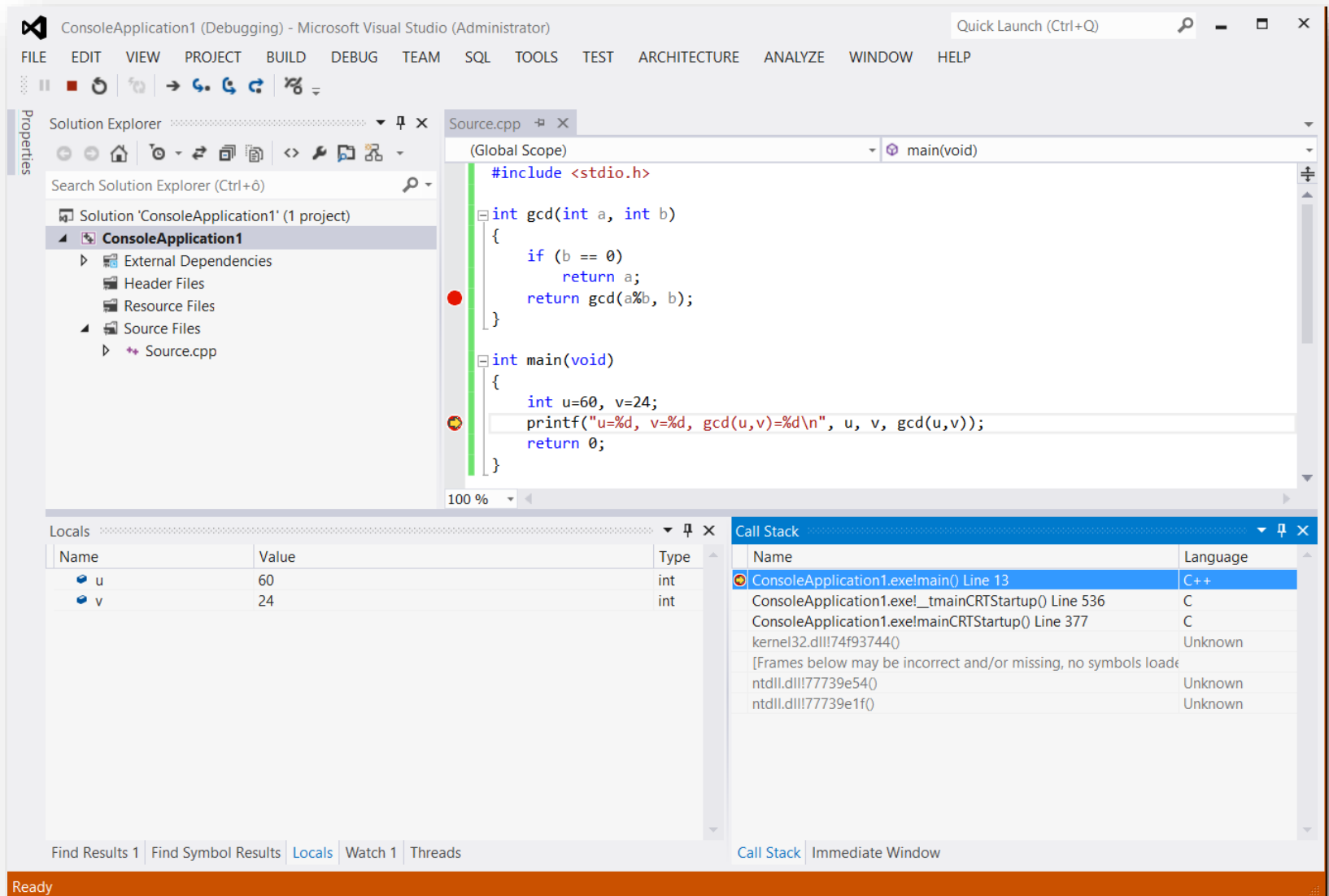
MS VS2012 – Parametre projektu



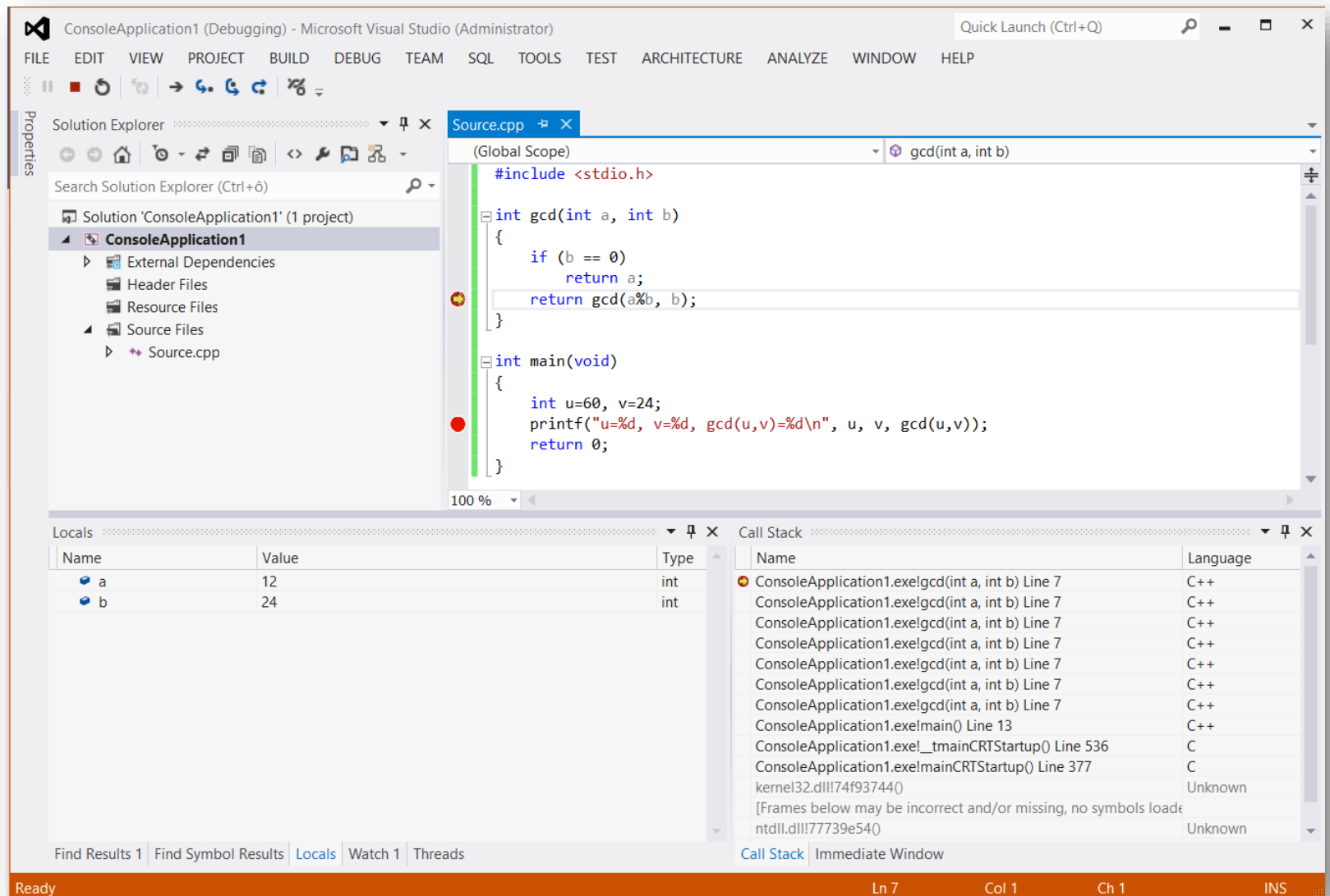
MS VS2012 – Spustenie a Ladenie



MS VS2012 – Ladenie



MS VS2012 – Ladenie



GCC, the GNU Compiler Collection

- Použit' oblíbený textový editor, IDE, ...

```
I source.c (Modified) (c Row 16 Col 1 6:42 Ctrl-K H for help)
#include <stdio.h>

int gcd(int a, int b)
{
    if (b == 0)
        return a;
    return gcd(a%b, b);
}

int main(void)
{
    int u=60, v=24;
    printf("u=%d, v=%d, gcd(u,v)=%d\n", u, v, gcd(u,v));
    return 0;
}
```

GCC, the GNU Compiler Collection

- Najjednoduchší príkaz: kompilácia a zlinkovanie

```
gcc source.c
```

vytvorí **a.out**

- **Prepínače**

- **-c** len kompilácia, vytvorí .o súbor

```
gcc -c source.c
```

- **-g** pre ladenie
- **-Wall** všetky možné warningy
- **-o** výstupný súbor

```
gcc -g -Wall source.c -o vysledok
```


GDB: The GNU Project Debugger

- Najjednoduchší príkaz: spustenie v ladiacom režime

```
gdb ./vysledok
```

```
GNU gdb (Gentoo 7.7.1 p1) 7.7.1
Copyright (C) 2014 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.  Type "show copying"
and "show warranty" for details.
This GDB was configured as "x86_64-pc-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<http://bugs.gentoo.org/>.
Find the GDB manual and other documentation resources online at:
<http://www.gnu.org/software/gdb/documentation/>.
For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from ./vysledok...done.
(gdb) █
```


GDB – Nastavenie breakpointov

```
Reading symbols from ./vysledok...done.
(gdb) list
3      int gcd(int a, int b)
4      {
5          if (b == 0)
6              return a;
7          return gcd(a%b, b);
8      }
9
10     int main(void)
11     {
12         int u=60, v=24;
(gdb)
13         printf("u=%d, v=%d, gcd(u,v)=%d\n", u, v, gcd(u,v));
14         return 0;
15     }
(gdb) break 13
Breakpoint 1 at 0x400593: file source.c, line 13.
(gdb) break gcd
Breakpoint 2 at 0x40055b: file source.c, line 5.
(gdb) 
```

GDB – Spustenie (run), výpis (print) krokovanie (step, next)

```
(gdb) run
Starting program: /home/jozef/test/vysledok
```

```
Breakpoint 1, main () at source.c:13
13         printf("u=%d, v=%d, gcd(u,v)=%d\n", u, v, gcd(u,v));
(gdb) print u
$1 = 60
(gdb) print v
$2 = 24
(gdb) next

Breakpoint 2, gcd (a=60, b=24) at source.c:5
5         if (b == 0)
(gdb) █
```

GDB – pokračovanie po najbližší breakpoint (continue), zásobník volaní (bt)

```
(gdb) run
Starting program: /home/jozef/test/vysledok
```

```
Breakpoint 2, gcd (a=12, b=24) at source.c:5
```

```
5         if (b == 0)
```

```
(gdb)
```

```
7         return gcd(a%b, b);
```

```
(gdb)
```

```
Breakpoint 2, gcd (a=12, b=24) at source.c:5
```

```
5         if (b == 0)
```

```
(gdb) cont
```

```
Continuing.
```

```
Breakpoint 2, gcd (a=12, b=24) at source.c:5
```

```
5         if (b == 0)
```

```
(gdb) bt
```

```
#0  gcd (a=12, b=24) at source.c:5
```

```
#1  0x000000000040057b in gcd (a=12, b=24) at source.c:7
```

```
#2  0x000000000040057b in gcd (a=12, b=24) at source.c:7
```

```
#3  0x000000000040057b in gcd (a=12, b=24) at source.c:7
```

```
#4  0x000000000040057b in gcd (a=60, b=24) at source.c:7
```

```
#5  0x00000000004005a2 in main () at source.c:13
```

```
(gdb) █
```

Makefile

- Predpis pre vytvorenie komplikovanejších projektov
 - `gcc main.c hello.c factorial.c -o hello`
- Štruktúra Makefile (defaultný target je all)
 - target: dependencies
 - [tab] system command
- Ukážka Makefile
 - all:**
 - `gcc main.c hello.c factorial.c -o hello`**

Makefile – ukážka so závislosťami

all: hello

hello: main.o factorial.o hello.o
gcc main.o factorial.o hello.o -o hello

main.o: main.c
gcc -c main.c

factorial.o: factorial.c
gcc -c factorial.c

hello.o: hello.c
gcc -c hello.c

clean:
rm *o hello

Makefile – ukážka s premennými

```
# komentár...
CC=gcc
CFLAGS=-c -Wall

all: hello

hello: main.o factorial.o hello.o
    $(CC) main.o factorial.o hello.o -o hello

main.o: main.c
    $(CC) $(CFLAGS) main.c

factorial.o: factorial.c
    $(CC) $(CFLAGS) factorial.c

hello.o: hello.c
    $(CC) $(CFLAGS) hello.c

clean:
    rm *o hello
```



Nabudúce...

dátum	prednáška	8:00	9:00	cvičenie	obsah
20.3.	6		Čítanie kódu, Hľadanie chýb v kóde	6	Projekt 1: odovzdanie
27.3.	7	Test 3	Riešenie testu 3, Spájané zoznamy	7	
3.4.	8		Tezeus, Bitové operácie	8	Projekt 2 Tezeus a Minotaurus
10.4.	9	Test 4	Riešenie testu 4, Rekurzia, Minotaurus	9	
17.4.	Veľká noc			X	
24.4.	10		Ďalšie prvky jazyka C	10	Projekt 2: odovzdanie, konzultovanie
1.5.	Sviatok			11	
8.5.	Sviatok			12	
9.5.	11		Opakovanie	X	
15.5.	12	Predtermín?		X	